Intersection of Objects with Linear and Angular Velocities using Oriented Bounding Boxes

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/

Created: March 2, 1999
Last Modified: March 2, 2008

Contents

1 Introduction

This document is based on the ideas in *Dynamic Collision Detection using Oriented Bounding Boxes*, another document at this web site. The latter document describes the algorithm for determining if two oriented bounding boxes are intersecting. The boxes can have constant linear velocities. The method for showing non-intersection is based on computing a separating axis for the two boxes. Such an axis has the property that the projections of the two boxes onto the axis do not intersect.

The current document shows how to extend the ideas of non-intersection to oriented bounding boxes that have both linear and angular velocities. The algorithm is first presented in a closed-form fashion. Determination of non-intersection is equivalent to showing that at least one function of time (from a set of fifteen non-negative functions) is positive on the specified time interval. The functions contain sine and cosine terms of two different frequencies. Evaluation of these is expensive unless lookup tables are used. Moreover, to show that the function is positive requires a numerical method for constructing the minimum of the function. The iterative schemes can be expensive, too. Alternatively one may select a fixed number of samples on the specified time interval and evaluate the functions and those samples. Given that the trigonometric values are handled by lookup tables, there is an additional problem in that the function values at the samples are all positive, yet the minimum function value is zero.

A better alternative is to formulate the problem in terms of differential equations. In this form no trigonometric function evaluations are required. A numerical method must be used for solving the equations, but it can be as simple as using Euler's method. The same problem exists as in the last paragraph–the numerical method effectively generates a sequence of function values that might all be positive, yet the function minimum is zero. For better accuracy and stability, a Runge-Kutta method can be used. Regardless of differential equation solver, the alternative algorithm creates a sequence of oriented bounding boxes for each initial box and uses the linear velocity algorithm to determine intersection of two boxes between two consecutive elements of the sequence.

2 Equations of Motion

The high level abstraction is to have an object that is tagged with a center point $\mathbf{C}$, an origin for a frame of reference, and a coordinate frame $\mathbf{U}_0$, $\mathbf{U}_1$, and $\mathbf{U}_2$, three mutually orthonormal vectors forming a right-handed system. The coordinate axes are represented as the three columns of a rotation matrix $R = [\mathbf{U}_0\ \mathbf{U}_1\ \mathbf{U}_2]$. The object is assigned a linear velocity $\mathbf{V}$ and an angular velocity $\mathbf{W}$. The axis of rotation has origin $\mathbf{C}$, direction $\mathbf{W}$, and the speed of rotation is $|\mathbf{W}|$.

The object is associated with an oriented bounding box tree. Each box in the OBB tree is offset from the object center and has a frame that is typically not oriented the same as the object frame. The motion of each box is determined by the motion of the object.

Coordinates of a point $\mathbf{X}$ can be measured relative to the object coordinate system as $\mathbf{X} = \mathbf{C} + R\mathbf{Y}$. The relative coordinates are represented by $\mathbf{Y}$. For time varying center $\mathbf{C}(t)$ and frame $R(t)$, the initial point $\mathbf{Y}$ follows the path

$$\mathbf{X}(t, \mathbf{Y}) = \mathbf{C}(t) + R(t)\mathbf{Y}$$

where the dependency on $\mathbf{Y}$ is emphasized by the explicit mention of $\mathbf{Y}$ in the functional form of $\mathbf{X}$. For

constant linear and angular velocities, the center and frame are

$$\begin{aligned}\mathbf{C}(t) &= \mathbf{C}_0 + t\mathbf{V} \\ R(t) &= \exp(\text{skew}\,(\mathbf{W})\,t)R_0\end{aligned}$$

where $\mathbf{C}_0$ is the initial center and R_0 is the initial frame. If $\mathbf{W} = (w_0, w_1, w_2)$, then $[S_{ij}] = \text{skew}\,(\mathbf{W})$ is the skew-symmetric matrix whose diagonal entries are $S_{00} = S_{11} = S_{22} = 0$ and whose other entries are $S_{01} = -S_{10} = w_2$, $S_{02} = -S_{20} = -w_1$, and $S_{12} = -S_{21} = w_0$. The matrix $\exp(\text{skew}\,(\mathbf{W})\,t)$ is the rotation matrix that represents the rotation whose axis is in the direction of $\mathbf{W}$ and whose angle of rotation is $|\mathbf{W}|t$.

