Dynamic Collision Detection using Oriented Bounding Boxes

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: March 2, 1999
Modified: June 27, 2000; November 11, 2002
Modified: December 29, 2002 (incorrect Latex cross product macro was used)

Contents

1 Introduction

The constructions contained in this paper are motivated by the 1996 SIGGRAPH article [1] on oriented bounding boxes (OBBs) and on OBB trees that are used to provide a hierarchical way of deciding if two objects intersect. The computational goal is to minimize the time spent determining if two objects do not intersect. An OBB tree essentially provides a multiscale representation of the object. The root of the tree corresponds to an approximation of the object by a single OBB. The boxes corresponding to the middle levels of the tree represent smaller pieces of the object, thus providing a somewhat better approximation to the object than the root. The leaf nodes of the tree represent the actual geometry of the object. For all practical purposes, the object is a triangular mesh and each leaf node of the OBB tree corresponds to a single triangle in the mesh.

Intersection testing is based on the following theorem from computational geometry:

> Two non–intersecting convex polyhedra can be separated by a plane that is either parallel to a face of one of the polyhedra or that contains an edge from each of the polyhedra.

It is necessary and sufficient to determine whether or not two convex polyhedra intersect by examining the intersections of the projections of the polyhedra on lines that are perpendicular to the planes described in the theorem. If the minimal intervals containing the projections of the polyhedra onto one of these lines do not intersect, then the polyhedra themselves do not intersect. In this case the line is said to be a *separating axes*. Testing for intersection amounts to processing each of the potential separating axes by projecting the objects onto an axis and testing for intersection of the minimal intervals containing the projections of the objects. If a separating axis is found, the remaining ones of course are not processed. For a 3D environment that is not heavily populated, the expectation is that at any point in time, most objects are not intersecting. The "quick out" provided when a separating axis is found early in the list of potential separating axes helps to minimize computational expense of the collision system. The implementation for determining nonintersection of two triangles in 3D using the method of separation axes turns out to be faster than the method in [2]. The algorithm in that paper attempts to determine overlap of projected intervals along the line on which two noncoplanar triangles have to intersect. It requires divisions to compute the intervals whereas the separation axis tests do not.

The OBB tree consists of OBBs as well as triangles at the leaf nodes. Intersection testing therefore involves comparisons of OBBs to OBBs, triangles to OBBS, and triangles to triangles. For two OBBs, there are 15 potential separating axes: 3 for the independent faces of the first OBB, 3 for the independent faces of the second OBB, and 9 generated by an edge from the first OBB and an edge from the second OBB. For triangle and OBB, there are 13 potential separating axes: 1 for the triangle face, 3 for the independent faces of the OBB, and 9 generated by an edge from the triangle and an edge from the OBB. For two triangles, the problem is slightly more complicated. If the two triangles are parallel but not coplanar, then a separating axis whose direction is normal to the triangles will separate the two triangles. If the two triangles are coplanar, then there are 6 additional potential separating axes, each one generated by the common normal and an edge from either the first triangle or second triangle. If the two triangles are not parallel, then there are 11 potential separating axes: 1 for the first triangle face, 1 for the second triangle face, and 9 generated by an edge from the first triangle and an edge from the second triangle.

The ideas in [1] are formulated for intersection testing of objects as if they are stationary, but the construction also applies when the objects are moving. In particular, when each object has a constant velocity during the specified time interval, the extension is mathematically straightforward. When object motion is constrained generally by the system of ordinary differential equations $d\vec{x}/dt = \vec{V}(t, \vec{x})$ where $\vec{V}$ is the velocity vector

field that is (possibly) dependent on both current time and current position, a numerical integration of the differential equations can be applied during the specified time interval. The simplest method to apply is Euler's method. For each time step, the object is assumed to have constant velocity during that step. The methods for handling collision of objects with constant velocities can then be applied during that time step. If more positional accuracy is desired, a higher order numerical integrator can be used to determine various positions during the time interval. The difference between consecutive positions can be used as the constant velocity vector for that time step. Of particular interest is the case when the objects have constant linear velocities and constant angular velocities. The differential equations are $d\vec{x}/dt = \vec{V} + \vec{W} \times (\vec{x} - \vec{K})$ where $\vec{V}$ is the constant linear velocity and the axis of rotation is $\vec{K} + t\vec{W}$ where $\vec{W}$ is the constant angular velocity and whose length is the angular speed. The motion is $\vec{x}(t) = \vec{K} + t\vec{V} + R(t, \vec{W})(\vec{x}_0 - \vec{K})$ where $R(t, \vec{W})$ is a rotation matrix about the axis $\vec{K} + t\vec{W}$. While it is possible to perform intersection testing using the closed form solution for position, it is not recommended. The closed form leads to a test equivalent to showing the minimum of a function containing sinusoidals and polynomials is positive for the specified time interval. Since minimization algorithms are iterative and since the trigonometric function evaluations are expensive, it is better to numerically solve the differential equation (also iterative) and avoid the trigonometric function calls.

Intersection testing for moving objects translates to intersection testing of the moving intervals of projection on the potential separating axes. If the two time–dependent projection intervals are $[u_0(t), u_1(t)]$ and $[v_0(t), v_1(t)]$, then nonintersection for $t \in [0, T]$ is equivalent to showing: $u_1(t) < v_0(t)$ for all $t \in [0, T]$ *or* $v_1(t) < u_0(t)$ for all $t \in [0, T]$.

Of additional interest for moving objects is the ability to determine the first time and first point of intersection for the objects during a specified time interval. These quantities can be determined by processing all the potential separating axes and computing the last time that a separating axis exists. At this time the two objects are just touching (no interpenetration). The various quantities computed for the separation tests provide enough information to reconstruct a first point of intersection (if this point is unique) or to reconstruct one of the points of intersection (if not unique). It is also possible to extract all points of intersection, but this comes with an additional computational cost.

The paper is structured as follows. There is a section for each of the comparisons: two OBBS, one triangle and one OBB, two triangles. Each section provides the conditions for the objects to be separated by a separating axis, both for static objects and for moving objects. Each section also contains the reconstruction algorithm for the first time and first point of intersection when the objects have constant velocities. A separate section contains the details for how to test for collisions in the case of general motion (using differential equation solvers).

A section is provided to illustrate one method for automatically generating oriented bounding box trees. The algorithm fits a mesh of triangles with an OBB using either an analysis of a covariance matrix of the vertices and triangles in that mesh or a minimum volume bounding box fit. Once the OBB is computed, a basic splitting algorithm is used to partition the mesh into two submeshes. The tree generation is recursive in that the algorithm is applied to each of the two submeshes. The result is a binary tree of OBBs. The application has the option of limiting how deep a tree is built by specifying how many triangles are to occur at a leaf node. The default is one triangle. If the leaf nodes have multiple triangles, then only the representing OBB is stored in the tree. The idea is to reduce computation time at the expense of accuracy.

The last section presents an implementation of a simple dynamic collision detection system. Given two OBB trees, the problem is to traverse them simultaneously and test/find intersections. The application can specify how deep in the tree to traverse, again to reduce computation time at the expense of accuracy. Once an intersection is predicted, the first time and first point of intersection as well as other information is given to

the application via callbacks associated with the objects. This scheme makes the collision detection effectively transparent to the application. The application can concentrate solely on the physics of the response, for example, arranging for objects to bounce off walls with the proper angle and angular momentum.

2 Oriented Bounding Boxes

In the following discussion, all vectors are in $\mathbb{R}^3$. An *oriented bounding box* is defined by a *center* $\vec{C}$, a set of right–handed orthonormal *axes* $\vec{A}_0$, $\vec{A}_1$, and $\vec{A}_2$, and a set of *extents* $a_0 > 0$, $a_1 > 0$, and $a_2 > 0$. As a solid box, the OBB is represented by

$$\left\{ \vec{C} + \sum_{i=0}^{2} x_i \vec{A}_i \ : \ |x_i| \le |a_i| \ \text{ for all } i \right\}$$

and the eight vertices of the box are

$$\vec{C} + \sum_{i=0}^{2} \sigma_i a_i \vec{A}_i$$

where $|\sigma_i| = 1$ for all i.

2.1 Separation of OBBs

Let the first OBB have center $\vec{C}_0$, axes $\vec{A}_0$, $\vec{A}_1$, $\vec{A}_2$, and extents a_0, a_1, a_2. Let the second OBB have center $\vec{C}_1$, axes $\vec{B}_0$, $\vec{B}_1$, $\vec{B}_2$, and extents b_0, b_1, b_2. The potential separating axes are of the form $\vec{C}_0 + s\vec{L}$ where $\vec{L}$ is one of $\vec{A}_i$, $\vec{B}_j$, or $\vec{A}_i \times \vec{B}_j$ for $i = 0, 1, 2$ and $j = 0, 1, 2$.

Projection of a point $\vec{P}$ onto line $\vec{C}_0 + s\vec{L}$ relative to the line origin $\vec{C}_0$ is

$$\frac{\vec{L} \cdot (\vec{P} - \vec{C}_0)}{\vec{L} \cdot \vec{L}} \vec{L}.$$

The distance of the projection from the line origin is

$$\text{ProjDist}(\vec{P}) = \frac{\vec{L} \cdot (\vec{P} - \vec{C}_0)}{\vec{L} \cdot \vec{L}}.$$

The projection distances of the first OBB's vertices relative to the line origin $\vec{C}_0$ are

$$\text{ProjDist}\left(\vec{C}_0 + \sum_{i=0}^{2} \sigma_i a_i \vec{A}_i \right) = \sum_{i=0}^{2} \sigma_i a_i \frac{\vec{L} \cdot \vec{A}_i}{\vec{L} \cdot \vec{L}}.$$

The minimum length interval containing all eight projection distances has center $K_0 = 0$ and radius

$$r_0 = \frac{\sum_{i=0}^{2} a_i \text{Sign}(\vec{L} \cdot \vec{A}_i) \vec{L} \cdot \vec{A}_i}{\vec{L} \cdot \vec{L}}.$$

The projection distances of the second OBB's vertices relative to the line origin $\vec{C}_0$ are

$$\text{ProjDist}\left(\vec{C}_0 + \sum_{i=0}^{2} \sigma_i a_i \vec{A}_i\right) = \frac{\vec{L}\cdot\vec{D}}{\vec{L}\cdot\vec{L}} + \sum_{i=0}^{2} \sigma_i b_i \frac{\vec{L}\cdot\vec{B}_i}{\vec{L}\cdot\vec{L}}$$

where $\vec{D} = \vec{C}_1 - \vec{C}_0$. The minimum length interval containing all eight projection distances has center $K_1 = \vec{L}\cdot\vec{D}$ and radius

$$r_1 = \frac{\sum_{i=0}^{2} b_i \text{Sign}(\vec{L}\cdot\vec{B}_i)\vec{L}\cdot\vec{B}_i}{\vec{L}\cdot\vec{L}}.$$

The two projected intervals do not intersect whenever the distance between interval centers is larger than the sum of the radii of the intervals: $|K_1 - K_0| > r_0 + r_1$. Each of the quantities involved has in its denominator $\vec{L}\cdot\vec{L}$. The division is therefore not necessary, so the nonintersection test is

$$|\vec{L}\cdot\vec{D}| > \sum_{i=0}^{2} a_i \text{Sign}(\vec{L}\cdot\vec{A}_i)\vec{L}\cdot\vec{A}_i + \sum_{i=0}^{2} b_i \text{Sign}(\vec{L}\cdot\vec{B}_i)\vec{L}\cdot\vec{B}_i.$$

In the remainder of the discussion we use notation

$$\begin{aligned} R &= |\vec{L}\cdot\vec{D}| \\ R_0 &= \textstyle\sum_{i=0}^{2} a_i \text{Sign}(\vec{L}\cdot\vec{A}_i)\vec{L}\cdot\vec{A}_i \\ R_1 &= \textstyle\sum_{i=0}^{2} b_i \text{Sign}(\vec{L}\cdot\vec{B}_i)\vec{L}\cdot\vec{B}_i \end{aligned} \tag{1}$$

to represent the various distances without the division by the squared length of $\vec{L}$. The nonintersection test for an axis is $R > R_0 + R_1$.

The axes of the second OBB can be written as combinations of axes of the first:

$$\vec{B}_i = c_{0i}\vec{A}_0 + c_{1i}\vec{A}_1 + c_{2i}\vec{A}_2$$

for $i = 0, 1, 2$. Let A be the matrix whose columns are the $\vec{A}_i$, let B be the matrix whose columns are the $\vec{B}_i$, and let C be the matrix whose entries are c_{ij}; then $B = AC$ in which case $C = A^T B$ where the superscript T indicates the transpose operation. The components of C are just $c_{ij} = \vec{A}_i \cdot \vec{B}_j$. Similarly, the axes of the first box can be written as linear combinations of axes of the second box:

$$\vec{A}_i = c_{i0}\vec{B}_0 + c_{i1}\vec{B}_1 + c_{i2}\vec{B}_2$$

for $i = 0, 1, 2$. These relationships allow us to compute the various dot products between the separating axis directions and the box axes in terms of the c_{ij} and extents. In particular, the nonintersection tests involve various triple scalar products involving the box axes:

$$\vec{A}_{i_0} \cdot \vec{A}_{i_1} \times \vec{B}_j = \text{Sign}(i_0, i_1) c_{i_2 j} \;\text{ and }\; \vec{B}_{j_0} \cdot \vec{A}_i \times \vec{B}_{j_1} = \text{Sign}(j_1, j_0) c_{i j_2} \tag{2}$$

where $\text{Sign}(0,1) = \text{Sign}(1,2) = \text{Sign}(2,0) = +1$ and $\text{Sign}(1,0) = \text{Sign}(2,1) = \text{Sign}(0,2) = -1$. Equations (1)

and (2) lead to the table below.

$\vec{L}$	R_0	R_1	R
$\vec{A}_0$	a_0	$b_0\|c_{00}\| + b_1\|c_{01}\| + b_2\|c_{02}\|$	$\|\vec{A}_0 \cdot \vec{D}\|$
$\vec{A}_1$	a_1	$b_0\|c_{10}\| + b_1\|c_{11}\| + b_2\|c_{12}\|$	$\|\vec{A}_1 \cdot \vec{D}\|$
$\vec{A}_2$	a_2	$b_0\|c_{20}\| + b_1\|c_{21}\| + b_2\|c_{22}\|$	$\|\vec{A}_2 \cdot \vec{D}\|$
$\vec{B}_0$	$a_0\|c_{00}\| + a_1\|c_{10}\| + a_2\|c_{20}\|$	b_0	$\|\vec{B}_0 \cdot \vec{D}\|$
$\vec{B}_1$	$a_0\|c_{01}\| + a_1\|c_{11}\| + a_2\|c_{21}\|$	b_1	$\|\vec{B}_1 \cdot \vec{D}\|$
$\vec{B}_2$	$a_0\|c_{02}\| + a_1\|c_{12}\| + a_2\|c_{22}\|$	b_2	$\|\vec{B}_2 \cdot \vec{D}\|$
$\vec{A}_0 \times \vec{B}_0$	$a_1\|c_{20}\| + a_2\|c_{10}\|$	$b_1\|c_{02}\| + b_2\|c_{01}\|$	$\|c_{10}\vec{A}_2 \cdot \vec{D} - c_{20}\vec{A}_1 \cdot \vec{D}\|$
$\vec{A}_0 \times \vec{B}_1$	$a_1\|c_{21}\| + a_2\|c_{11}\|$	$b_0\|c_{02}\| + b_2\|c_{00}\|$	$\|c_{11}\vec{A}_2 \cdot \vec{D} - c_{21}\vec{A}_1 \cdot \vec{D}\|$
$\vec{A}_0 \times \vec{B}_2$	$a_1\|c_{22}\| + a_2\|c_{12}\|$	$b_0\|c_{01}\| + b_1\|c_{00}\|$	$\|c_{12}\vec{A}_2 \cdot \vec{D} - c_{22}\vec{A}_1 \cdot \vec{D}\|$
$\vec{A}_1 \times \vec{B}_0$	$a_0\|c_{20}\| + a_2\|c_{00}\|$	$b_1\|c_{12}\| + b_2\|c_{11}\|$	$\|c_{20}\vec{A}_0 \cdot \vec{D} - c_{00}\vec{A}_2 \cdot \vec{D}\|$
$\vec{A}_1 \times \vec{B}_1$	$a_0\|c_{21}\| + a_2\|c_{01}\|$	$b_0\|c_{12}\| + b_2\|c_{10}\|$	$\|c_{21}\vec{A}_0 \cdot \vec{D} - c_{01}\vec{A}_2 \cdot \vec{D}\|$
$\vec{A}_1 \times \vec{B}_2$	$a_0\|c_{22}\| + a_2\|c_{02}\|$	$b_0\|c_{11}\| + b_1\|c_{10}\|$	$\|c_{22}\vec{A}_0 \cdot \vec{D} - c_{02}\vec{A}_2 \cdot \vec{D}\|$
$\vec{A}_2 \times \vec{B}_0$	$a_0\|c_{10}\| + a_1\|c_{00}\|$	$b_1\|c_{22}\| + b_2\|c_{21}\|$	$\|c_{00}\vec{A}_1 \cdot \vec{D} - c_{10}\vec{A}_0 \cdot \vec{D}\|$
$\vec{A}_2 \times \vec{B}_1$	$a_0\|c_{11}\| + a_1\|c_{01}\|$	$b_0\|c_{22}\| + b_2\|c_{20}\|$	$\|c_{01}\vec{A}_1 \cdot \vec{D} - c_{11}\vec{A}_0 \cdot \vec{D}\|$
$\vec{A}_2 \times \vec{B}_1$	$a_0\|c_{12}\| + a_1\|c_{02}\|$	$b_0\|c_{21}\| + b_1\|c_{20}\|$	$\|c_{02}\vec{A}_1 \cdot \vec{D} - c_{12}\vec{A}_0 \cdot \vec{D}\|$

Table 1. Values for R, R_0, and R_1 for the nonintersection test $R > R_0 + R_1$ for two OBBs.

2.2 Testing for Intersection of OBBs

2.2.1 Stationary

Testing for intersection amounts to processing each axis of the 15 potential separating axes. If a separating axis is found, the remaining ones of course are not processed. The various entries c_{ij} and $|c_{ij}|$ are computed only when needed. This avoids unnecessary calculations in the event that a separating axis is found quickly and some of the c_{ij} do not need to be computed. The basic separating axis test involves computing R_0, R_1, and R and then testing for nonintersection by comparing $R > R_0 + R_1$.