The motion for constant linear and angular velocities is therefore

$$\mathbf{X}(t, \mathbf{Y}) = \mathbf{C}_0 + t\mathbf{V} + \exp(\text{skew}\,(\mathbf{W})\,t)R_0\mathbf{Y}$$

for $t \geq 0$. The differential equation governing the motion of the object is

$$\begin{aligned}\frac{d\mathbf{X}}{dt} &= \mathbf{V} + \text{skew}\,(\mathbf{W})\,\exp(\text{skew}\,(\mathbf{W})\,t)R_0\mathbf{Y} \\ &= \mathbf{V} + \text{skew}\,(\mathbf{W})\,(\mathbf{X} - (\mathbf{C}_0 + t\mathbf{V})) \\ &= \mathbf{V} + \mathbf{W} \times (\mathbf{X} - (\mathbf{C}_0 + t\mathbf{V}))\,.\end{aligned}$$

Since the motion is rigid, the OBBs in the OBB tree are governed by the same differential equation. However, the center and frame for an OBB can be derived from basic principles. If $\mathbf{C}_1$ is the OBB initial center and R_1 is the OBB initial frame, then in terms of the object coordinate system, $\mathbf{C}_1 = \mathbf{C}_0 + R_0\boldsymbol{\xi}$ for some relative coordinate vector $\boldsymbol{\xi}$. The time-varying path of the OBB center is

$$\begin{aligned}\mathbf{K}(t) &= \mathbf{X}(t, \boldsymbol{\xi}) \\ &= \mathbf{C}(t) + R(t)\boldsymbol{\xi} \\ &= \mathbf{C}_0 + t\mathbf{V} + [\exp(\text{skew}\,(\mathbf{W})\,t)R_0][R_0^{\mathsf{T}}(\mathbf{C}_1 - \mathbf{C}_0)] \\ &= \mathbf{C}_0 + t\mathbf{V} + \exp(\text{skew}\,(\mathbf{W})\,t)(\mathbf{C}_1 - \mathbf{C}_0)\end{aligned}$$

The time-varying OBB frame is simply the application of the object's relative rotation to the box's frame,

$$P(t) = \exp(\text{skew}\,(\mathbf{W})\,t)R_1.$$

If $\mathbf{X}(t, \mathbf{Y}) = \mathbf{K}(t) + P(t)\mathbf{Y}$, then

$$\begin{aligned}\frac{d\mathbf{X}}{dt} &= \frac{d\mathbf{K}(t)}{dt} + \frac{d\mathbf{P}(t)}{dt}\mathbf{Y} \\ &= \mathbf{V} + \text{skew}\,(\mathbf{W})\,\exp(\text{skew}\,(\mathbf{W})\,t)(\mathbf{C}_1 - \mathbf{C}_0) + \text{skew}\,(\mathbf{W})\,\exp(\text{skew}\,(\mathbf{W})\,t)R_1\mathbf{Y} \\ &= \mathbf{V} + \text{skew}\,(\mathbf{W})\,(\mathbf{X} - (\mathbf{C}_0 + t\mathbf{V})) \\ &= \mathbf{V} + \mathbf{W} \times (\mathbf{X} - (\mathbf{C}_0 + t\mathbf{V}))\,.\end{aligned}$$

This verifies that in fact the OBB is governed by the equations of motion of the object.

3 Closed-Form Algorithm

The closed-form approach is illustrated with one of the separating axis tests. The other axes are processed in a similar fashion. The notation in this section is based on that of *Dynamic Collision Detection using Oriented Bounding Boxes*.

For two stationary OBBs (one from each OBB tree for the two interacting objects), the non-intersection test based on the axis with direction $\mathbf{A}_0$ is

$$|\mathbf{A}_0 \cdot \mathbf{D}| > a_0 + |b_0 c_{00}| + |b_1 c_{01}| + |b_2 c_{02}|$$

where the c_{ij} are the entries of matrix $C = A^{\mathsf{T}} B$ with columns of A being the axes of the first box and columns of B being the axes of the second box. The values a_i and b_i are the extents of the boxes. Finally, $\mathbf{D}$ is the difference between the second and first box centers.

Let $\mathbf{K}_i(t)$ and $P_i(t)$, $i = 0, 1$ represent the time-varying centers and coordinate frames for the two boxes. Let $\mathbf{A}_0(t)$ be the first column of $P_0(t)$. The non-intersection test is now time-varying and is

$$|\mathbf{A}_0(t) \cdot (\mathbf{K}_1(t) - \mathbf{K}_0(t))| > a_0 + |b_0 c_{00}(t)| + |b_1 c_{01}(t)| + |b_2 c_{02}(t)|$$

where $C(t) = P_0(t)^{\mathsf{T}} P_1(t)$.