2.2.2 Constant Velocities

The code for stationary boxes needs to be modified only slightly to handle the case of constant velocities. The velocity of the second box is subtracted from the velocity of the first box so that all calculations are done relative to a stationary first box. If the box velocities are $\vec{V}_0$ and $\vec{V}_1$, define the relative velocity

to be $\vec{W} = \vec{V}_1 - \vec{V}_0$. Let the initial time and final time of the calculations be $t = 0$ and $t = T$. Let $\vec{D}(t) = \vec{C}_1 - \vec{C}_0 + t\vec{W}$ for $t \in [0, T]$.

Consider the separating axis $\vec{C}_0 + s\vec{L}$. The interval values R_0 and R_1 are independent of t and can be calculated as in the stationary case. The quantity R is dependent on time. The nonintersection test is to show $R(t) > R_0 + R_1$ for nonintersection for all $t \in [0, T]$. One potential problem is that the moving projected interval may start out on one side of the stationary interval, then pass through it to the other side during the time period. However, because of the linear velocity it is enough for nonintersection to show that $R(0) > R_0 + R_1$, $R(T) > R_0 + R_1$, and $\text{Sign}(\vec{L} \cdot \vec{D}_0) = \text{Sign}(\vec{L} \cdot \vec{D}_1)$.

Abstractly the problem amounts to showing that $|p + tw| > r > 0$ for all $t \in [0, T]$. Pseudocode is

```
if ( p > r )
{
    if ( p+T*w > r )
        return no_intersection;
}
else if ( p < -r )
{
    if ( p+T*w < -r )
        return no_intersection;
}
```

If any of the 15 axes separates the two boxes over the full time interval, then no intersection occurs. However, if none of the 15 axes separates the boxes for the full time, it is still possible that the boxes are not intersecting. The motion complicates things slightly. If you view the boxes along the direction of motion, the boxes appear to be stationary. That is, their projections onto any plane perpendicular to $\vec{W}$ are not moving. If the projections are not intersecting, then the two boxes do not intersect over the full time interval. If the projections do intersect, then the boxes intersect. Testing the projections for nonintersection requires up to 6 additional axis tests, the axes being $\vec{W} \times \vec{A}_i$ and $\vec{W} \times \vec{B}_i$ for $0 \leq\leq 2$. Define $\alpha_i = \vec{W} \cdot \vec{A}_i$ and $\beta_i = \vec{W} \cdot \vec{B}_i$ for $0 \leq i \leq 2$. Table 1 is extended by the following to handle the cases with motion,

$\vec{L}$	R_0	R_1	R
$\vec{W} \times \vec{A}_0$	$a_1\lvert\alpha_2\rvert + a_2\lvert\alpha_1\rvert$	$\sum_{i=0}^{2} b_i\lvert c_{1i}\alpha_2 - c_{2i}\alpha_1\rvert$	$\lvert\vec{A}_0 \cdot \vec{W} \times \vec{D}\rvert$
$\vec{W} \times \vec{A}_1$	$a_0\lvert\alpha_2\rvert + a_2\lvert\alpha_0\rvert$	$\sum_{i=0}^{2} b_i\lvert c_{0i}\alpha_2 - c_{2i}\alpha_0\rvert$	$\lvert\vec{A}_1 \cdot \vec{W} \times \vec{D}\rvert$
$\vec{W} \times \vec{A}_2$	$a_0\lvert\alpha_1\rvert + a_1\lvert\alpha_0\rvert$	$\sum_{i=0}^{2} b_i\lvert c_{0i}\alpha_1 - c_{1i}\alpha_0\rvert$	$\lvert\vec{A}_2 \cdot \vec{W} \times \vec{D}\rvert$
$\vec{W} \times \vec{B}_0$	$\sum_{i=0}^{2} a_i\lvert c_{i1}\beta_2 - c_{i2}\beta_1\rvert$	$b_1\lvert\beta_2\rvert + b_2\lvert\beta_1\rvert$	$\lvert\vec{B}_0 \cdot \vec{W} \times \vec{D}\rvert$
$\vec{W} \times \vec{B}_1$	$\sum_{i=0}^{2} a_i\lvert c_{i0}\beta_2 - c_{i2}\beta_0\rvert$	$b_0\lvert\beta_2\rvert + b_2\lvert\beta_0\rvert$	$\lvert\vec{B}_1 \cdot \vec{W} \times \vec{D}\rvert$
$\vec{W} \times \vec{B}_2$	$\sum_{i=0}^{2} a_i\lvert c_{i0}\beta_1 - c_{i1}\beta_0\rvert$	$b_0\lvert\beta_1\rvert + b_1\lvert\beta_0\rvert$	$\lvert\vec{B}_2 \cdot \vec{W} \times \vec{D}\rvert$

Table 1'. Values for R, R_0, and R_1 for the nonintersection test $R > R_0 + R_1$ for two OBBs in the direction of motion.

2.3 Finding an Intersection of OBBs

Testing for intersection is a boolean operation. The algorithm returns true if there is an intersection, false if not. No information is provided about where an intersection occurs (there may be many such points).

In the case of two stationary objects that are intersection, the region of intersection can be computed with great pain. This is the realm of computation solid geometry. For a dynamic system, the more interesting case is to have two moving objects that are initially not intersecting, but then do intersect during the specified time interval. Of interest to an application is the first time of intersection and a point (or points) of intersection at that time. The following construction provides a point of intersection. In the cases of vertex–to–vertex, vertex–to–edge, vertex–to–face, or edge–to–edge (transversely), the intersection point (at first time) is unique. In the other cases of edge–to–face or face–to–face, the intersection is not unique, but the construction provides one of the intersection points.

2.3.1 Finding the First Time of Intersection

Given that the two objects do not intersect at time $t = 0$, but do intersect at some later time, a simple modification of the algorithm for testing for an intersection provides the first time of intersection. The first time is computed as the *maximum* time $T > 0$ for which there is at least one separating axis for any $t \in [0, T)$, but for which no separating axis exists at time T. The idea is to test each potential separating axis and keep track of the time at which the intervals of projection intersect for the first time. The largest such time is the first time at which the OBBs intersect. Also, it is important to keep track of which side of $[-R_0, R_0]$ the other interval intersects. Finally, knowing the separating axis associated with the maximum time T allows us to reconstruct a point of intersection.

2.3.2 Finding a Point of Intersection

If T is the first time of intersection, the problem is now to find a point P in the intersection of the two OBBs at that time. We need to solve

$$\sum_{i=0}^{2} x_i \vec{A}_i = \vec{D} + \sum_{j=0}^{2} y_j \vec{B}_j \tag{3}$$

where $\vec{D} = (\vec{C}_1 + T\vec{V}_1) - (\vec{C}_0 + T\vec{V}_0)$, and for x_i with $|x_i| \le a_i$, $i = 0, 1, 2$, and for y_j with $|y_j| \le b_j$, $j = 0, 1, 2$.

Last Separating Axis $\vec{A}_i$. If the separating axis at time T is $\vec{A}_i$, then the intersection must occur on one of the two faces perpendicular to $\vec{A}_i$. Dotting equation (3) with $\vec{A}_i$ yields

$$x_i = \vec{A}_i \cdot \vec{D} + \sum_{j=0}^{2} c_{ij} y_j.$$

If $\vec{A}_i \cdot \vec{D} > 0$, then $\vec{A}_i \cdot \vec{D} = R_0 + R_1$ since the two intervals intersect at the right endpoint of $[-R_0, R_0]$. If $\vec{A}_i \cdot \vec{D} < 0$, then $\vec{A}_i \cdot \vec{D} = -(R_0 + R_1)$ since the two intervals intersect at the left endpoint of $[-R_0, R_0]$.

Thus, $\vec{A}_i \cdot \vec{D} = \sigma(R_0 + R_1)$ where $|\sigma| = 1$ and

$$\begin{aligned} x_i &= \sigma(R_0 + R_1) + \textstyle\sum_{j=0}^{2} c_{ij} y_j \\ &= \sigma(a_i + \textstyle\sum_{j=0}^{2} b_j |c_{ij}|) + \textstyle\sum_{j=0}^{2} c_{ij} y_j \\ 0 &= \textstyle\sum_{j=0}^{2} |c_{ij}|(b_j + \sigma \mathrm{Sign}(c_{ij}) y_j) + (a_i - \sigma x_i). \end{aligned} \tag{4}$$

Since $|y_j| \le b_j$ and $|\sigma \mathrm{Sign}(c_{ij})| \le 1$, it must be that $b_j + \sigma \mathrm{Sign}(c_{ij}) y_j \ge 0$. Similarly, $a_i - \sigma x_i \ge 0$ in which case $x_i = \sigma a_i$. If $c_{ij} \ne 0$, then $y_j = -\sigma \mathrm{Sign}(c_{ij}) b_j$ is required to make the sum in equation (4) zero.

If any $c_{ij} = 0$, then the sum in equation (4) places no restriction on y_j. For example, this happens when the intersection is edge–to–face or face–to–face. Instead take the dot product of equation (3) with $\vec{B}_j$ to obtain

$$y_j = -\vec{B}_j \cdot \vec{D} + \sum_{k=0}^{2} x_k c_{kj}.$$

Using $|x_k| \le a_k$, we have

$$\min(y_j) = -\vec{B}_j \cdot \vec{D} - \sum_{k=0}^{2} |c_{kj}| a_k \le y_j \le -\vec{B}_j \cdot \vec{D} + \sum_{k=0}^{2} |c_{kj}| a_k = \max(y_j).$$

Additionally we know $|y_j| \le b_j$. We must choose a value $y_j \in [\min(y_j), \max(y_j)] \cap [-b_j, b_j]$. Since we know an intersection must occur, $\min(y_j) \le b_j$ and $-b_j \le \max(y_j)$. If $b_j \le \max(y_j)$, then $y_j = b_j$ is an intersection point. If $-b_j \ge \min(y_j)$, then $y_j = -b_j$ is an intersection point. Otherwise, we may choose $y_j = \min(y_j)$ as an intersection point.

Last Separating Axis $\vec{B}_i$. If the separating axis at time T is $\vec{B}_i$, then the intersection must occur on one of the two faces perpendicular to $\vec{B}_i$. Dotting equation (3) with $\vec{B}_i$ yields

$$\sum_{j=0}^{2} c_{ji} x_j = \vec{B}_i \cdot \vec{D} + y_i.$$

As in the last section it can be shown that $\vec{B}_i \cdot \vec{D} = \sigma(R_0 + R_1)$ where $|\sigma| = 1$. Moreover,

$$\begin{aligned} \textstyle\sum_{j=0}^{2} c_{ji} x_j &= \sigma(R_0 + R_1) + y_i \\ &= \sigma(\textstyle\sum_{j=0}^{2} a_j |c_{ji}| + b_i) + y_i \\ 0 &= \textstyle\sum_{j=0}^{2} |c_{ji}|(a_j - \sigma \mathrm{Sign}(c_{ji}) x_j) + (b_0 + \sigma y_0). \end{aligned} \tag{5}$$

Since $|x_j| \le a_j$ and $|\sigma \mathrm{Sign}(c_{ji})| \le 1$, it must be that $a_j - \sigma \mathrm{Sign}(c_{ji}) x_j \ge 0$. Similarly, $b_i + \sigma y_i \ge 0$ in which case $y_i = -\sigma b_i$. If $c_{ji} \ne 0$, then $x_j = \sigma \mathrm{Sign}(c_{ji}) a_j$ is required to make the sum in equation (5) zero.

If any $c_{ji} = 0$, then the sum in the displayed equation places no restriction on x_i. For example, this happens when the intersection is edge–to–face or face–to–face. Instead take the dot product of equation (3) with $\vec{A}_j$ to obtain

$$x_j = \vec{A}_j \cdot \vec{D} + \sum_{k=0}^{2} y_k c_{jk}.$$

Using $|y_k| \le b_k$, we have

$$\min(x_j) = \vec{A}_j \cdot \vec{D} - \sum_{k=0}^{2} |c_{jk}| b_k \le x_j \le \vec{A}_j \cdot \vec{D} + \sum_{k=0}^{2} |c_{jk}| b_k = \max(x_j).$$

Additionally we know $|x_j| \le a_j$. We must choose a value $x_j \in [\min(x_j), \max(x_j)] \cap [-a_j, a_j]$. Since we know an intersection must occur, $\min(x_j) \le a_j$ and $-a_j \le \max(x_j)$. If $a_j \le \max(x_j)$, then $x_j = a_j$ is an intersection point. If $-a_j \ge \min(x_j)$, then $x_j = -a_j$ is an intersection point. Otherwise, we may choose $x_j = \min(x_j)$ as an intersection point.

Last Separating Axis $\vec{A}_i \times \vec{B}_j$. Let (i_0, i_1, i_2) and (j_0, j_1, j_2) be permutations of $(0, 1, 2)$ in the set $\{(0,1,2), (1,0,2), (2,1,0)\}$. Dot equation (3) with $\vec{A}_{i_0} \times \vec{B}_{j_0}$ to obtain

$$\begin{aligned}
(\vec{A}_{i_1} \cdot \vec{A}_{i_0} \times \vec{B}_{j_0}) x_{i_1} + (\vec{A}_{i_2} \cdot \vec{A}_{i_0} \times \vec{B}_{j_0}) x_{i_2} &= \vec{A}_{i_0} \times \vec{B}_{j_0} \cdot \vec{D} + (\vec{B}_{j_1} \cdot \vec{A}_{i_0} \times \vec{B}_{j_0}) y_{j_1} + (\vec{B}_{j_2} \cdot \vec{A}_{i_0} \times \vec{B}_{j_0}) y_{j_2} \\
\mathrm{Sign}(i_1, i_0) c_{i_2 j_0} x_{i_1} + \mathrm{Sign}(i_2, i_0) c_{i_1 j_0} x_{i_1} &= \sigma(|c_{i_2 j_0}| a_{i_1} + |c_{i_1 j_0}| a_{i_2} + |c_{i_0 j_2}| b_{j_1} + |c_{i_0 j_1}| b_{j_2}) \\
&\quad + \mathrm{Sign}(j_0, j_1) c_{i_0 j_2} y_{j_1} + \mathrm{Sign}(j_0, j_2) c_{i_0 j_1} y_{j_2}.
\end{aligned}$$

where $|\sigma| = 1$. Grouping terms and factoring yields

$$\begin{aligned}
0 = {} & |c_{i_2 j_0}|(a_{i_1} - \sigma \mathrm{Sign}(i_1, i_0) \mathrm{Sign}(c_{i_2 j_0}) x_{i_1}) + |c_{i_1 j_0}|(a_{i_2} - \sigma \mathrm{Sign}(i_1, i_0) \mathrm{Sign}(c_{i_1 j_0}) x_{i_2}) + \\
& |c_{i_0 j_2}|(b_{j_1} - \sigma \mathrm{Sign}(j_1, j_0) \mathrm{Sign}(c_{i_0 j_2}) y_{j_1}) + |c_{i_0 j_1}|(b_{j_2} - \sigma \mathrm{Sign}(j_2, j_0) \mathrm{Sign}(c_{i_0 j_1}) y_{j_2})
\end{aligned}$$

As in the previous section, the quantities multiplying the $|c_{ij}|$ terms must be zero when the c_{ij} term is not zero.

The first case to consider is $c_{i_2 j_0} \neq 0$, $c_{i_1 j_0} \neq 0$, $c_{i_0 j_2} \neq 0$, and $c_{i_0 j_1} \neq 0$. Then

$$\begin{aligned}
x_{i_1} &= \sigma \mathrm{Sign}(i_1, i_0) \mathrm{Sign}(c_{i_2 j_0}) a_{i_1} \\
x_{i_2} &= \sigma \mathrm{Sign}(i_2, i_0) \mathrm{Sign}(c_{i_1 j_0}) a_{i_2} \\
y_{j_1} &= \sigma \mathrm{Sign}(j_1, j_0) \mathrm{Sign}(c_{i_0 j_2}) b_{j_1} \\
y_{j_2} &= \sigma \mathrm{Sign}(j_2, j_0) \mathrm{Sign}(c_{i_0 j_1}) b_{j_2}.
\end{aligned}$$

To solve for x_{i_0} and y_{j_0}, dot equation (3) with $\vec{A}_{i_0}$ and $\vec{B}_{j_0}$ to obtain

$$\begin{aligned}
x_{i_0} &= \vec{A}_{i_0} \cdot \vec{D} + c_{i_0 j_0} y_{j_0} + c_{i_0 j_1} y_{j_1} + c_{i_0 j_2} y_{j_2}, \\
c_{i_0 j_0} x_{i_0} + c_{i_1 j_0} x_{i_1} + c_{i_2 j_0} x_{i_2} &= \vec{B}_{j_0} \cdot \vec{D} + y_{j_0}.
\end{aligned}$$

Replacing each equation in the other yields

$$\begin{aligned}
x_{i_0} &= \frac{1}{1 - c_{i_0 j_0}^2} \left[\vec{A}_{i_0} \cdot \vec{D} + c_{i_0 j_0} \left(-\vec{B}_{j_0} \cdot \vec{D} + c_{i_1 j_0} x_{i_1} + c_{i_2 j_0} x_{i_2} \right) + c_{i_0 j_1} y_{j_1} + c_{i_0 j_2} y_{j_2} \right] \\
y_{j_0} &= \frac{1}{1 - c_{i_0 j_0}^2} \left[-\vec{B}_{j_0} \cdot \vec{D} + c_{i_0 j_0} \left(\vec{A}_{i_0} \cdot \vec{D} + c_{i_0 j_1} y_{j_1} + c_{i_0 j_2} y_{j_2} \right) + c_{i_1 j_0} x_{i_1} + c_{i_2 j_0} x_{i_2} \right]
\end{aligned}$$

The denominator of the fraction is not zero since $1 - c_{i_0 j_0}^2 = c_{i_1 j_0}^2 + c_{i_2 j_0}^2 \neq 0$ since $c_{i_1 j_0} \neq 0$ and $c_{i_2 j_0} \neq 0$.

Geometrically, these four c_{ij} numbers must be zero since this case represents either (1) an edge–to–edge collision and the intersection point must be unique or (2) an edge–to–edge collision where the edges are perfectly aligned. In the latter case, a face axis should separate the two OBBs. Just in case the face axis separation does not happen due to numerical round–off errors, the code has cases to handle whenever any of the $c_{ij} = 0$. The handlers are similar to what was discussed earlier. The intersection equation is dotted with the appropriate vector to solve explicitly for the to–be–determined variable (a x_i or a y_j term). Inequalitites are obtained for that variable and the minimum and maximum values are used as before to find a point in the intersection of two intervals for that variable.

The coefficients needed to produce the unique points of intersection are summarized in the following table.

$\vec{L}$	coefficients
$\vec{A}_i$	$y_j = -\sigma\text{Sign}(c_{ij})b_j,\ j = 0, 1, 2$
$\vec{B}_j$	$x_i = +\sigma\text{Sign}(c_{ij})a_i,\ i = 0, 1, 2$
$\vec{A}_0 \times \vec{B}_0$	$x_1 = -\sigma\text{Sign}(c_{20})a_1,\ x_2 = +\sigma\text{Sign}(c_{10})a_2,\ y_1 = -\sigma\text{Sign}(c_{02})b_1,\ y_2 = +\sigma\text{Sign}(c_{01})b_2,$ $x_0 = \frac{1}{1-c_{00}^2}\left(\vec{A}_0 \cdot \vec{D} + c_{00}(-\vec{B}_0 \cdot \vec{D} + c_{10}x_1 + c_{20}x_2) + c_{01}y_1 + c_{02}y_2\right)$
$\vec{A}_0 \times \vec{B}_1$	$x_1 = -\sigma\text{Sign}(c_{21})a_1,\ x_2 = +\sigma\text{Sign}(c_{11})a_2,\ y_0 = +\sigma\text{Sign}(c_{02})b_0,\ y_2 = -\sigma\text{Sign}(c_{00})b_2,$ $x_0 = \frac{1}{1-c_{01}^2}\left(\vec{A}_0 \cdot \vec{D} + c_{01}(-\vec{B}_1 \cdot \vec{D} + c_{11}x_1 + c_{21}x_2) + c_{00}y_0 + c_{02}y_2\right)$
$\vec{A}_0 \times \vec{B}_2$	$x_1 = -\sigma\text{Sign}(c_{22})a_1,\ x_2 = +\sigma\text{Sign}(c_{12})a_2,\ y_0 = -\sigma\text{Sign}(c_{01})b_0,\ y_1 = +\sigma\text{Sign}(c_{00})b_1,$ $x_0 = \frac{1}{1-c_{02}^2}\left(\vec{A}_0 \cdot \vec{D} + c_{02}(-\vec{B}_2 \cdot \vec{D} + c_{12}x_1 + c_{22}x_2) + c_{00}y_0 + c_{01}y_1\right)$
$\vec{A}_1 \times \vec{B}_0$	$x_0 = +\sigma\text{Sign}(c_{20})a_0,\ x_2 = -\sigma\text{Sign}(c_{00})a_2,\ y_1 = -\sigma\text{Sign}(c_{12})b_1,\ y_2 = +\sigma\text{Sign}(c_{11})b_2,$ $x_1 = \frac{1}{1-c_{10}^2}\left(\vec{A}_1 \cdot \vec{D} + c_{10}(-\vec{B}_0 \cdot \vec{D} + c_{00}x_0 + c_{20}x_2) + c_{11}y_1 + c_{12}y_2\right)$
$\vec{A}_1 \times \vec{B}_1$	$x_0 = +\sigma\text{Sign}(c_{21})a_0,\ x_2 = -\sigma\text{Sign}(c_{01})a_2,\ y_0 = +\sigma\text{Sign}(c_{12})b_0,\ y_2 = -\sigma\text{Sign}(c_{10})b_2,$ $x_1 = \frac{1}{1-c_{11}^2}\left(\vec{A}_1 \cdot \vec{D} + c_{11}(-\vec{B}_1 \cdot \vec{D} + c_{01}x_0 + c_{21}x_2) + c_{10}y_0 + c_{12}y_2\right)$
$\vec{A}_1 \times \vec{B}_2$	$x_0 = +\sigma\text{Sign}(c_{22})a_0,\ x_2 = -\sigma\text{Sign}(c_{02})a_2,\ y_0 = -\sigma\text{Sign}(c_{11})b_0,\ y_1 = +\sigma\text{Sign}(c_{10})b_1,$ $x_1 = \frac{1}{1-c_{12}^2}\left(\vec{A}_1 \cdot \vec{D} + c_{12}(-\vec{B}_2 \cdot \vec{D} + c_{02}x_0 + c_{22}x_2) + c_{10}y_0 + c_{11}y_1\right)$
$\vec{A}_2 \times \vec{B}_0$	$x_0 = -\sigma\text{Sign}(c_{10})a_0,\ x_1 = +\sigma\text{Sign}(c_{00})a_1,\ y_1 = -\sigma\text{Sign}(c_{22})b_1,\ y_2 = +\sigma\text{Sign}(c_{21})b_2,$ $x_2 = \frac{1}{1-c_{20}^2}\left(\vec{A}_2 \cdot \vec{D} + c_{20}(-\vec{B}_0 \cdot \vec{D} + c_{00}x_0 + c_{10}x_1) + c_{21}y_1 + c_{22}y_2\right)$
$\vec{A}_2 \times \vec{B}_1$	$x_0 = -\sigma\text{Sign}(c_{11})a_0,\ x_1 = +\sigma\text{Sign}(c_{01})a_1,\ y_0 = +\sigma\text{Sign}(c_{22})b_0,\ y_2 = -\sigma\text{Sign}(c_{20})b_2,$ $x_2 = \frac{1}{1-c_{21}^2}\left(\vec{A}_2 \cdot \vec{D} + c_{21}(-\vec{B}_1 \cdot \vec{D} + c_{01}x_0 + c_{11}x_1) + c_{20}y_0 + c_{22}y_2\right)$
$\vec{A}_2 \times \vec{B}_2$	$x_0 = -\sigma\text{Sign}(c_{12})a_0,\ x_1 = +\sigma\text{Sign}(c_{02})a_1,\ y_0 = -\sigma\text{Sign}(c_{21})b_0,\ y_1 = +\sigma\text{Sign}(c_{20})b_1,$ $x_2 = \frac{1}{1-c_{22}^2}\left(\vec{A}_2 \cdot \vec{D} + c_{22}(-\vec{B}_2 \cdot \vec{D} + c_{02}x_0 + c_{12}x_1) + c_{20}y_0 + c_{21}y_1\right)$

Table 2. Coefficients for unique points of OBB–OBB intersection.

3 Triangle and Oriented Bounding Box

Triangles are represented in this framework as a collection of three vertices $\vec{U}_i$, $i = 0, 1, 2$. The edges of the triangle are $\vec{E}_0 = \vec{U}_1 - \vec{U}_0$, $\vec{E}_1 = \vec{U}_2 - \vec{U}_0$, and $\vec{E}_2 = \vec{E}_1 - \vec{E}_0$. A normal for the triangle is $\vec{N} = \vec{E}_0 \times \vec{E}_1$ and is not necessarily unit length. The triangle and its interior are given by

$$\left\{\vec{U}_0 + s\vec{E}_0 + t\vec{E}_1 : 0 \le s \le 1,\ 0 \le t \le 1,\ \ s + t \le 1\right\}.$$

3.1 Separation of Triangle and OBB

Let the OBB have center $\vec{C}$, axes $\vec{A}_0$, $\vec{A}_1$, $\vec{A}_2$, and extents a_0, a_1, a_2. Let the triangle have vertices $\vec{U}_0$, $\vec{U}_1$, $\vec{U}_2$; edges $\vec{E}_0 = \vec{U}_1 - \vec{U}_0$, $\vec{E}_1 = \vec{U}_2 - \vec{U}_0$, $\vec{E}_2 = \vec{E}_1 - \vec{E}_0$; and normal $\vec{N} = \vec{E}_0 \times \vec{E}_1$. Define $\vec{D} = \vec{U}_0 - \vec{C}$. The potential separating axes are of the form $\vec{C} + s\vec{L}$ where $\vec{L}$ is one of $\vec{N}$, $\vec{A}_i$, or $\vec{A}_i \times \vec{E}_j$ for $i = 0, 1, 2$ and $j = 0, 1, 2$.

For the OBB, the projection distances of the vertices relative to the line origin $\vec{C}$ are

$$\text{ProjDist}\left(\vec{C} + \sum_{i=0}^{2} \sigma_i a_i \vec{A}_i\right) = \sum_{i=0}^{2} \sigma_i a_i \frac{\vec{L} \cdot \vec{A}_i}{\vec{L} \cdot \vec{L}}$$

and the minimum length interval containing all eight projection distances has center 0 and radius

$$r = \frac{\sum_{i=0}^{2} a_i \text{Sign}(\vec{L} \cdot \vec{A}_i)\vec{L} \cdot \vec{A}_i}{\vec{L} \cdot \vec{L}}.$$

The projection distances of the triangle's vertices relative to the line origin are

$$\text{ProjDist}(\vec{U}_k - \vec{C}) = \frac{\vec{L} \cdot (\vec{U}_k - \vec{C})}{\vec{L} \cdot \vec{L}}$$

for $k = 0, 1, 2$. The projection of the triangle does not have a natural center or radius as does the OBB. Nonintersection now amounts to showing that the minimal interval containing the three projected triangle vertices is separated from the projected OBB interval.

Define R to be r without the division by squared length of $\vec{L}$,

$$R = \sum_{i=0}^{2} a_i \text{Sign}(\vec{L} \cdot \vec{A}_i)\vec{L} \cdot \vec{A}_i. \tag{6}$$

The projection distances without the division by squared length of $\vec{L}$ are

$$\begin{aligned} p_0 &= \vec{L} \cdot (\vec{U}_0 - \vec{C}) = \vec{L} \cdot \vec{D} \\ p_1 &= \vec{L} \cdot (\vec{U}_1 - \vec{C}) = \vec{L} \cdot (\vec{D} + \vec{E}_0) = p_0 + \vec{L} \cdot \vec{E}_0 \\ p_2 &= \vec{L} \cdot (\vec{U}_2 - \vec{C}) = \vec{L} \cdot (\vec{D} + \vec{E}_1) = p_0 + \vec{L} \cdot \vec{E}_1. \end{aligned} \tag{7}$$

Equations (6) and (7) lead to the entries in the table below.

$\vec{L}$	p_0	p_1	p_2	R
$\vec{N}$	$\vec{N}\cdot\vec{D}$	p_0	p_0	$a_0\lvert\vec{N}\cdot\vec{A}_0\rvert + a_1\lvert\vec{N}\cdot\vec{A}_1\rvert + a_2\lvert\vec{N}\cdot\vec{A}_2\rvert$
$\vec{A}_0$	$\vec{A}_0\cdot\vec{D}$	$p_0 + \vec{A}_0\cdot\vec{E}_0$	$p_0 + \vec{A}_0\cdot\vec{E}_1$	a_0
$\vec{A}_1$	$\vec{A}_1\cdot\vec{D}$	$p_0 + \vec{A}_1\cdot\vec{E}_0$	$p_0 + \vec{A}_1\cdot\vec{E}_1$	a_1
$\vec{A}_2$	$\vec{A}_2\cdot\vec{D}$	$p_0 + \vec{A}_2\cdot\vec{E}_0$	$p_0 + \vec{A}_2\cdot\vec{E}_1$	a_2
$\vec{A}_0\times\vec{E}_0$	$\vec{A}_0\times\vec{E}_0\cdot\vec{D}$	p_0	$p_0 + \vec{A}_0\cdot\vec{N}$	$a_1\lvert\vec{A}_2\cdot\vec{E}_0\rvert + a_2\lvert\vec{A}_1\cdot\vec{E}_0\rvert$
$\vec{A}_0\times\vec{E}_1$	$\vec{A}_0\times\vec{E}_1\cdot\vec{D}$	$p_0 - \vec{A}_0\cdot\vec{N}$	p_0	$a_1\lvert\vec{A}_2\cdot\vec{E}_1\rvert + a_2\lvert\vec{A}_1\cdot\vec{E}_1\rvert$
$\vec{A}_0\times\vec{E}_2$	$\vec{A}_0\times\vec{E}_2\cdot\vec{D}$	$p_0 - \vec{A}_0\cdot\vec{N}$	$p_0 - \vec{A}_0\cdot\vec{N}$	$a_1\lvert\vec{A}_2\cdot\vec{E}_2\rvert + a_2\lvert\vec{A}_1\cdot\vec{E}_2\rvert$
$\vec{A}_1\times\vec{E}_0$	$\vec{A}_1\times\vec{E}_0\cdot\vec{D}$	p_0	$p_0 + \vec{A}_1\cdot\vec{N}$	$a_0\lvert\vec{A}_2\cdot\vec{E}_0\rvert + a_2\lvert\vec{A}_0\cdot\vec{E}_0\rvert$
$\vec{A}_1\times\vec{E}_1$	$\vec{A}_1\times\vec{E}_1\cdot\vec{D}$	$p_0 - \vec{A}_1\cdot\vec{N}$	p_0	$a_0\lvert\vec{A}_2\cdot\vec{E}_1\rvert + a_2\lvert\vec{A}_0\cdot\vec{E}_1\rvert$
$\vec{A}_1\times\vec{E}_2$	$\vec{A}_1\times\vec{E}_2\cdot\vec{D}$	$p_0 - \vec{A}_1\cdot\vec{N}$	$p_0 - \vec{A}_1\cdot\vec{N}$	$a_0\lvert\vec{A}_2\cdot\vec{E}_2\rvert + a_2\lvert\vec{A}_0\cdot\vec{E}_2\rvert$
$\vec{A}_2\times\vec{E}_0$	$\vec{A}_2\times\vec{E}_0\cdot\vec{D}$	p_0	$p_0 + \vec{A}_2\cdot\vec{N}$	$a_0\lvert\vec{A}_1\cdot\vec{E}_0\rvert + a_1\lvert\vec{A}_0\cdot\vec{E}_0\rvert$
$\vec{A}_2\times\vec{E}_1$	$\vec{A}_2\times\vec{E}_1\cdot\vec{D}$	$p_0 - \vec{A}_2\cdot\vec{N}$	p_0	$a_0\lvert\vec{A}_1\cdot\vec{E}_1\rvert + a_1\lvert\vec{A}_0\cdot\vec{E}_1\rvert$
$\vec{A}_2\times\vec{E}_2$	$\vec{A}_2\times\vec{E}_2\cdot\vec{D}$	$p_0 - \vec{A}_2\cdot\vec{N}$	$p_0 - \vec{A}_2\cdot\vec{N}$	$a_0\lvert\vec{A}_1\cdot\vec{E}_2\rvert + a_1\lvert\vec{A}_0\cdot\vec{E}_2\rvert$

Table 3. Values for R, p_0, p_1, and p_2 for the nonintersection test $\min(p_0, p_1, p_2) > R$ or $\max(p_0, p_1, p_2) < -R$ for triangle and OBB.

For axis direction $\vec{N}$, the projected triangle vertices are identical, so the nonintersection test amounts to showing $\vec{N}\cdot\vec{D}$ is not in the interval $[-R, R]$. For axis directions $\vec{A}_i$, the projected triangle vertices may all be distinct. For axis directions $\vec{A}_i \times \vec{E}_j$, at most two of the projected vertices are distinct. If the triangle interval is $[\min(p_0, p_1, p_2), \max(p_0, p_1, p_2)]$ and the OBB interval is $[-R, R]$, then the triangle and OBB do not intersect whenever $\min(p_0, p_1, p_2) > R$ or $\max(p_0, p_1, p_2) < -R$.

3.2 Testing for Intersection of Triangle and OBB

3.2.1 Stationary

Testing for intersection amounts to processing each axis of the 13 potential separating axes. If a separating axis is found, the remaining ones are not processed. Any quantities that are needed multiple times are calculated only once and only when needed.

Axis $\vec{N}$. The nonintersection test is $|\vec{N}\cdot\vec{D}| > R$. Pseudocode for testing if p is not in $[-R, R]$ is

```
if ( |p| > R )
    return no_intersection;
```

Axes $\vec{A}_k$. The p–values are p, $p+d_0$, and $p+d_1$. The nonintersection test is: $\min(p, p+d_0, p+d_1) > R$ or $\max(p, p+d_0, p+d_1) < -R$. Pseudocode is

```
if ( p > R )
{
    if ( d0 >= 0 )
    {
        if ( d1 >= 0 )
            return no_intersection;  // pmin = p > R
        if ( p+d1 > R )
            return no_intersection;  // pmin = p+d1 > R
    }
    else if ( d1 <= d0 )
    {
        if ( p+d1 > R )
            return no_intersection;  // pmin = p+d1 > R
    }
    else
    {
        if ( p+d0 > R )
            return no_intersection;  // pmin = p+d0 > R
    }
}
else if ( p < -R )
{
    if ( d0 <= 0 )
    {
        if ( d1 <= 0 )
            return no_intersection;  // pmax = p < -R
        if ( p+d1 < -R )
            return no_intersection;  // pmax = p+d1 < -R
    }
    else if ( d1 >= d0 )
    {
        if ( p+d1 < -R )
            return no_intersection;  // pmax = p+d1 < -R
    }
    else
    {
        if ( p+d0 < -R )
            return no_intersection;  // pmax = p+d0 < -R
    }
}
```

Axes $\vec{A}_i \times \vec{E}_j$. There are at most two distinct values for the p_i, call them p and $p+d$. The nonintersection test is: $\min(p, p+d) > R$ or $\max(p+d) < -R$). Pseudocode is

```
if ( p > R )
```

```
{
    if ( d >= 0 )
        return no_intersection;  // pmin = p > R
    if ( p+d > R )
        return no_intersection;  // pmin = p+d > R
}
else if ( p < -R )
{
    if ( d <= 0 )
        return no_intersection;  // pmax = p < -R
    if ( p+d < -R )
        return no_intersection;  // pmax = p+d < -R
}
```

3.2.2 Constant Velocities

The code for stationary triangle and OBB needs to be modified slightly to handle the case of constant velocities. The velocity of the OBB is subtracted from the velocity of the triangle so that all calculations are done relative to a stationary box. If the box velocity is $\vec{V}_0$ and the triangle velocity is $\vec{V}_1$, define the relative velocity to be $\vec{W} = \vec{V}_1 - \vec{V}_0$. Let the initial time and final time of the calculations be $t = 0$ and $t = T$. Let $\vec{D}_0 = \vec{U}_0 - \vec{C}$ and $\vec{D}_1 = \vec{D}_0 + T\vec{W}$.