Each object has (possibly) different angular velocities. Generally the sinusoidal terms in $P_0(t)$ and $P_1(t)$ have different frequencies. The non-intersection test therefore contains sinusoidals of two frequencies. The centers $\mathbf{K}_i(t)$ themselves include sinusoidal terms whenever the box centers are not the same as the object centers. Verifying the inequality in the test for all t in a specified interval is not a simple problem. The verification of $\ell(t) > r(t)$ for $t \in [0, T]$ is equivalent to showing that $\delta(t) = \ell(t) - r(t)$ has a positive minimum for $t \in [0, T]$. This can be done by applying a numerical minimizer to $\delta(t)$, probably using an inverse parabolic interpolator such as Brent's method. This is expensive in that each function evaluation requires computing sinusoidal functions. The cost can be reduced by using table lookups, but the iteration itself might not always converge to an acceptable value in the same number of steps per non-intersection test. Better control on the process would be to select a fixed number of evaluations per test, say $N > 0$, and compute $\delta_i = \delta(Ti/N)$ for $0 \le i \le N$. The minimum of the δ_i is computed and, if sufficiently larger than zero (the application must select a threshold), the boxes are determined not to intersect.

4 Algorithm Based on a Numerical ODE Solver

Rather than evaluating the δ_i per potential separating axis, a better approach is to numerically solve the equations of motion for $t \in [0, T]$. For the first box, solve

$$\frac{d\mathbf{X}}{dt} = \mathbf{V} + \mathbf{W} \times (\mathbf{X} - (\mathbf{C}_0 + t\mathbf{V}))$$

for $t > 0$ with initial condition $\mathbf{X}(0) = \mathbf{K}_0$. The simplest approach is to use Euler's method and iterate N times on the given interval. The time steps are $t_i = Ti/N$ for $0 \le i \le N$. Define $\mathbf{X}_i$ to be the numerical approximation to $\mathbf{X}(t_i)$; then $\mathbf{X}_0 = \mathbf{K}_0$ and

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \frac{T}{N} [\mathbf{V} + \mathbf{W} \times (\mathbf{X}_i - \mathbf{C}_0 + t_i \mathbf{V})]$$

where the first object has center $\mathbf{C}_0$, linear velocity $\mathbf{V}$, and angular velocity $\mathbf{W}$. The iteration scheme is evaluated for $0 \le i < N$. This produces a sequence of centers for the first box, $\mathbf{X}_0$ through $\mathbf{X}_N$. The coordinate frame is also integrated using

$$\frac{dP}{dt} = \text{skew}\,(\mathbf{W})\,P$$

for $t > 0$ with initial condition $P(0) = R_1$. Using Euler's method, the values P_i are approximations to $P(t_i)$ and the iterations are

$$P_{i+1} = P_i + \frac{T}{N}\text{skew}\,(\mathbf{W})\,P_i = \left(I + \frac{T}{N}\text{skew}\,(\mathbf{W})\right)P_i.$$

The numerical scheme does not preserve the orthonormality of the matrix. That is, $P_0 = R_1$ is orthonormal, but P_1 is not necessarily orthonormal. For very small T/N, P_1 is nearly orthonormal and can be used as is. However, it might be desirable to renormalize P_i at each step. This can be done by using Gram-Schmidt orthonormalization on the three columns of P_i.

After the center and frame iterates have been generated, the sequence of centers and frames for the first moving OBB are $\mathbf{X}_i$ and P_i. The second moving OBB has similar sequences $\mathbf{Y}_i$ and Q_i. For each i, the linear velocity of the first OBB over the corresponding time subinterval is set to $\mathbf{V}_0 = \mathbf{X}_{i+1} - \mathbf{X}_i$ and that of the second OBB is set to $\mathbf{V}_1 = \mathbf{Y}_{i+1} = \mathbf{Y}_i$. On the time subinterval, the two boxes are compared as if they have only linear velocities. The first OBB has center $\mathbf{X}_i$, coordinate frame P_i, and velocity $\mathbf{V}_0$. The second OBB has center $\mathbf{Y}_i$, coordinate frame Q_i, and velocity $\mathbf{V}_1$. The algorithm (and code) mentioned in *Dynamic Collision Detection using Oriented Bounding Boxes* is applied each of the N pairs of OBBs. For a given potential separating axis, if the N tests all show non-intersection, then the axis is separating and the two OBBS do not intersect on the given time interval.

A more accurate but more expensive method such as Runge-Kutta could be used to generate the iterates for the centers and frames.

5 What is Next

Just a note to others (and myself). When I get enough time and motivation, I will modify the OBB tree collision system at my web site to include angular velocity and an executable that illustrates how it works. This is a major project.