The projected OBB interval $[-R, R]$ is stationary. The projected triangle interval is dependent on time, $[\min(p)(t), \max(p)(t)]$. The test for nonintersection during the time interval $[0, T]$ is $\min(p)(t) > R$ for all $t \in [0, T]$ or $\max(p)(t) < -R$ for all $t \in [0, T]$. Because the linear velocity is constant, it is enough to show nonintersection by verifying that $\min\{\min(p)(0), \min(p)(T)\} > R$ or $\max\{\max(p)(0), \max(p)(T)\} < -R$.

Axis $\vec{N}$. The nonintersection test amounts to showing $p + tw$ is not in $[-R, R]$ for $t \in [0, T]$ ($p = \vec{N} \cdot \vec{D}$ and $w = \vec{N} \cdot \vec{W}$). Moreover, the point $p + tw$ must not pass through $[-R, R]$ during the given time period, so the algorithm keeps track of which side of $[-R, R]$ the point starts on. Pseudocode is

```
if ( p > R )
{
    if ( p+T*w > R )
        return no_intersection;
}
else if ( p < -R )
{
    if ( p+T*w < -R )
        return no_intersection;
}
```

Axes $\vec{A}_k$. The problem is to make sure that the minimum interval containing $\{p+Tw, p+d_0+Tw, p+d_1+Tw\}$ does not intersect $[-R, R]$. When $w = 0$ (the static case), the pseudocode for nonintersection was given earlier. For the case when $w \neq 0$, each case where a "no intersection" is returned when $t = 0$ must be refined to make sure the projected triangle interval remains on the same side of $[-R, R]$ at $t = T$. The pseudocode is

```
if ( p > R )
{
    if ( d0 >= 0 )
    {
        if ( d1 >= 0 )
        {
            min = p;
            if ( min+T*w > R )
                return no_intersection;
        }
        else
        {
            min = p+d1;
            if ( min > R && min+T*w > R )
                return no_intersection;
        }
    }
    else if ( d1 <= d0 )
    {
        min = p+d1;
        if ( min > R && min+T*w > R )
            return no_intersection;
    }
    else
    {
        min = p+d0;
        if ( min > R && min+T*w > R )
            return no_intersection;
        }
    }
}
else if ( p < -R )
{
    if ( d0 <= 0 )
    {
        if ( d1 <= 0 )
        {
            max = p;
            if ( max+T*w < -R )
                return no_intersection;
        }
        else
        {
            max = p+d1;
            if ( max < -R && max+T*w < -R )
                return no_intersection;
        }
    }
```

```
    else if ( d1 >= d0 )
    {
        max = p+d1;
        if ( max < -R && max+T*w < -R )
            return no_intersection;
    }
    else
    {
        max = p+d0;
        if ( max < -R && max+T*w < -R )
            return no_intersection;
    }
}
```

Axes $\vec{A}_i \times \vec{E}_j$. The problem is to make sure that the minimum interval containing $\{p + Tw, p + d + Tw\}$ does not intersect $[-R, R]$. When $w = 0$ (the static case), the pseudocode for nonintersection was given earlier. For the case when $w \neq 0$, each case where a "no intersection" is returned when $t = 0$ must be refined to make sure the projected triangle interval remains on the same side of $[-R, R]$ at $t = T$. The pseudocode is

```
if ( p > R )
{
    if ( d >= 0 )
    {
        min = p;
        if ( min+T*w > R )
            return no_intersection;
    }
    else
    {
        min = p+d;
        if ( min > R && min+T*w > R )
            return no_intersection;
    }
}
else if ( p < -R )
{
    if ( d <= 0 )
    {
        max = p;
        if ( max+T*w < -R )
            return no_intersection;
    }
    else
    {
        max = p+d;
        if ( max < -R && max+T*w < -R )
            return no_intersection;
```

```
    }
}
```

3.3 Finding an Intersection of Triangle and OBB

Testing for intersection is a boolean operation. The algorithm returns true if there is an intersection, false if not. No information is provided about where an intersection occurs (there may be many such points).

In the case of two stationary objects that are intersecting, the region of intersection can be computed with great pain. For a dynamic system, the more interesting case is to have two moving objects that are initially not intersecting, but then do intersect during the specified time interval. Of interest to an application is the first time of intersection and a point (or points) of intersection at that time. The following construction provides a point of intersection. In the cases of vertex–to–vertex, vertex–to–edge, vertex–to–face, or edge–to–edge (transversely), the intersection point (at first time) is unique. In the other cases of edge–to–face or face–to–face, the intersection is not unique, but the construction provides one of the intersection points.

3.3.1 Finding the First Time of Intersection

Given that the two objects do not intersect at time $t = 0$, but do intersect at some later time, a simple modification of the algorithm for testing for an intersection provides the first time of intersection. The first time is computed as the *maximum* time $T > 0$ for which there is at least one separating axis for any $t \in [0, T)$, but for which no separating axis exists at time T. The idea is to test each potential separating axis and keep track of the time at which the intervals of projection intersect for the first time. The largest such time is the first time at which the triangle and OBB intersect. Also, it is important to keep track of which side of $[-R, R]$ the other interval intersects. Finally, knowing the separating axis associated with the maximum time T allows us to reconstruct a point of intersection.

3.3.2 Finding a Point of Intersection

If T is the first time of intersection, the problem is now to find a point P in the intersection of the triangle and OBB at that time. We need to solve

$$\sum_{i=0}^{2} x_i \vec{A}_i = \vec{D} + y_0 \vec{E}_0 + y_1 \vec{E}_1 \tag{8}$$

where $\vec{D} = (\vec{C} + T\vec{V}_1) - (\vec{U}_0 + T\vec{V}_0)$, and for x_i with $|x_i| \le a_i$, $i = 0, 1, 2$, and for y_j with $0 \le y_0 \le 1$, $0 \le y_1 \le 1$, and $y_0 + y_1 \le 1$.

Equation (8) can be solved for each variable individually. The solutions are

$$\begin{array}{rcl} x_i & = & \vec{A}_i \cdot \vec{D} + (\vec{A}_i \cdot \vec{E}_0) y_0 + (\vec{A}_i \cdot \vec{E}_1) y_1 \\ y_j & = & \frac{1-2j}{|\vec{N}|^2} \left(-\vec{N} \cdot \vec{D} \times \vec{E}_{1-j} + \sum_{i=0}^{2} x_i \vec{N} \cdot \vec{A}_i \times \vec{E}_{1-j} \right) \end{array} \tag{9}$$

for $i = 0, 1, 2$ and $j = 0, 1$. The equations define the left–hand side as a linear function of the variables in the right–hand side. The x extreme values occur at the vertices of the triangular domain of the function:

$(0,0)$, $(1,0)$, and $(0,1)$. The extreme values of the equations define intervals bounding each variable, $x_i \in [\min(x_i), \max(x_i)]$. The interval end points are

$$\begin{aligned} \min(x_i) &= \vec{A}_i \cdot \vec{D} + \min(0, \vec{A}_i \cdot \vec{E}_0, \vec{A}_i \cdot \vec{E}_1), \\ \max(x_i) &= \vec{A}_i \cdot \vec{D} + \max(0, \vec{A}_i \cdot \vec{E}_0, \vec{A}_i \cdot \vec{E}_1). \end{aligned} \tag{10}$$

The y extreme values occur at the vertices of the rectoidal domain of the function (all $|x_i| = a_i$). The extreme values of the equations define intervals bounding each variable, $y_j \in [\min(y_j), \max(y_j)]$. The interval end points are

$$\begin{aligned} \min(y_j) &= \frac{(2j-1)\vec{N}\cdot\vec{D}\times\vec{E}_{1-j} - \sum_{i=0}^{2} a_i |\vec{N}\cdot\vec{A}_i\times\vec{E}_{1-j}|}{|\vec{N}|^2}, \\ \max(y_j) &= \frac{(2j-1)\vec{N}\cdot\vec{D}\times\vec{E}_{1-j} + \sum_{i=0}^{2} a_i |\vec{N}\cdot\vec{A}_i\times\vec{E}_{1-j}|}{|\vec{N}|^2}. \end{aligned} \tag{11}$$

In the following constructions of the first point of intersection, if any of the variables is not uniquely constrained by the derived equations, then such a variable can be selected from the intervals $[\min(x_i), \max(x_i)]$ or $[\min(y_j), \max(y_j)]$ and subject to the other domain constraints for that variable.

The following are a few useful functions in the constructions. Define α, β, γ, and δ by

$$\alpha(k) = \left\{ \begin{array}{rl} 0, & k = 0 \\ 1, & k = 1 \\ 1, & k = 2 \end{array} \right\}, \quad \beta(k) = \left\{ \begin{array}{rl} 1, & k = 0 \\ 0, & k = 1 \\ 1, & k = 2 \end{array} \right\},$$

$$\gamma(k) = \left\{ \begin{array}{rl} -1, & k = 0 \\ 0, & k = 1 \\ 1, & k = 2 \end{array} \right\}, \quad \delta(k) = \left\{ \begin{array}{rl} -1, & k = 0 \\ 1, & k = 1 \\ 1, & k = 2 \end{array} \right\}.$$

Some useful identities are $\delta \equiv 2\alpha - 1$, $\delta^2 \equiv 1$, $\alpha\delta \equiv \alpha$, and $\gamma\delta \equiv \beta$.

Last Separating Axis $\vec{N}$. If the separating axis at time T is $\vec{N}$, then the intersection must occur on the triangle itself. Dotting equation (8) with $\vec{N}$ yields

$$\sum_{i=0}^{2} x_i \vec{N} \cdot \vec{A}_i = \vec{N} \cdot \vec{D}.$$

Note that $\vec{N} \cdot \vec{D} = \sigma R$ for $|\sigma| = 1$. Thus,

$$\begin{aligned} \textstyle\sum_{i=0}^{2} x_i \vec{N} \cdot \vec{A}_i &= \sigma R \\ \textstyle\sum_{i=0}^{2} x_i \vec{N} \cdot \vec{A}_i &= \sigma \textstyle\sum_{i=0}^{2} a_i |\vec{N} \cdot \vec{A}_i| \\ 0 &= \textstyle\sum_{i=0}^{2} |\vec{N} \cdot \vec{A}_i| (a_i - \sigma \mathrm{Sign}(\vec{N} \cdot \vec{A}_i) x_i). \end{aligned} \tag{12}$$

Since $|x_i| \le a_i$ and $|\sigma \mathrm{Sign}(\vec{N} \cdot \vec{A}_i)| \le 1$, it must be that $a_i - \sigma \mathrm{Sign}(\vec{N} \cdot \vec{A}_i) x_i \ge 0$. If $\vec{N} \cdot \vec{A}_i \neq 0$, then $x_i = \sigma \mathrm{Sign}(\vec{N} \cdot \vec{A}_i) a_i$ is required to make the sum in equation (12) zero. If any $\vec{N} \cdot \vec{A}_i = 0$, then the sum in equation (12) places no restriction on x_i. For example, this happens when the intersection is edge–to–triangle or face–to–triangle. Any $x_i \in [\min(x_i), \max(x_i)) \cap [-a_i, a_i]$ can be selected for a point of intersection.

Last Separating Axis $\vec{A}_i$. If the separating axis at time T is $\vec{A}_i$, then the intersection must occur on one of the two OBB faces perpendicular to $\vec{A}_i$ and $R = a_i$. The formula for x_i from equation (9) has three cases to be considered.

The first case is $p_0 = \min_j(p_j) = a_i$ in which case $\sigma = 1$ or $p_0 = \max_j(p_j) = -a_i$ in which case $\sigma = -1$. Then $\sigma\vec{A}_i \cdot \vec{E}_0 \geq 0$, $\sigma\vec{A}_i \cdot \vec{E}_1 \geq 0$, and $x_i = \sigma a_i + y_0\vec{A}_i \cdot \vec{E}_0 + y_1\vec{A}_i \cdot \vec{E}_1$. The intersection occurs on a face of the OBB perpendicular to $\vec{A}_i$, so it must be that $x_i = \sigma a_i$ and

$$0 = (\sigma\vec{A}_i \cdot \vec{E}_0)y_0 + (\sigma\vec{A}_i \cdot \vec{E}_1)y_1. \tag{13}$$

If $\vec{A}_i \cdot \vec{E}_0 \neq 0$ and $\vec{A}_i \cdot \vec{E}_1 \neq 0$, then $y_0 = 0$ and $y_1 = 0$ are required. If $\vec{A}_i \cdot \vec{E}_0 = 0$ and $\vec{A}_i \cdot \vec{E}_1 \neq 0$, equation (13) requires $y_1 = 0$ but does not constrain y_0. In this case $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (11). If $\vec{A}_i \cdot \vec{E}_0 \neq 0$ and $\vec{A}_i \cdot \vec{E}_1 = 0$, equation (13) requires $y_0 = 0$ but does not constrain y_1. In this case $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (11). If $\vec{A}_i \cdot \vec{E}_0 = 0$ and $\vec{A}_i \cdot \vec{E}_1 = 0$, equation (13) constrains neither y_0 nor y_1. The OBB intersects the triangle face–to–face, a case handled by the separating axis test for $\vec{N}$.

The second case is $p_1 = \min_j(p_j) = a_i$ in which case $\sigma = 1$ or $p_1 = \max_j(p_j) = -a_i$ in which case $\sigma = -1$. Then $-\sigma\vec{A}_i \cdot \vec{E}_0 \geq 0$, $\sigma(\vec{A}_i \cdot \vec{E}_1 - \vec{A}_i \cdot \vec{E}_0) \geq 0$, and $x_i = \sigma a_i - \vec{A}_i \cdot \vec{E}_0 + y_0\vec{A}_i \cdot \vec{E}_0 + y_1\vec{A}_i \cdot \vec{E}_1$. Once again, $x_i = \sigma a_i$ and

$$0 = (-\sigma\vec{A}_i \cdot \vec{E}_0)(1 - y_0 - y_1) + [\sigma(\vec{A}_i \cdot \vec{E}_1 - \vec{A}_i \cdot \vec{E}_0)]y_1. \tag{14}$$

If $\vec{A}_i \cdot \vec{E}_0 \neq 0$ and $\vec{A}_i \cdot \vec{E}_1 \neq \vec{A}_i \cdot \vec{E}_0$, then $1 - y_0 - y_1 = 0$ and $y_1 = 0$ are required. Therefore, $y_0 = 1$ and $y_1 = 0$. If $\vec{A}_i \cdot \vec{E}_0 = 0$ and $\vec{A}_i \cdot \vec{E}_1 \neq \vec{A}_i \cdot \vec{E}_0$, equation (14) requires $y_1 = 0$ but does not constrain y_0. In this case $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (11). If $\vec{A}_i \cdot \vec{E}_0 \neq 0$ and $\vec{A}_i \cdot \vec{E}_1 = \vec{A}_i \cdot \vec{E}_0$, equation (14) requires $y_0 + y_1 = 1$ but does not constrain y_1. In this case $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (11). Given a choice of y_1, then $y_0 = 1 - y_1$. If $\vec{A}_i \cdot \vec{E}_0 = \vec{A}_i \cdot \vec{E}_1 = 0$, then neither y_0 nor y_1 is constrained. In this event, $\vec{N}$ and $\vec{A}_i$ are parallel, a case handled by the separating axis test for $\vec{N}$.

The third case is $p_2 = \min_j(p_j) = a_i$ in which case $\sigma = 1$ or $p_2 = \max_j(p_j) = -a_i$ in which case $\sigma = -1$. Then $-\sigma\vec{A}_i \cdot \vec{E}_1 \geq 0$, $\sigma(\vec{A}_i \cdot \vec{E}_0 - \vec{A}_i \cdot \vec{E}_1) \geq 0$, and $x_i = \sigma a_i - \vec{A}_i \cdot \vec{E}_1 + y_0\vec{A}_i \cdot \vec{E}_0 + y_1\vec{A}_i \cdot \vec{E}_1$. Once again, $x_i = \sigma a_i$ and

$$0 = [\sigma(\vec{A}_i \cdot \vec{E}_0 - \vec{A}_i \cdot \vec{E}_1)]y_0 + (-\sigma\vec{A}_i \cdot \vec{E}_1)(1 - y_0 - y_1). \tag{15}$$

If $\vec{A}_i \cdot \vec{E}_1 \neq 0$ and $\vec{A}_i \cdot \vec{E}_0 \neq \vec{A}_i \cdot \vec{E}_1$, then $1 - y_0 - y_1 = 0$ and $y_0 = 0$ are required. Therefore, $y_0 = 0$ and $y_1 = 1$. If $\vec{A}_i \cdot \vec{E}_1 = 0$ and $\vec{A}_i \cdot \vec{E}_0 \neq \vec{A}_i \cdot \vec{E}_1$, equation (15) requires $y_0 = 0$ but does not constrain y_1. In this case $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (11). Given a choice of y_1, then $y_0 = 1 - y_1$. If $\vec{A}_i \cdot \vec{E}_1 \neq 0$ and $\vec{A}_i \cdot \vec{E}_0 = \vec{A}_i \cdot \vec{E}_1$, equation (15) requires $1 - y_0 - y_1 = 0$ but does not constrain y_0. In this case $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (11). Given a choice of y_0, then $y_1 = 1 - y_0$. If $\vec{A}_i \cdot \vec{E}_1 = \vec{A}_i \cdot \vec{E}_0 = 0$, then neither y_0 nor y_1 is constrained. In this event, $\vec{N}$ and $\vec{A}_i$ are parallel, a case handled by the separating axis test for $\vec{N}$.

Last Separating Axis $\vec{A}_i \times \vec{E}_j$. Let (i_0, i_1, i_2) and (j_0, j_1, j_2) be permutations of $(0, 1, 2)$ in the set $\{(0, 1, 2), (1, 0, 2), (2, 1, 0)\}$. Dot equation (8) with $\vec{A}_{i_0} \times \vec{E}_{j_0}$ to obtain

$$(\vec{A}_{i_1} \cdot \vec{A}_{i_0} \times \vec{E}_{j_0})x_{i_1} + (\vec{A}_{i_2} \cdot \vec{A}_{i_0} \times \vec{E}_{j_0})x_{i_2} = \vec{A}_{i_0} \times \vec{E}_{j_0} \cdot \vec{D} + (\vec{E}_0 \cdot \vec{A}_{i_0} \times \vec{E}_{j_0})y_0 + (\vec{E}_1 \cdot \vec{A}_{i_0} \times \vec{E}_{j_0})y_1.$$

Using various identities, the equation reduces to

$$(\text{Sign}(i_1, i_0)\vec{A}_{i_2} \cdot \vec{E}_{j_0})x_{i_1} + (\text{Sign}(i_2, i_0)\vec{A}_{i_1} \cdot \vec{E}_{j_0})x_{i_2} = p_0 - \alpha(j_0)\vec{N} \cdot \vec{A}_{i_0}y_0 - \gamma(j_0)\vec{N} \cdot \vec{A}_{i_0}y_1. \tag{16}$$

where $p_0 = \vec{A}_{i_0} \times \vec{E}_{j_0} \cdot \vec{D}$ from Table 3. The projection of the triangle's vertices leads to (possibly) two distinct p values, $\{p_0, p_0 - \delta(j_0)\vec{N} \cdot \vec{A}_{i_0}\}$. There are two cases to consider depending on which of the p values are minima or maxima. In all cases,

$$R = a_{i_1}|\vec{A}_{i_2} \cdot \vec{E}_{j_0}| + a_{i_2}|\vec{A}_{i_1} \cdot \vec{E}_{j_0}|.$$

The first case is $p_0 = \min_k(p_k) = R$ in which case $\sigma = 1$ or $p_0 = \max_k(p_k) = -R$ in which case $\sigma = -1$. Then $p_0 = \sigma R$ and $-\sigma\delta(j_0)\vec{A}_{i_0} \cdot \vec{N} \geq 0$. Equation (16) is equivalent to

$$\begin{aligned} 0 &= |\vec{A}_{i_2} \cdot \vec{E}_{j_0}|(a_{i_1} - \sigma\text{Sign}(i_1, i_0)\text{Sign}(\vec{A}_{i_2} \cdot \vec{E}_{j_0})x_{i_1}) + \\ &\quad |\vec{A}_{i_1} \cdot \vec{E}_{j_0}|(a_{i_2} - \sigma\text{Sign}(i_2, i_0)\text{Sign}(\vec{A}_{i_1} \cdot \vec{E}_{j_0})x_{i_2}) + \\ &\quad (-\sigma\delta(j_0)\vec{A}_{i_0} \cdot \vec{N})(\alpha(j_0)y_0 + \beta(j_0)y_1). \end{aligned}$$

If $\vec{A}_{i_2} \cdot \vec{E}_{j_0} \neq 0$, then $x_{i_1} = \sigma\text{Sign}(i_1, i_0)\text{Sign}(\vec{A}_{i_2} \cdot \vec{E}_{j_0})a_{i_1}$. If $\vec{A}_{i_1} \cdot \vec{E}_{j_0} \neq 0$, then $x_{i_2} = \sigma\text{Sign}(i_2, i_0)\text{Sign}(\vec{A}_{i_1} \cdot \vec{E}_{j_0})a_{i_2}$. If $\vec{A}_{i_0} \cdot \vec{N} \neq 0$, then $\alpha(j_0)y_0 + \beta(j_0)y_1 = 0$. These provide three equations in five unknowns. Two additional equations to form an invertible system can be selected from equations (9).

The second case is $p_0 - \delta(j_0)\vec{N} \cdot \vec{A}_{i_0} = \min_k(p_k) = R$ in which case $\sigma = 1$ or $p_0 - \delta(j_0)\vec{N} \cdot \vec{A}_{i_0} = \max_k(p_k) = -R$ in which case $\sigma = -1$. Then $p_0 = \sigma R + \delta(j_0)\vec{N} \cdot \vec{A}_{i_0}$ and $\sigma\delta(j_0)\vec{A}_{i_0} \cdot \vec{N} \geq 0$. Equation (16) is equivalent to

$$\begin{aligned} 0 &= |\vec{A}_{i_2} \cdot \vec{E}_{j_0}|(a_{i_1} - \sigma\text{Sign}(i_1, i_0)\text{Sign}(\vec{A}_{i_2} \cdot \vec{E}_{j_0})x_{i_1}) + \\ &\quad |\vec{A}_{i_1} \cdot \vec{E}_{j_0}|(a_{i_2} - \sigma\text{Sign}(i_2, i_0)\text{Sign}(\vec{A}_{i_1} \cdot \vec{E}_{j_0})x_{i_2}) + \\ &\quad (\sigma\delta(j_0)\vec{A}_{i_0} \cdot \vec{N})(1 - \alpha(j_0)y_0 - \beta(j_0)y_1). \end{aligned}$$

If $\vec{A}_{i_2} \cdot \vec{E}_{j_0} \neq 0$, then $x_{i_1} = \sigma\text{Sign}(i_1, i_0)\text{Sign}(\vec{A}_{i_2} \cdot \vec{E}_{j_0})a_{i_1}$. If $\vec{A}_{i_1} \cdot \vec{E}_{j_0} \neq 0$, then $x_{i_2} = \sigma\text{Sign}(i_2, i_0)\text{Sign}(\vec{A}_{i_1} \cdot \vec{E}_{j_0})a_{i_2}$. If $\vec{A}_{i_0} \cdot \vec{N} \neq 0$, then $\alpha(j_0)y_0 + \beta(j_0)y_1 = 1$. These provide three equations in five unknowns. Two additional equations to form an invertible system can be selected from equations (9).

If any of the x_i or y_j are not constrained because their coefficients are zero, a similar construction can be used as before where intervals are obtained on each of the variables and the intersection of those intervals with their natural restrictions produce a point of intersection.

The coefficients needed to produce the unique points of intersection are summarized in the following tables.

$\vec{L}$	coefficients
$\vec{N}$	$x_i = +\sigma\text{Sign}(\vec{N} \cdot \vec{A}_i)a_i,\ i = 0, 1, 2$
$\vec{A}_i$	$y_0 = 0,\ y_1 = 0, \quad \sigma p_0 = \min_k(\sigma p_k)$
	$y_0 = 1,\ y_1 = 0, \quad \sigma p_1 = \min_k(\sigma p_k)$
	$y_0 = 0,\ y_1 = 1, \quad \sigma p_2 = \min_k(\sigma p_k)$

Table 4a. Coefficients for unique points of OBB–Triangle intersection for $\vec{N}$ and $\vec{A}_i$.

$\vec{L}$	coefficients
$\vec{A}_0 \times \vec{E}_0$	$x_1 = -\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_0)a_1,\ x_2 = +\sigma\mathrm{Sign}(\vec{A}_1 \cdot \vec{E}_0)a_2,$ $y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 + \vec{N} \cdot \vec{A}_0) = \min_k(\sigma p_k) \end{array} \right\}$ $x_0 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_0 - \lvert\vec{N}\rvert^2 y_1 - \vec{N} \cdot \vec{A}_1 \times \vec{E}_0 x_1 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_0 x_2}{\vec{N} \cdot \vec{A}_0 \times \vec{E}_0}$
$\vec{A}_0 \times \vec{E}_1$	$x_1 = -\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_1)a_1,\ x_2 = +\sigma\mathrm{Sign}(\vec{A}_1 \cdot \vec{E}_1)a_2,$ $y_0 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_0) = \min_k(\sigma p_k) \end{array} \right\}$ $x_0 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_1 + \lvert\vec{N}\rvert^2 y_0 - \vec{N} \cdot \vec{A}_1 \times \vec{E}_1 x_1 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_1 x_2}{\vec{N} \cdot \vec{A}_0 \times \vec{E}_1}$
$\vec{A}_0 \times \vec{E}_2$	$x_1 = -\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_2)a_1,\ x_2 = +\sigma\mathrm{Sign}(\vec{A}_1 \cdot \vec{E}_2)a_2,$ $y_0 + y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_0) = \min_k(\sigma p_k) \end{array} \right\}$ $x_0 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_2 + \lvert\vec{N}\rvert^2 (y_0 + y_1) - \vec{N} \cdot \vec{A}_1 \times \vec{E}_2 x_1 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_2 x_2}{\vec{N} \cdot \vec{A}_0 \times \vec{E}_2}$

Table 4b. Coefficients for unique points of OBB–Triangle intersection for $\vec{A}_0 \times \vec{E}_j$.

$\vec{L}$	coefficients
$\vec{A}_1 \times \vec{E}_0$	$x_0 = +\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_0)a_0,\ x_2 = -\sigma\mathrm{Sign}(\vec{A}_0 \cdot \vec{E}_0)a_2,$ $y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(p_k) \\ 1, & \sigma(p_0 + \vec{N} \cdot \vec{A}_1) = \min_k(\sigma p_k) \end{array} \right\}$ $x_1 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_0 - \lvert\vec{N}\rvert^2 y_1 - \vec{N} \cdot \vec{A}_0 \times \vec{E}_0 x_0 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_0 x_2}{\vec{N} \cdot \vec{A}_1 \times \vec{E}_0}$
$\vec{A}_1 \times \vec{E}_1$	$x_0 = +\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_1)a_0,\ x_2 = -\sigma\mathrm{Sign}(\vec{A}_0 \cdot \vec{E}_1)a_2,$ $y_0 = \left\{ \begin{array}{ll} 0, & \sigma(p_0 = \min_k(p_k)) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_1) = \min_k(\sigma p_k) \end{array} \right\}$ $x_1 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_1 + \lvert\vec{N}\rvert^2 y_0 - \vec{N} \cdot \vec{A}_0 \times \vec{E}_1 x_0 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_1 x_2}{\vec{N} \cdot \vec{A}_1 \times \vec{E}_1}$
$\vec{A}_1 \times \vec{E}_2$	$x_0 = +\sigma\mathrm{Sign}(\vec{A}_2 \cdot \vec{E}_2)a_0,\ x_2 = -\sigma\mathrm{Sign}(\vec{A}_0 \cdot \vec{E}_2)a_2,$ $y_0 + y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(p_k) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_1) = \min_k(\sigma p_k) \end{array} \right\}$ $x_1 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_2 + \lvert\vec{N}\rvert^2 (y_0 + y_1) - \vec{N} \cdot \vec{A}_0 \times \vec{E}_2 x_0 - \vec{N} \cdot \vec{A}_2 \times \vec{E}_2 x_2}{\vec{N} \cdot \vec{A}_1 \times \vec{E}_2}$

Table 4c. Coefficients for unique points of OBB–Triangle intersection for $\vec{A}_1 \times \vec{E}_j$.

$\vec{L}$	coefficients
$\vec{A}_2 \times \vec{E}_0$	$x_0 = -\sigma\text{Sign}(\vec{A}_1 \cdot \vec{E}_0)a_0,\ x_1 = +\sigma\text{Sign}(\vec{A}_0 \cdot \vec{E}_0)a_1,$ $y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 + \vec{N} \cdot \vec{A}_2) = \min_k(\sigma p_k) \end{array} \right\}$ $x_2 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_0 - \lvert\vec{N}\rvert^2 y_1 - \vec{N} \cdot \vec{A}_0 \times \vec{E}_0 x_0 - \vec{N} \cdot \vec{A}_1 \times \vec{E}_0 x_1}{\vec{N} \cdot \vec{A}_2 \times \vec{E}_0}$
$\vec{A}_2 \times \vec{E}_1$	$x_0 = -\sigma\text{Sign}(\vec{A}_1 \cdot \vec{E}_1)a_0,\ x_1 = +\sigma\text{Sign}(\vec{A}_0 \cdot \vec{E}_1)a_1,$ $y_0 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_2) = \min_k(\sigma p_k) \end{array} \right\}$ $x_2 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_1 + \lvert\vec{N}\rvert^2 y_0 - \vec{N} \cdot \vec{A}_0 \times \vec{E}_1 x_0 - \vec{N} \cdot \vec{A}_1 \times \vec{E}_1 x_1}{\vec{N} \cdot \vec{A}_2 \times \vec{E}_1}$
$\vec{A}_2 \times \vec{E}_2$	$x_0 = -\sigma\text{Sign}(\vec{A}_1 \cdot \vec{E}_2)a_0,\ x_1 = +\sigma\text{Sign}(\vec{A}_0 \cdot \vec{E}_2)a_1,$ $y_0 + y_1 = \left\{ \begin{array}{ll} 0, & \sigma p_0 = \min_k(\sigma p_k) \\ 1, & \sigma(p_0 - \vec{N} \cdot \vec{A}_2) = \min_k(\sigma p_k) \end{array} \right\}$ $x_2 = \frac{\vec{N} \cdot \vec{D} \times \vec{E}_2 + \lvert\vec{N}\rvert^2 (y_0 + y_1) - \vec{N} \cdot \vec{A}_0 \times \vec{E}_2 x_0 - \vec{N} \cdot \vec{A}_1 \times \vec{E}_2 x_1}{\vec{N} \cdot \vec{A}_2 \times \vec{E}_2}$

Table 4d. Coefficients for unique points of OBB–Triangle intersection for $\vec{A}_2 \times \vec{E}_j$.

4 Triangles

4.1 Separation of Triangles

Let the first triangle have vertices $\vec{A}_0$, $\vec{A}_1$, $\vec{A}_2$, edges $\vec{E}_0 = \vec{A}_1 - \vec{A}_0$, $\vec{E}_1 = \vec{A}_2 - \vec{A}_0$, $\vec{E}_2 = \vec{E}_1 - \vec{E}_0$, and normal $\vec{N} = \vec{E}_0 \times \vec{E}_1$ (not necessarily unit length). Let the second triangle have vertices $\vec{B}_0$, $\vec{B}_1$, $\vec{B}_2$, edges $\vec{F}_0 = \vec{B}_1 - \vec{B}_0$, $\vec{F}_1 = \vec{B}_2 - \vec{B}_0$, $\vec{F}_2 = \vec{F}_1 - \vec{F}_0$, and normal $\vec{M} = \vec{F}_0 \times \vec{F}_1$ (not necessarily unit length). Define $\vec{D} = \vec{B}_0 - \vec{A}_0$.

Triangles in 3D present an interesting problem for nonintersection by the separating axis approach. The set of potential separating axes depends on whether or not the triangles are parallel. If the two triangles are parallel but not coplanar, then the triangle normals will provide separating axes. However, if the triangles are coplanar, then neither normal provides a separating axis. Moreover, cross products of pair of edges from the triangles are all normal vectors, so they do not yield separating axes. It turns out that for the coplanar case, cross products of a triangle normal with the edges of the other triangle provide the potential separating axes.

For nonparallel triangles, the potential separating axes are of the form $\vec{A}_0 + s\vec{L}$ where $\vec{L}$ is one of $\vec{N}$, $\vec{M}$, or $\vec{E}_i \times \vec{F}_j$ for $i = 0, 1, 2$ and $j = 0, 1, 2$. For parallel or coplanar triangles, $\vec{L}$ is one of $\vec{N}$, $\vec{N} \times \vec{E}_i$, or $\vec{N} \times \vec{F}_i$ for $i = 0, 1, 2$.

The projection distances of the triangle vertices without the division by the squared length of $\vec{L}$ are

$$\begin{aligned} p_0 &= \vec{L} \cdot (\vec{A}_0 - \vec{A}_0) = 0 \\ p_1 &= \vec{L} \cdot (\vec{A}_1 - \vec{A}_0) = \vec{L} \cdot \vec{E}_0 \\ p_2 &= \vec{L} \cdot (\vec{A}_2 - \vec{A}_0) = \vec{L} \cdot \vec{E}_1 \end{aligned} \tag{17}$$

and

$$\begin{aligned} q_0 &= \vec{L} \cdot (\vec{B}_0 - \vec{A}_0) = \vec{L} \cdot \vec{D} \\ q_1 &= \vec{L} \cdot (\vec{B}_1 - \vec{A}_0) = \vec{L} \cdot (\vec{D} + \vec{F}_0) = q_0 + \vec{L} \cdot \vec{F}_0 \\ q_2 &= \vec{L} \cdot (\vec{B}_2 - \vec{A}_0) = \vec{L} \cdot (\vec{D} + \vec{F}_1) = q_0 + \vec{L} \cdot \vec{F}_1. \end{aligned} \tag{18}$$

Equations (17) and (18) lead to the entries in the table below. This table represents the potential separating axes for nonparallel triangles.

$\vec{L}$	p_0	p_1	p_2	q_0	q_1	q_2
$\vec{N}$	0	0	0	$\vec{N} \cdot \vec{D}$	$q_0 + \vec{N} \cdot \vec{F}_0$	$q_0 + \vec{N} \cdot \vec{F}_1$
$\vec{M}$	0	$\vec{M} \cdot \vec{E}_0$	$\vec{M} \cdot \vec{E}_1$	$\vec{M} \cdot \vec{D}$	q_0	q_0
$\vec{E}_0 \times \vec{F}_0$	0	0	$-\vec{N} \cdot \vec{F}_0$	$\vec{E}_0 \times \vec{F}_0 \cdot \vec{D}$	q_0	$q_0 + \vec{M} \cdot \vec{E}_0$
$\vec{E}_0 \times \vec{F}_1$	0	0	$-\vec{N} \cdot \vec{F}_1$	$\vec{E}_0 \times \vec{F}_1 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_0$	q_0
$\vec{E}_0 \times \vec{F}_2$	0	0	$-\vec{N} \cdot \vec{F}_2$	$\vec{E}_0 \times \vec{F}_2 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_0$	$q_0 - \vec{M} \cdot \vec{E}_0$
$\vec{E}_1 \times \vec{F}_0$	0	$\vec{N} \cdot \vec{F}_0$	0	$\vec{E}_1 \times \vec{F}_0 \cdot \vec{D}$	q_0	$q_0 + \vec{M} \cdot \vec{E}_1$
$\vec{E}_1 \times \vec{F}_1$	0	$\vec{N} \cdot \vec{F}_1$	0	$\vec{E}_1 \times \vec{F}_1 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_1$	q_0
$\vec{E}_1 \times \vec{F}_2$	0	$\vec{N} \cdot \vec{F}_2$	0	$\vec{E}_1 \times \vec{F}_2 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_1$	$q_0 - \vec{M} \cdot \vec{E}_1$
$\vec{E}_2 \times \vec{F}_0$	0	$\vec{N} \cdot \vec{F}_0$	$\vec{N} \cdot \vec{F}_0$	$\vec{E}_2 \times \vec{F}_0 \cdot \vec{D}$	q_0	$q_0 + \vec{M} \cdot \vec{E}_2$
$\vec{E}_2 \times \vec{F}_1$	0	$\vec{N} \cdot \vec{F}_1$	$\vec{N} \cdot \vec{F}_1$	$\vec{E}_2 \times \vec{F}_1 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_2$	q_0
$\vec{E}_2 \times \vec{F}_2$	0	$\vec{N} \cdot \vec{F}_2$	$\vec{N} \cdot \vec{F}_2$	$\vec{E}_2 \times \vec{F}_2 \cdot \vec{D}$	$q_0 - \vec{M} \cdot \vec{E}_2$	$q_0 - \vec{M} \cdot \vec{E}_2$

Table 5. Values for p_i and q_j for the nonintersection test $\min_i(p_i) > \max_j(q_j)$ or $\max_i(p_i) < \min_j(q_j)$ for noncoplanar triangles.

The table for the potential separating axes for coplanar triangles is given below. The quantites in that table

are $\ell = |\vec{N}|$; $\vec{E}_i \times \vec{F}_j = \lambda_{ij}\vec{N}$ for $i = 0, 1$ and $j = 0, 1$; and $\vec{F}_0 \times \vec{F}_1 = \mu\vec{N}$.

$\vec{L}$	p_0	p_1	p_2	q_0	q_1	q_2
$\vec{N}$	0	0	0	$\vec{N} \cdot \vec{D}$	q_0	q_0
$\vec{N} \times \vec{E}_0$	0	0	ℓ^2	$\vec{N} \times \vec{E}_0 \cdot \vec{D}$	$q_0 + \lambda_{00}\ell^2$	$q_0 + \lambda_{01}\ell^2$
$\vec{N} \times \vec{E}_1$	0	$-\ell^2$	0	$\vec{N} \times \vec{E}_1 \cdot \vec{D}$	$q_0 + \lambda_{10}\ell^2$	$q_0 + \lambda_{11}\ell^2$
$\vec{N} \times \vec{E}_2$	0	$-\ell^2$	$-\ell^2$	$\vec{N} \times \vec{E}_2 \cdot \vec{D}$	$q_0 + (\lambda_{10} - \lambda_{00})\ell^2$	$q_0 + (\lambda_{11} - \lambda_{01})\ell^2$
$\vec{N} \times \vec{F}_0$	0	$-\lambda_{00}\ell^2$	$-\lambda_{10}\ell^2$	$\vec{N} \times \vec{F}_0 \cdot \vec{D}$	q_0	$q_0 + \mu\ell^2$
$\vec{N} \times \vec{F}_1$	0	$-\lambda_{01}\ell^2$	$-\lambda_{11}\ell^2$	$\vec{N} \times \vec{F}_1 \cdot \vec{D}$	$q_0 - \mu\ell^2$	q_0
$\vec{N} \times \vec{F}_2$	0	$(\lambda_{00} - \lambda_{01})\ell^2$	$(\lambda_{10} - \lambda_{11})\ell^2$	$\vec{N} \times \vec{F}_2 \cdot \vec{D}$	$q_0 - \mu\ell^2$	$q_0 - \mu\ell^2$

Table 6. Values for p_i and q_j for the nonintersection test $\min_i(p_i) > \max_j(q_j)$ or $\max_i(p_i) < \min_j(q_j)$ for coplanar triangles.

4.2 Testing for Intersection of Noncoplanar Triangles

I consider only the noncoplanar case. For 3D collision systems, the objects may as well be formed from triangles (rather than polygons of four or more sides). Moreover, the objects tend to have volume, so if two objects are colliding and the contact occurs between two coplanar triangles, one from each object, an instant later other triangles, (1) not in the same plane as the colliding triangles and (2) attached to the colliding triangles, will transversely intersect. The collision system will detect those intersections as occurring between noncoplanar triangles.

4.2.1 Stationary

Testing for intersection amounts to processing the various potential separating axes. Any quantities that are needed multiple times are calculated only once and only when needed. The basic separating axis test involves computing the triangle intervals $[\min_i(p_i), \max_i(p_i)]$ and $[\min_j(q_j), \max_j(q_j)]$ and then testing for nonintersection by comparing the two intervals.

Axes $\vec{N}$ or $\vec{M}$. One triangle projects to a single point p and the other projects to up to three points q, $q + d_0$, and $q + d_1$. The nonintersection test is: $\min(q, q + d_0, q + d_1) > p$ or $\max(q, q + d_0, q + d_1) < p$. Pseudocode is

```
if ( q > p )
{
    if ( q+d0 > p && q+d1 > p )
        return no_intersection;
}
else if ( q < p )
{
```

```
    if ( q+d0 < p && q+d1 < p )
        return no_intersection;
}
```

Axes $\vec{E}_i \times \vec{F}_j$. The triangles each project to exactly two points, the first with values 0 and p, the second with values q and $q+d$. The nonintersection test is: $\min(q, q+d_0, q+d_1) > \max\{0, p\}$ or $\max(q, q+d_0, q+d_1) < \min\{0, p\}$. Pseudocode is

```
if ( p >= 0 )
{
    if ( (q < 0 && q+d < 0) || (q > p && q+d > p) )
        return no_intersection;
}
else
{
    if ( (q > 0 && q+d > 0) || (q < p && q+d < p) )
        return no_intersection;
}
```

4.2.2 Constant Velocities

The code for stationary triangles needs to be modified slightly to handle the case of constant velocities. The velocity of the first triangle is subtracted from the velocity of the second triangle so that all calculations are done relative to a stationary first triangle. If the triangle velocities are $\vec{V}_0$ and $\vec{V}_1$, define the relative velocity to be $\vec{W} = \vec{V}_1 - \vec{V}_0$. Let the initial time and final time of the calculations be $t = 0$ and $t = T$.

Axes $\vec{N}$ or $\vec{M}$. One triangle projects to a single p–value. The other triangle projects to three moving q–values. Abstractly the problem is to show that $\min(q+tw, q+d_0+tw, q+d_1+tw) > p$ or $\max(q+tw, q+d_0+tw, q+d_1+tw) < p$ for $t \in [0, T]$. Pseudocode is

```
if ( q > p )
{
    if ( d0 >= 0 )
    {
        if ( d1 >= 0 )
        {
            min = q;
            if ( min+T*w > p )
                return no_intersection;
        }
        else
        {
            min = q+d1;
            if ( min > p && min+T*w > p )
                return no_intersection;
        }
    }
```

```
        else if ( d1 <= d0 )
        {
            min = q+d1;
            if ( min > p && min+T*w > p )
                return no_intersection;
        }
        else
        {
            min = q+d0;
            if ( min > p && min+T*w > p )
                return no_intersection;
        }
    }
    else if ( q < p )
    {
        if ( d0 <= 0 )
        {
            if ( d1 <= 0 )
            {
                max = q;
                if ( max+T*w < p )
                    return no_intersection;
            }
            else
            {
                max = q+d1;
                if ( max < p && max+T*w < p )
                    return no_intersection;
            }
        }
        else if ( d1 >= d0 )
        {
            max = q+d1;
            if ( max < p && max+T*w < p )
                return no_intersection;
        }
        else
        {
            max = q+d0;
            if ( max < p && max+T*w < p )
                return no_intersection;
        }
    }
}
```

Axes $\vec{E}_i \times \vec{F}_j$. The first triangle projects to 0 and p. The second triangle projects to $q+tw$ and $q+d+tw$ for $t \in [0,T]$. The abstract problem is to show that $\min(q+tw, q+d+tw) > \max(0,p)$ or $\max(q+tw, q+d+tw) < \min(0,p)$. Pseudocode is

```
if ( p >= 0 )
{
    if ( q < 0 )
    {
        if ( d <= 0 )
        {
            max = q;
            if ( max+T*w < 0 )
                return no_intersection;
        }
        else
        {
            max = q+d;
            if ( max < 0 && max+T*w < 0 )
                return no_intersection;
        }
    }
    else if ( q > p )
    {
        if ( d >= 0 )
        {
            min = q;
            if ( min+T*w > p )
                return no_intersection;
        }
        else
        {
            min = q+d;
            if ( min > p && min+T*w > p )
                return no_intersection;
        }
    }
}
else
{
    if ( q > 0 )
    {
        if ( d >= 0 )
        {
            min = q;
            if ( min+T*w > 0 )
                return no_intersection;
        }
        else
        {
            min = q+d;
            if ( min > 0 && min+T*w > 0 )
                return no_intersection;
```

```
            }
        }
        else if ( q < p )
        {
            if ( d <= 0 )
            {
                max = q;
                if ( max+T*w < p )
                    return no_intersection;
            }
            else
            {
                max = q+d;
                if ( max < p && max+T*w < p )
                    return no_intersection;
            }
        }
    }
```

4.3 Finding an Intersection of Noncoplanar Triangles

4.3.1 Finding the First Time of Intersection

Given that the two triangles do not intersect at time $t = 0$, but do intersect at some later time, a simple modification of the algorithm for testing for an intersection provides the first time of intersection. The first time is computed as the *maximum* time $T > 0$ for which there is at least one separating axis for any $t \in [0, T)$, but for which no separating axis exists at time T. The idea is to test each potential separating axis and keep track of the time at which the intervals of projection intersect for the first time. The largest such time is the first time at which the triangles intersect. Also, it is important to keep track of which side each of the intervals is relative to the other interval. Finally, knowing the separating axis associated with the maximum time T allows us to reconstruct a point of intersection.

4.3.2 Finding a Point of Intersection

If T is the first time of intersection, the problem is to find a point P in the intersection of the two triangles. Since the triangles are not coplanar, the only possibilities for the set of intersections is a single point or a line segment. We need to solve

$$x_0\vec{E}_0 + x_1\vec{E}_1 = \vec{D} + y_0\vec{F}_0 + y_1\vec{F}_1 \tag{19}$$

where $\vec{D} = (\vec{B}_0 + T\vec{V}_1) - (\vec{A}_0 + T\vec{V}_0)$; for x_i with $0 \le x_0 \le 1$, $0 \le x_1 \le 1$, $x_0 + x_1 \le 1$; and for y_j with $0 \le y_0 \le 1$, $0 \le y_1 \le 1$, $y_0 + y_1 \le 1$.

Equation (19) can be solved for each variable individually by crossing then dotting equation with the proper vectors. The solutions are

$$\begin{aligned} x_i &= \frac{1-2i}{|\vec{N}|^2}\left(\vec{N}\cdot\vec{D}\times\vec{E}_{1-i} + (\vec{N}\cdot\vec{F}_0\times\vec{E}_{1-i})y_0 + (\vec{N}\cdot\vec{F}_1\times\vec{E}_{1-i})y_1\right) \\ y_j &= \frac{1-2j}{|\vec{M}|^2}\left(-\vec{M}\cdot\vec{D}\times\vec{F}_1 + (\vec{M}\cdot\vec{E}_0\times\vec{F}_{1-j})x_0 + (\vec{M}\cdot\vec{E}_1\times\vec{F}_{1-j})x_1\right) \end{aligned}$$

for $i = 0, 1$ and $j = 0, 1$. Each of these equations defines the left–hand side as a linear function of the variables in the right–hand side. The extreme values occur at the vertices of the triangular domain of the function: $(0, 0)$, $(1, 0)$, and $(0, 1)$. The extreme values of the equations define intervals bounding each variable, $x_i \in [\min(x_i), \max(x_i)]$ and $y_j \in [\min(y_j), \max(y_j)]$. The interval end points are

$$\begin{aligned}
\min(x_i) &= \tfrac{1}{|\vec{N}|^2} \min\left((1-2i)\phi_0(i), (1-2i)\phi_1(i), (1-2i)\phi_2(i)\right), \\
\max(x_i) &= \tfrac{1}{|\vec{N}|^2} \max\left((1-2i)\phi_0(i), (1-2i)\phi_1(i), (1-2i)\phi_2(i)\right), \\
\min(y_j) &= \tfrac{1}{|\vec{M}|^2} \min\left((1-2j)\psi_0(j), (1-2j)\psi_1(j), (1-2j)\psi_2(j)\right), \\
\max(y_j) &= \tfrac{1}{|\vec{M}|^2} \max\left((1-2j)\psi_0(j), (1-2j)\psi_1(j), (1-2j)\psi_2(j)\right),
\end{aligned} \tag{20}$$

where $\phi_0(i) = \vec{N} \cdot \vec{D} \times \vec{E}_{1-i}$, $\phi_1 = \phi_0(i) + \vec{N} \cdot \vec{F}_0 \times \vec{E}_{1-i}$, $\phi_2(i) = \phi_0(i) + \vec{N} \cdot \vec{F}_1 \times \vec{E}_{1-i}$, $\psi_0(j) = -\vec{M} \cdot \vec{D} \times \vec{F}_{1-j}$, $\psi_1(j) = \psi_0(j) + \vec{M} \cdot \vec{E}_0 \times \vec{F}_{1-j}$, and $\psi_2(j) = \psi_0(j) + \vec{M} \cdot \vec{E}_1 \times \vec{F}_{1-j}$.

In the following constructions of the first point of intersection, if any of the variables is not uniquely constrained by the derived equations, then such a variable can be selected from the intervals $[\min(x_i), \max(x_i)]$ or $[\min(y_j), \max(y_j)]$ and subject to triangular domain constraints for that variable.

Last Separating Axis $\vec{N}$. Dot equation (19) with $\vec{N}$ to obtain

$$0 = \vec{N} \cdot \vec{D} + y_0 \vec{N} \cdot \vec{F}_0 + y_1 \vec{N} \cdot \vec{F}_1.$$

The projection of the first triangle's vertices leads to a single p value of 0. The projection of the second triangle's vertices leads to (possibly) distinct q values, $\{q_0, q_1, q_2\}$, defined in Table 5. There are three cases to consider.

The first case is $q_0 = \min_i(q_i)$ in which case $\sigma = 1$ or $q_0 = \max_i(q_i)$ in which case $\sigma = -1$. Then $\vec{N} \cdot \vec{D} = 0$, $\sigma \vec{N} \cdot \vec{F}_0 \geq 0$, $\sigma \vec{N} \cdot \vec{F}_1 \geq 0$, and

$$0 = (\sigma \vec{N} \cdot \vec{F}_0) y_0 + (\sigma \vec{N} \cdot \vec{F}_1) y_1. \tag{21}$$

If $\vec{N} \cdot \vec{F}_0 \neq 0$ and $\vec{N} \cdot \vec{F}_1 \neq 0$, then $y_0 = 0$ and $y_1 = 0$ are required. If $\vec{N} \cdot \vec{F}_0 = 0$ and $\vec{N} \cdot \vec{F}_1 \neq 0$, equation (21) requires $y_1 = 0$, but does not constraint y_0. A point of intersection is provided by any $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (20). If $\vec{N} \cdot \vec{F}_0 \neq 0$ and $\vec{N} \cdot \vec{F}_1 = 0$, equation (21) requires $y_0 = 0$, but does not constrain y_1. A point of intersection is provided by any $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (20). If both $\vec{N} \cdot \vec{F}_0 = 0$ and $\vec{N} \cdot \vec{F}_1 = 0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. While the assumption of this section is that the two vectors are not parallel, numerical error might generate this case. Discussion of intersection of coplanar triangles occurs later in this document.

The second case is $q_1 = \min_i(q_i)$ in which case $\sigma = 1$ or $q_1 = \max_i(q_i)$ in which case $\sigma = -1$. Then $\vec{N} \cdot \vec{D} = -\vec{N} \cdot \vec{F}_0$, $\sigma \vec{N} \cdot \vec{F}_0 \leq 0$, $\sigma(\vec{N} \cdot \vec{F}_1 - \vec{N} \cdot \vec{F}_0) \geq 0$, and

$$0 = (-\sigma \vec{N} \cdot \vec{F}_0)(1 - y_0 - y_1) + [\sigma(\vec{N} \cdot \vec{F}_1 - \vec{N} \cdot \vec{F}_0)] y_1. \tag{22}$$

If $\vec{N} \cdot \vec{F}_0 \neq 0$ and $\vec{N} \cdot \vec{F}_1 \neq \vec{N} \cdot \vec{F}_0$, then $y_0 + y_1 = 1$ and $y_1 = 0$ are required, Therefore, $y_0 = 1$ and $y_1 = 0$. If $\vec{N} \cdot \vec{F}_0 = 0$ and $\vec{N} \cdot \vec{F}_1 \neq \vec{N} \cdot \vec{F}_0$, equation (22) requires $y_1 = 0$, but does not constrain y_0. In this case $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (20). If $\vec{N} \cdot \vec{F}_0 \neq 0$ and $\vec{N} \cdot \vec{F}_1 = \vec{N} \cdot \vec{F}_0$, equation (22) requires $y_0 + y_1 = 1$, but does not constrain y_1. In this case $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (20). Given a choice for

y_1, set $y_0 = 1 - y_1$. If both $\vec{N} \cdot \vec{F}_0 = 0$ and $\vec{N} \cdot \vec{F}_1 = \vec{N} \cdot \vec{F}_0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. As before, the discussion of intersection of coplanar triangles is postponed until later in this document.

The third case is $q_2 = \min_i(q_i)$ in which case $\sigma = 1$ or $q_2 = \max_i(q_i)$ in which case $\sigma = -1$. Then $\vec{N} \cdot \vec{D} = -\vec{N} \cdot \vec{F}_1$, $\sigma(\vec{N} \cdot \vec{F}_0 - \vec{N} \cdot \vec{F}_1) \geq 0$, $\sigma \vec{N} \cdot \vec{F}_1 \leq 0$, and

$$0 = [\sigma(\vec{N} \cdot \vec{F}_0 - \vec{N} \cdot \vec{F}_1)]y_0 + (-\sigma \vec{N} \cdot \vec{F}_1)(1 - y_0 - y_1). \tag{23}$$

If $\vec{N} \cdot \vec{F}_0 \neq \vec{N} \cdot \vec{F}_1$ and $\vec{N} \cdot \vec{F}_1 \neq 0$, then $y_0 = 0$ and $y_0 + y_1 = 1$ are required. Therefore, $y_0 = 0$ and $y_1 = 1$. If $\vec{N} \cdot \vec{F}_0 = \vec{N} \cdot \vec{F}_1$ and $\vec{N} \cdot \vec{F}_1 \neq 0$, equation (23) requires $y_0 + y_1 = 1$, but does not constrain y_0. In this case $y_0 \in [\min(y_0), \max(y_0)] \cap [0, 1]$ where $\min(y_0)$ and $\max(y_0)$ are defined in equation (20). Given a choice for y_0, set $y_1 = 1 - y_0$. If $\vec{N} \cdot \vec{F}_0 \neq \vec{N} \cdot \vec{F}_1$ and $\vec{N} \cdot \vec{F}_1 = 0$, equation (23) requires $y_0 = 0$, but does not constrain y_1. In this case $y_1 \in [\min(y_1), \max(y_1)] \cap [0, 1]$ where $\min(y_1)$ and $\max(y_1)$ are defined in equation (20). If both $\vec{N} \cdot \vec{F}_0 = \vec{N} \cdot \vec{F}_1$ and $\vec{N} \cdot \vec{F}_1 = 0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. As before, the discussion of intersection of coplanar triangles is postponed until later in this document.

Last Separating Axis $\vec{M}$. Dot equation (19) with $\vec{M}$ to obtain

$$x_0 \vec{M} \cdot \vec{E}_0 + x_1 \vec{M} \cdot \vec{E}_1 = \vec{M} \cdot \vec{D}.$$

The projection of the first triangle's vertices leads to (possibly) distinct p values $\{p_0, p_1, p_2\}$. The projection of the second triangle's vertices leads to a single q value, q_0, all defined in Table 5. There are three cases to consider.

The first case is $p_0 = \min_i(p_i)$ in which case $\sigma = -1$ or $p_0 = \max_i(p_i)$ in which case $\sigma = 1$. Then $\vec{M} \cdot \vec{D} = 0$, $\sigma \vec{M} \cdot \vec{E}_0 \leq 0$, $\sigma \vec{M} \cdot \vec{E}_1 \leq 0$, and

$$(\sigma \vec{M} \cdot \vec{E}_0)x_0 + (\sigma \vec{M} \cdot \vec{E}_1)x_1 = 0. \tag{24}$$

If $\vec{M} \cdot \vec{E}_0 \neq 0$ and $\vec{M} \cdot \vec{E}_1 \neq 0$, then $x_0 = 0$ and $x_1 = 0$ are required. If $\vec{M} \cdot \vec{E}_0 = 0$ and $\vec{M} \cdot \vec{E}_1 \neq 0$, equation (24) requires $x_1 = 0$, but does not constrain x_0. A point of intersection is provided by any $x_0 \in [\min(x_0), \max(x_0)] \cap [0, 1]$ where $\min(x_0)$ and $\max(x_0)$ are defined in equation (20). If $\vec{M} \cdot \vec{E}_0 \neq 0$ and $\vec{M} \cdot \vec{E}_1 = 0$, equation (24) requires $x_0 = 0$, but does not constrain x_1. A point of intersection is provided by any $x_1 \in [\min(x_1), \max(x_1)] \cap [0, 1]$ where $\min(x_1)$ and $\max(x_1)$ are defined in equation (20). If both $\vec{M} \cdot \vec{E}_0 = 0$ and $\vec{M} \cdot \vec{E}_1 = 0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. While the assumption of this section is that the two vectors are not parallel, numerical error might generate this case. Discussion of intersection of coplanar triangles occurs later in this document.

The second case is $p_1 = \min_i(p_i)$ in which case $\sigma = -1$ or $p_1 = \max_i(p_i)$ in which case $\sigma = 1$. Then $\vec{M} \cdot \vec{D} = \vec{M} \cdot \vec{E}_0$, $\sigma \vec{M} \cdot \vec{E}_0 \geq 0$, $\sigma(\vec{M} \cdot \vec{E}_1 - \vec{M} \cdot \vec{E}_0) \leq 0$, and

$$0 = (-\sigma \vec{M} \cdot \vec{E}_0)(1 - x_0 - x_1) + [\sigma(\vec{M} \cdot \vec{E}_1 - \vec{M} \cdot \vec{E}_0)]x_1. \tag{25}$$

If $\vec{M} \cdot \vec{E}_0 \neq 0$ and $\vec{M} \cdot \vec{E}_1 \neq \vec{M} \cdot \vec{E}_0$, then $x_0 + x_1 = 1$ and $x_1 = 0$ are required. Therefore, $x_0 = 1$ and $x_1 = 0$. If $\vec{M} \cdot \vec{E}_0 = 0$ and $\vec{M} \cdot \vec{E}_1 \neq \vec{M} \cdot \vec{E}_0$, equation (25) requires $x_1 = 0$, but does not constrain x_0. In this case $x_0 \in [\min(x_0), \max(x_0)] \cap [0, 1]$ where $\min(x_0)$ and $\max(x_0)$ are defined in equation (20). If $\vec{M} \cdot \vec{E}_0 \neq 0$ and $\vec{M} \cdot \vec{E}_1 = \vec{M} \cdot \vec{E}_0$, equation (25) requires $x_0 + x_1 = 1$, but does not constrain x_1. In this case $x_1 \in [\min(x_1), \max(x_1)] \cap [0, 1]$ where $\min(x_1)$ and $\max(x_1)$ are defined in equation (20). Given a choice for x_1, set $x_0 = 1 - x_1$. If both $\vec{M} \cdot \vec{E}_0 = 0$ and $\vec{M} \cdot \vec{E}_1 = \vec{M} \cdot \vec{E}_0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. As before, the discussion of intersection of coplanar triangles is postponed until later in this document.

The third case is $p_2 = \min_i(p_i)$ in which case $\sigma = -1$ or $p_2 = \max_i(p_i)$ in which case $\sigma = 1$. Then $\vec{M} \cdot \vec{D} = \vec{M} \cdot \vec{E}_1$, $\sigma(\vec{M} \cdot \vec{E}_0 - \vec{M} \cdot \vec{E}_1) \le 0$, $\sigma \vec{M} \cdot \vec{E}_1 \ge 0$, and

$$0 = [\sigma(\vec{M} \cdot \vec{E}_1 - \vec{M} \cdot \vec{E}_0)]x_0 + (\sigma \vec{M} \cdot \vec{E}_1)(1 - x_0 - x_1). \tag{26}$$

If $\vec{M} \cdot \vec{E}_0 \neq \vec{M} \cdot \vec{E}_1$ and $\vec{M} \cdot \vec{E}_1 \neq 0$, then $x_0 = 0$ and $x_0 + x_1 = 1$ are required. Therefore, $x_0 = 0$ and $x_1 = 1$. If $\vec{M} \cdot \vec{E}_0 = \vec{M} \cdot \vec{E}_1$ and $\vec{M} \cdot \vec{E}_1 \neq 0$, equation (25) requires $x_0 + x_1 = 1$, but does not constrain x_0. In this case $x_0 \in [\min(x_0), \max(x_0)] \cap [0, 1]$ where $\min(x_0)$ and $\max(x_0)$ are defined in equation (20). Given a choice for x_0, set $x_1 = 1 - x_0$. If $\vec{M} \cdot \vec{E}_0 \neq \vec{M} \cdot \vec{E}_1$ and $\vec{M} \cdot \vec{E}_1 = 0$, equation (25) requires $x_0 = 0$, but does not constrain x_1. In this case $x_1 \in [\min(x_1), \max(x_1)] \cap [0, 1]$ where $\min(x_1)$ and $\max(x_1)$ are defined in equation (20). If both $\vec{M} \cdot \vec{E}_0 = \vec{M} \cdot \vec{E}_1$ and $\vec{M} \cdot \vec{E}_1 = 0$, then $\vec{N}$ and $\vec{M}$ must be parallel and the triangles must be coplanar. As before, the discussion of intersection of coplanar triangles is postponed until later in this document.

Last Separating Axis $\vec{E}_i \times \vec{F}_j$. Let (i_0, i_1, i_2) and (j_0, j_1, j_2) be permutations of $(0, 1, 2)$ in the set $\{(0, 1, 2), (1, 0, 2), (2, 1, 0)\}$. The functions α, β, γ, and δ are the same used in the section on finding intersections between OBBs and triangles.

Dot equation (19) with $\vec{E}_{i_0} \times \vec{F}_{j_0}$ to obtain

$$(\vec{E}_0 \cdot \vec{E}_{i_0} \times \vec{F}_{j_0})x_0 + (\vec{E}_1 \cdot \vec{E}_{i_0} \times \vec{F}_{j_0})x_1 = \vec{D} \cdot \vec{E}_{i_0} \times \vec{F}_{j_0} + (\vec{F}_0 \cdot \vec{E}_{i_0} \times \vec{F}_{j_0})y_0 + (\vec{F}_1 \cdot \vec{E}_{i_0} \times \vec{F}_{j_0})y_1.$$

Using the various identities mentioned earlier, the equation reduces to

$$\alpha(i_0)(\vec{N} \cdot \vec{F}_{j_0})x_0 + \gamma(i_0)(\vec{N} \cdot F_{j_0})x_1 = \vec{D} \cdot \vec{E}_{i_0} \times \vec{F}_{j_0} - \alpha(j_0)(\vec{M} \cdot \vec{E}_{i_0})y_0 - \gamma(j_0)(\vec{M} \cdot \vec{E}_{i_0})y_1. \tag{27}$$

The projection of the first triangle's vertices leads to two distinct p values, $p_0 = 0$ and $p_1 = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$. The projection of the second triangle's vertices leads to two distinct q values, $q_0 = \vec{D} \cdot \vec{E}_{i_0} \times \vec{F}_{j_0}$ and $q_1 = q_0 - \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}\}$. There are four cases to consider depending on which of the projection values are minima or maxima. In each case I derive the solutions when the intersection point is unique. Nonuniqueness is discussed after the four cases.

The four cases each require two additional constraints on the variables. Dotting equation (19) with $\vec{M}$ and $\vec{N}$ yields equations

$$\begin{aligned} \vec{M} \cdot \vec{E}_0 x_0 + \vec{M} \cdot \vec{E}_1 x_1 &= \vec{M} \cdot \vec{D} \\ \vec{N} \cdot \vec{F}_0 y_0 + \vec{N} \cdot \vec{F}_1 y_1 &= -\vec{N} \cdot \vec{D}. \end{aligned} \tag{28}$$

The first case is $\min(q) = q_0$ and $\max(p) = 0$ in which case $\sigma = 1$ or $\max(q) = q_0$ and $\min(p) = 0$ in which case $\sigma = -1$. Then $q_0 = 0$, $\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0} \le 0$, and $\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0} \le 0$. Equation (27) is equivalent to

$$0 = [-\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0}][\alpha(i_0)x_0 + \beta(i_0)x_1] + [-\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0}][\alpha(j_0)y_0 + \beta(j_0)y_1].$$

If $\vec{N} \cdot \vec{F}_{j_0} \neq 0$ and $\vec{M} \cdot \vec{E}_{i_0} \neq 0$, then $\alpha(i_0)x_0 + \beta(i_0)x_1 = 0$ and $\alpha(j_0)y_0 + \beta(j_0)y_1 = 0$ are required. These two constraints and equations (28) uniquely determine x_0, x_1, y_0, and y_1.

The second case is $\min(q) = q_0$ and $\max(p) = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$ in which case $\sigma = 1$ or $\max(q) = q_0$ and $\min(p) = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$ in which case $\sigma = -1$. Then $q_0 = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$, $\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0} \ge 0$, and $\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0} \le 0$. Equation (27) is equivalent to

$$0 = [\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0}][1 - \alpha(i_0)x_0 - \beta(i_0)x_1] + [-\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0}][\alpha(j_0)y_0 + \beta(j_0)y_1].$$

If $\vec{N} \cdot \vec{F}_{j_0} \neq 0$ and $\vec{M} \cdot \vec{E}_{i_0} \neq 0$, then $\alpha(i_0)x_0 + \beta(i_0)x_1 = 1$ and $\alpha(j_0)y_0 + \beta(j_0)y_1 = 0$ are required. These two constraints and equations (28) uniquely determine x_0, x_1, y_0, and y_1.

The third case is $\min(q) = q_0 - \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$ and $\max(p) = 0$ in which case $\sigma = 1$ or $\max(q) = q_0 - \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$ and $\min(p) = 0$ in which case $\sigma = -1$. Then $q_0 = \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$, $\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0} \leq 0$, and $\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0} \geq 0$. Equation (27) is equivalent to

$$0 = [-\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0}][\alpha(i_0)x_0 + \beta(i_0)x_1] + [\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0}][1 - \alpha(j_0)y_0 - \beta(j_0)y_1].$$

If $\vec{N} \cdot \vec{F}_{j_0} \neq 0$ and $\vec{M} \cdot \vec{E}_{i_0} \neq 0$, then $\alpha(i_0)x_0 + \beta(i_0)x_1 = 0$ and $\alpha(j_0)y_0 + \beta(j_0)y_1 = 1$ are required. These two constraints and equations (28) uniquely determine x_0, x_1, y_0, and y_1.

The fourth case is $\min(q) = q_0 - \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$ and $\max(p) = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$ in which case $\sigma = 1$ or $\max(q) = q_0 - \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$ and $\min(p) = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0}$ in which case $\sigma = -1$. Then $q_0 = \delta(i_0)\vec{N} \cdot \vec{F}_{j_0} + \delta(j_0)\vec{M} \cdot \vec{E}_{i_0}$, $\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0} \geq 0$, and $\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0} \geq 0$. Equation (27) is equivalent to

$$0 = [\sigma\delta(i_0)\vec{N} \cdot \vec{F}_{j_0}][1 - \alpha(i_0)x_0 - \beta(i_0)x_1] + [\sigma\delta(j_0)\vec{M} \cdot \vec{E}_{i_0}][1 - \alpha(j_0)y_0 - \beta(j_0)y_1].$$

If $\vec{N} \cdot \vec{F}_{j_0} \neq 0$ and $\vec{M} \cdot \vec{E}_{i_0} \neq 0$, then $\alpha(i_0)x_0 + \beta(i_0)x_1 = 1$ and $\alpha(j_0)y_0 + \beta(j_0)y_1 = 1$ are required. These two constraints and equations (28) uniquely determine x_0, x_1, y_0, and y_1.

The coefficients needed to produce the unique points of intersection are summarized in the following tables.

$\vec{L}$	coefficients
$\vec{N}$	$y_0 = 0,\ y_1 = 0, \quad \sigma q_0 = \min_k(\sigma q_k)$
	$y_0 = 1,\ y_1 = 0, \quad \sigma q_1 = \min_k(\sigma q_k)$
	$y_0 = 0,\ y_1 = 1, \quad \sigma q_2 = \min_k(\sigma q_k)$
$\vec{M}$	$x_0 = 0,\ x_1 = 0, \quad \sigma p_0 = \min_k(\sigma p_k)$
	$x_0 = 1,\ x_1 = 0, \quad \sigma p_1 = \min_k(\sigma p_k)$
	$x_0 = 0,\ x_1 = 1, \quad \sigma p_2 = \min_k(\sigma p_k)$
$\vec{E}_0 \times \vec{F}_j$	$x_1 = 0,\ x_0 = \vec{M} \cdot \vec{D}/\vec{M} \cdot \vec{E}_0, \quad 0 = \max_k(\sigma p_k)$
	$x_1 = 1,\ x_0 = 0, \quad 0 = \min_k(\sigma p_k)$
$\vec{E}_1 \times \vec{F}_j$	$x_0 = 0,\ x_1 = \vec{M} \cdot \vec{D}/\vec{M} \cdot \vec{E}_1, \quad 0 = \max_k(\sigma p_k)$
	$x_0 = 1,\ x_1 = 0, \quad 0 = \min_k(\sigma p_k)$
$\vec{E}_2 \times \vec{F}_j$	$x_0 = 0,\ x_1 = 0, \quad 0 = \max_k(\sigma p_k)$
	$x_0 = (\vec{M} \cdot \vec{E}_1 - \vec{M} \cdot \vec{D})/\vec{M} \cdot \vec{E}_2, \quad 0 = \min_k(\sigma p_k)$

Table 7. Coefficients for unique points of TRI–TRI intersection.

5 Processing of Moving Objects

Consider a rigid, moving object with a central point $\vec{K}(t)$ and a frame field $F(t)$, a 3×3 rotation matrix whose columns represent a coordinate system with origin $\vec{K}(t)$, where $t \in [0, T]$. The frame field does not necessarily have to be the Frenet frame for the curve traversed by the central point. The path followed by any other point $\vec{X}_0$ is $\vec{X}(t) = \vec{K}(t) + F(t)F(0)^T(\vec{X}_0 - \vec{K}(0))$. If G_0 is a frame field for $\vec{X}_0$, then the frame field for $\vec{X}(t)$ is $G(t) = F(t)F(0)^T G_0$.

In particular, the path and orientation of each triangle in the mesh representing the object is derivable from the path and orientation of the central point of the object. The path of an OBB center and the axes of the OBB are similarly derivable. The tests for nonintersection of OBB–OBB, triangle–OBB, and triangle–triangle for two rigid, moving objects are equivalent to showing that the minimum of a function of time with domain $[0, T]$ is positive. For example, the nonintersection test of OBB–OBB is of the form $R(t) - R_0(t) - R_1(t) > 0$ for $t \in [0, T]$.

Computing the minimum of a function can be an expensive operation, especially in the context of a dynamic collision detection system that is used in a real–time environment. Rather than minimize the given function $f(t)$ for $t \in [0, T]$, approximate f by a piecewise linear function by computing $N \geq 2$ samples $t_i = i*T/(N-1)$ and $f_i = f(t_i)$ for $0 \leq i \leq N-1$. On each subinterval $[t_i, t_{i+1}]$ the central point of the object is assumed to have constant velocity $\vec{V} = \vec{K}(t_{i+1}) - \vec{K}(t_i)$. The nonintersection tests on the subinterval and with the specified constant velocity are exactly those derived in the previous sections of this document. The application can select N as desired, the choice probably dependent on frame rate (or number of cycles to spare per frame for the collision system).

Of particular interest is the situation where the object motion is defined by the system of differential equations

$$\frac{d\vec{X}}{dt} = \vec{V} + \vec{W} \times (\vec{X} - \vec{K}_0)$$

where $\vec{K}_0$ is the initial location for the central point of the object, $\vec{V}$ is a constant tangential velocity, and $\vec{W}$ is a constant angular velocity. The length of $\vec{V}$ is the constant tangential speed and the length of $\vec{W}$ is the constant angular speed. The solution to the differential equation for initial condition $\vec{X}(0) = \vec{X}_0$ is

$$\vec{X}(t) = \vec{K}_0 + t\vec{V} + R(t, \vec{W})(\vec{X}_0 - \vec{K}_0)$$

where $R(t, \vec{W})$ is a rotation matrix about the axis $\vec{K}_0 + t\vec{W}$. The central path is $\vec{K}(t) = \vec{K}_0 + t\vec{V}$. The difference of the two paths $\vec{X}(t)$ and $\vec{K}(t)$ is

$$\vec{X}(t) - \vec{K}(t) = R(t, \vec{W})(\vec{X}_0 - \vec{K}_0).$$

The initial frame field $F(0)$ has colums of the form $\vec{X}_0 - \vec{K}_0$, so the frame field at other times is $F(t) = R(t, \vec{W})F(0)$.

Rather than compute the closed form for $F(t)$ which requires evaluation of trigonometric functions, it is better to numerically integrate the system of differential equations to obtain the sample positions. That is, solve $dF/dt = SF$ where S is the skew symmetric matrix defined by the linear operation $\vec{W} \times \vec{X} = S\vec{X}$. Euler's method is certainly the easies to apply, but nothing prevents you from using a higher–order method such as Runge–Kutta. Regardless of numerical integrator, it is necessary to take the iterate representing $F(t)$ and use Gram–Schmidt orthonormalization to guarantee that you have a rotation matrix to work with.

6 Constructing an Oriented Bounding Box Tree

Given a triangular mesh consisting of a collection of vertices and a connectivity list, the basic approach to constructing an OBB tree is recursive. An OBB is computed to contain the initial triangular mesh. The mesh is split ino two submeshes, the algorithm possibly using information about the OBB to determine how to split the mesh. If a submesh contains at least two triangles, then the process is repeated on that submesh. If a submesh has exactly one triangle, no OBB is constructed, but the triangle is considered to be at a leaf node of the tree.

The OBB nodes themselves must store various information to aid in collision detection. I assume that the trimesh represents a rigid body. Dynamically morphed objects are problematic in that OBB trees would need to be recomputed during runtime, an expensive operation. While there are many ways to organize the data, the simplest is to require each OBB node to store a pointer to the rigid body object, an OBB, pointers to the two child OBB nodes, and an index to a triangle.

The pointer to the object is used to query the object about motion information. For example, if the object velocity is a function of time, an OBB node may need to query the object to determine at a specific time what the velocity is. The pointers to the children are both not null for interior OBB nodes and both null for leaf OBB nodes. The index to a triangle is only used at OBB leaf nodes. This index is used in querying the object to get the actual triangle vertex data that is needed to compute triangle–triangle intersections.

The tree generation algorithm also allows for building less than a full tree. An application can specify a threshold on the number of triangles for an OBB leaf node. The full tree has a single triangle per leaf node. However, if an application specifies at least two triangles per leaf node, the splitting algorithm will be applied during construction of an OBB node only if that node has more than two triangles in its mesh. The number is heuristic in that an OBB node with three triangles is allowed to be split. The child with two triangles is no longer subdivided. The other child has a single triangle.

6.1 Generating an OBB for a Trimesh

A variety of methods can be used for computing an OBB for a triangular mesh. In real–time applications, these methods are applied in a preprocessing phase, so their execution times are not typically an issue. The three algorithms discussed are: minimum volume OBB, OBB based on distribution of mesh points, and OBB based on distribution of mesh triangles. The last type of OBB is what is described in [1].

6.1.1 Minimum Volume OBB

Ideally this provides the best fit of a mesh in the sense that the OBB requires the minimum volume of space of all the possible OBBs that can fit the mesh. Whether or not this is the best practical choice I leave up to the implementor.

Given a collection of points $\vec{X}_j$ for $0 \le j < n$, an OBB that fits the points can be constructed for each choice of coordinate axes $\vec{A}_i$, $i = 0, 1, 2$. The points are projected onto the axes $\vec{X}_0 + s\vec{A}_i$, the values being $\rho_{ij} = \vec{A}_i \cdot (\vec{X}_j - \vec{X}_0)$ for all j. Define $\alpha_i = \min_j(\rho_{ij})$, $\beta_i = \max_j(\rho_{ij})$, and $\gamma_i = (\alpha_i + \beta_i)/2$. The center of

the smallest volume OBB with specified axes is

$$\vec{C} = \vec{X}_0 + \sum_{i=0}^{2} \gamma_i \vec{A}_i.$$

The extents of the OBB are $a_i = (\beta_i - \alpha_i)/2$.

Each set of coordinate axes can be represented as the columns of rotation matrices. Each rotation matrix is generated by a unit-length vector $\vec{U}$ and an angle $\theta \in [-, 2\pi]$. The mapping from rotation matrices to coordinate axes is of course not one–to–one. However, the volume of the OBBs can be viewed as a function $V : S^2 \times [0, 2\pi] \to [0, \infty)$ where S^2 is the unit sphere. The volume is $V(\vec{U}, \theta) = \prod_{i=0}^{2}(\beta_i - \alpha_i)$. This function is continuous on its compact domain, so from calculus it must attain its minimum on that domain. Therefore, there exists an axis $\vec{U}_0$ and an angle θ_0 for which $V(\vec{U}_0, \theta_0) \leq V(\vec{U}, \theta)$ for all axes $\vec{U}$ and all angles θ.

The construction of $\vec{U}_0$ and θ_0 can be implemented as a numerical minimization using techniques that do not require derivatives. A good choice is Powell's direction set method, [3]. The rate of convergence to the minimum depends on initial guess for axis and angle. The algorithm discussed next provides a reasonable initial guess for the minimizer.

6.1.2 OBB from Points

An OBB can be constructed by fitting the mesh points $\vec{X}_j$ for $0 \leq j < n$ with an anisotropic Gaussian distribution. The center of the OBB is the mean of the points,

$$\vec{C} = \frac{1}{n} \sum_{j=0}^{n} \vec{X}_j.$$

The axes of the OBB are selected as unit–length eigenvectors of the covariance matrix

$$M = \frac{1}{n} \sum_{j=0}^{n-1} (\vec{X}_j - \vec{C})(\vec{X}_j - \vec{C})^T$$

where T indicates transpose. If $\vec{A}_i$ are unit-length eigenvectors, the extents along those axes are $a_i = \max_j |\vec{A}_i \cdot (\vec{X}_j - \vec{C})|$.

A minor variation that leads to a slightly better fit is to compute just the eigenvectors of the covariance matrix and compute the intervals of projection as in the case of building a minimum volume OBB. The center of the OBB is computed the same way as in the minimum volume algorithm.

The main problem with this approach is that the box orientation can be heavily influenced by mesh points that are interior to the convex hull of the points. As in the algorithm for minimum volume OBB construction, the box orientation should depend only on the convex hull of the mesh points. The solution is to first apply a convex hull algorithm, then process only those points in the covariance matrix.

6.1.3 OBB from Triangles

The fit of an OBB to the convex hull of the mesh points given previously still has problems with sampling. The mesh points on the convex hull may be irregularly distributed so that a small, dense collection of points

can unfairly affect the orientation of the bounding box. This effect can be minimized by using a continuous formulation of the covariance matrix.

Suppose there are ℓ triangles. If the i^{th} triangle has vertices $\vec{V}_{0,i}$, $\vec{V}_{1,i}$, and $\vec{V}_{i,2}$, then the triangle and its interior are represented by $\vec{X}_i(s,t) = \vec{V}_{0,i} + s(\vec{V}_{1,i} - \vec{V}_{0,i}) + t(\vec{V}_{2,i} - \vec{V}_{0,i})$ for $0 \le s \le 1$, $0 \le t \le 1$, and $s+t \le 1$. Let $m_i = |(\vec{V}_{1,i} - \vec{V}_{0,i}) \times (\vec{V}_{2,i} - \vec{V}_{0,i})|/2$ be the area of the triangle. Define the weights $w_i = m_i / \sum_{i=0}^{\ell-1} m_i$. The mean point of the convex hull is

$$\begin{aligned} \vec{C} &= \tfrac{2}{\ell} \textstyle\sum_{i=0}^{\ell-1} w_i \int_0^1 \int_0^{1-t} \vec{X}_i(s,t)\, ds\, dt \\ &= \tfrac{1}{3\ell} \textstyle\sum_{i=0}^{\ell-1} w_i \left(\sum_{j=0}^{2} \vec{V}_{j,i} \right) \end{aligned}$$

and the covariance matrix of the convex hull is

$$\begin{aligned} M &= \tfrac{2}{\ell} \textstyle\sum_{i=0}^{\ell-1} w_i \int_0^1 \int_0^{1-t} (\vec{X}_i(s,t) - \vec{C})(\vec{X}_i(s,t) - \vec{C})^T\, ds\, dt \\ &= \tfrac{1}{12\ell} \textstyle\sum_{i=0}^{\ell-1} w_i \left(\sum_{j=0}^{2} \sum_{k=0}^{2} (\vec{V}_{j,i} - \vec{C})(\vec{V}_{k,i} - \vec{C})^T \right). \end{aligned}$$

If $\vec{A}_i$ are unit-length eigenvectors, the extents along those axes are $a_i = \max_j |\vec{A}_i \cdot (\vec{X}_j - \vec{C})|$ where the $\vec{X}_j$ are the vertices. As in the section on fitting points with a Gaussian distribution, a variation allows adjustment of $\vec{C}$ once the axes $\vec{A}_i$ are known.

6.2 Splitting a Trimesh

Given a triangular mesh with corresponding oriented bounding box, the mesh can be split into two submeshes. The idea is to split the OBB by a plane orthgonal to the longest axis of the box, then partition the triangles based on which side of the splitting plane their centers lie. There are many heuristics for location of splitting plane, but I present only two.

The first algorithm uses the splitting plane orthogonal to the longest axis and passing through the center of the OBB. Because of variations in triangle size, this algorithm may not produce a balanced tree. Worse is that it may not provide a subdivision in that all the triangle centers occur on the same side of the plane. If the longest axis does not partition the triangles, the next longest axis can be used. If in turn this does not partition the triangles, then the last axis is used. If all three axes fail to partition the triangles, then some other criterion for splitting must be used.

The second algorithm uses the splitting plane orthogonal to the longest axis and passing through that point corresponding to the median value of the projection of the triangle centers onto the longest axis. This guarantees that the tree is balanced, a desirable trait since it keeps the height of the tree small compared to the number of triangles represented by the tree.

7 A Simple Dynamic Collision Detection System

There are many choices for testing for collisions between two OBB trees. I present one simple method that implements a dual recursion on the two OBB trees and compares OBBs and triangles for collisions. Effectively an OBB of one tree is compared against an OBB of the other tree. If the two OBBs intersect, then the children of the second OBB are compared against the current OBB of the first tree.

The algorithms assume a function `bool HasObb (ObbTree node)` that returns `true` if and only if the node has an associated OBB. It also assumes a function

```
bool HasChildren (ObbTree node, int depth)
{
    return ( Exists(node.Lchild) && Exists(node.Rchild) && depth != 0 );
}
```

Having children is necessary but not sufficient for this function to return `true`. The test on `depth` supports limiting the depth of traversal. The application specifes a positive depth to limit the traversal. To get a full traversal, the application specifies the depth to be a negative number. The depth is decremented for each recursive call of `TestCollisions`, so in the case of an initial positive depth, any visited node for which current depth value is zero is considered a leaf node. For an initial negative depth, the test for children is unaffected by the subsequent depth values. The semantics of `HasChildren` precludes the calls to `HasObb` being replaced by calls to `HasChildren`.

7.1 Testing for Collision

The method `TestIntersection` calls the appropriate intersection routine based on whether or not the tree node is interior or leaf. The returned value is `true` if and only if the corresponding OBBs or triangles intersect during the specified time interval. Motion parameters are maintained by the object whose pointer is stored in the OBB nodes and can be accessed within the intersection calls.

```
bool TestIntersection (float dt, ObbTree node0, ObbTree node1)
{
    if ( HasObb(node0) )
    {
        if ( HasObb(node1) )
            return ObbObbIntersect(dt,node0.Obb,node1.Obb);
        else
            return ObbTriIntersect(dt,node0.Obb,node1.Tri);
    }
    else
    {
        if ( HasObb(node1) )
            return TriObbIntersect(dt,node0.Tri,node1.Obb);
        else
            return TriTriIntersect(dt,node0.Tri,node1.Tri);
    }
}
```

The values `depth0` and `depth1`, when passed to the `TestCollision` for the root nodes of the OBB trees, are the application–specified maximum depths of traversal for the OBB trees. The returned value for `TestCollision` is `true` if and only if the two subtrees that are rooted at the input nodes do intersect.

```
bool TestCollision (float dt, ObbTree node0, int depth0, ObbTree node1, int depth1)
```

```
{
    if ( !TestIntersection(dt,node0,node1) )
        return false;

    if ( HasChildren(node0,depth0) )
    {
        if ( TestCollision(dt,node0.Lchild,depth0-1,node1,depth1) )
            return true;
        if ( TestCollision(dt,node0.Rchild,depth0-1,node1,depth1) )
            return true;
        if ( HasChildren(node1,depth1) )
        {
            if ( TestCollision(dt,node0,depth0,node1.Lchild,depth1-1) )
                return true;
            if ( TestCollision(dt,node0,depth0,node1.Rchild,depth1-1) )
                return true;
        }
        return false;
    }

    if ( HasChildren(node1,depth1) )
    {
        if ( TestCollision(dt,node0,depth0,node1.Lchild,depth1-1) )
            return true;
        if ( TestCollision(dt,node0,depth0,node1.Rchild,depth1-1) )
            return true;
        return false;
    }

    return true;
}
```

The last line of the function returns `true` since both `node0` and `node1` are at the end of the recursive calls and the call to `TestIntersection` already has shown that the corresponding OBBs or triangles are intersecting. Also note that the semantics of this routine say that if the traversal is limited by an application–specified depth, an intersection between two OBBs or between an OBB and a triangle is counted as a collision, even if the underlying trimesh geometry does not intersect. This illustrates once again the trade–off between accuracy and compute time.

7.2 Finding Collision Points

The method `FindIntersection` calls the appropriate intersection routine based on whether or not the tree node is interior or leaf. A returned value is `true` if the collision system is to continue searching other collisions. The value does not indicate that there is an intersection point between the OBBs, OBB and triangle, or triangles.

Any intersection points found by `FindIntersection` when applied to OBBs or triangles are passed onto the

application via a callback mechanism that is associated with the object whose pointer is stored by the OBB node. Normal vectors are also passed to the callback. A normal for an OBB is computed as if the OBB were an ellipsoid, thus providing a smoothed normal vector field for the box. The return value of the callback is boolean and indicates whether or not the collision system should continue searching for collisions. This gives the application the opportunity to terminate the search after one or more collisions rather than processing all possible collision points.

```
bool FindIntersection (float dt, ObbTree node0, ObbTree node1)
{
    // first time, location, and normals of intersection
    float time;
    Point3 intersect, normal0, normal1;

    if ( HasObb(node0) )
    {
        if ( HasObb(node1) )
        {
            FindObbObb(dt,node0.Obb,node1.Obb,time,intersect);
            node1.Obb.GetNormal(intersect);
        }
        else
        {
            FindObbTri(dt,node0.Obb,node1.Tri,time,intersect);
            node1.Tri.GetNormal(intersect);
        }

        normal0 = node0.Obb.GetNormal(intersect);
    }
    else
    {
        if ( HasObb(node1) )
        {
            FindTriObb(dt,node0.Tri,node1.Obb,time,intersect);
            node1.Obb.GetNormal(intersect);
        }
        else
        {
            FindTriTri(dt,node0.Tri,node1.Tri,time,intersect);
            node1.Tri.GetNormal(intersect);
        }

        normal0 = node0.Tri.GetNormal(intersect);
    }
}

// provide the application with the collision information
bool bContinue0;
if ( node0.Object.Callback )
```

```
{
    bContinue0 = node0.Object.Callback(node1.Object,time,intersect,
        normal0,normal1);
}
else
{
    bContinue0 = true;
}

bool bContinue1;
if ( node1.Object.Callback )
{
    bContinue1 = node1.Object.Callback(node0.Object,time,intersect,
        normal1,normal0);
}
else
{
    bContinue1 = true;
}

return bContinue0 && bContinue1;
```

Pseudocode for finding a point of intersection is given below. The return value is `true` if and only if the collision system should continue searching for collisions.

```
bool FindCollision (float dt, ObbTree node0, int depth0, ObbTree node1, int depth1)
{
    if ( !TestIntersection(dt,node0,node1) )
        return true;

    if ( HasChildren(node0,depth0) )
    {
        if ( !FindCollision(dt,node0.Lchild,depth0-1,node1,depth1) )
            return false;
        if ( !FindCollision(dt,node0.Rchild,depth0-1,node1,depth1) )
            return false;
        if ( HasChildren(node1,depth1) )
        {
            if ( !FindCollision(dt,node0,depth0,node1.Lchild,depth1-1) )
                return false;
            if ( !FindCollision(dt,node0,depth0,node1.Rchild,depth1-1) )
                return false;
        }
        return true;
    }

    if ( HasChildren(node1,depth1) )
    {
```

```
            if ( !FindCollision(dt,node0,depth0,node1.Lchild,depth1-1) )
                return false;
            if ( !FindCollision(dt,node0,depth0,node1.Rchild,depth1-1) )
                return false;
            return true;
        }

        // At this point we know there is an intersection.  Compute the
        // intersection and make this information available to the application
        // via the object callback mechanism.

        return FindIntersection(dt,pkTree1);
    }
```

References

[1] Stefan Gottschalk, Ming Lin, and Dinesh Manocha, *OBBTree: A Hierarchical Structure for Rapid Interference Detection*, In Proceedings of ACM Siggraph, pp. 171–180, 1996.

[2] Tomas Möller, *A Fast Triangle–Triangle Intersection Test*, Journal of Graphics Tools, vol. 2, no. 2, pp. 25–30, 1997.

[3] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, England, 1988