Efficient Continuous Collision Detection for Bounding Boxes under Rational Motion

Dan Albocher*, Uzi Sarel*, Yi-King Choi†, Gershon Elber*, Wenping Wang†
*Department of Computer Science, Technion, IIT, Haifa 32000, Israel
{sdannya,sareluzi}@t2.technion.ac.il, gershon@cs.technion.ac.il
†Department of Computer Science, The University of Hong Kong, Hong Kong, P.R. China
{ykchoi,wenping}@cs.hku.hk

***Abstract*— This paper presents a simple yet precise and efficient algorithm for collision prediction of two oriented bounding boxes under univariate (piecewise) rational motion. We present an analytic solution to the problem of finding the time of collision and the feature involved, or declaring that no collision should occur. Our solution can be applied to boxes of any size, under arbitrary rational rigid motion. The algorithm is based on the efficient examination of the Minkowski sum (MS) of the two boxes, using a spherical Gauss map dual representation, and a precise extraction of the collision time, if any, as a solution to a set of rational equations that are automatically derived.**

I. Introduction

Collision detection (CD) is a fundamental problem in a wide range of fields. Various CD applications are found in robotics (motion planning), computer graphics (physically based simulations), and 3D computer games. Collision detection between complex 3D models is considered a difficult task to perform directly. A common simplification of the general collision prediction problem computes the collision between simple bounding volumes, e.g., bounding spheres [13], [14], [19], axis aligned bounding boxes (AABB) [2], [12], oriented bounding boxes (OBB) [11], and discrete oriented polytopes (K-DOPS) [15]. The efficiency of the bounding volume approach has been demonstrated in [23], which stated that as long as convex bound volumes are used and a few reasonable assumptions are made regarding the relation between the original shape and its bounding volume, the number of collisions predicted is proportional to the number of actual collisions.

Many solutions for the CD problem discretely sample the objects' locations along the motion path and perform local collision detection per location [1], [11], [15], [24]. Due to their discrete nature, these algorithms perform poorly with small objects or fast motion. To overcome the limitations of discrete collision detection methods, several techniques have been proposed to model the motion between samples, by interpolating a continuous path between two or more samples and solving the continuous motion equations. A common approach for solving these continuous collision detection (CCD) problems is to calculate the swept volume of the moving objects along their paths, and perform collision tests between the swept volumes. Finding the collision time is not an inherent part of these methods, and therefore, their ability to predict the collision is limited. Other known CCD algorithms that are purely analytic are limited to specific types of motion, such as a screw motion, or suffer from very high complexity.

In this work, we present an algorithm that provides a simple analytic solution to the CCD problem of two oriented bounding boxes (OBBs) under rational rigid motion of any degree, by analyzing Minkowski sums (MS). The MS [26] of two sets, B_1 and B_2 in vector space, denoted by $B_1 \oplus B_2$, is the set $\{b_1 + b_2 \mid b_1 \in B_1, b_2 \in B_2\}$. Two objects, B_1 and B_2, intersect if there exist $b_1 \in B_1, b_2 \in B_2$ such that $b_1 = b_2$. Therefore, the intersection between objects B_1 and B_2, if any, can be found by identifying the time when the MS of B_1 and $-B_2$ contains the origin, having $b_1 \in B_1, b_2 \in B_2$ such that $b_1 - b_2 = 0$. We present a method for efficiently computing the precise time along the rational motion when the origin is contained in the boundary of the Minkowski sum of two moving OBBs, and consequently, for analytically predicting the time of collision. The CCD problem is reduced to a set of polynomial equations, with its degree bounded by a function of the degree of the rational motion.

The rest of this paper is organized as follows. In Section II, earlier relevant work on CD is reviewed. Section III gives an overview of our algorithm and some background. In Section IV, we present our algorithm in detail. Section V provides some results and examples. Finally, in Section VI, we discuss some limitations and conclude our approach.

II. Related Work

Since the early work on CD [3], [5], [7], the field has grown significantly and addressed various types of collision queries, differing in object types (e.g. polytopes, surfaces) and motion types (e.g. linear, screw, rational). Quite a few different approaches have been taken to determine whether two moving objects will collide. For an extensive survey see [17]. Here, only relevant work, on precise CCD under a univariate motion, is discussed.

Collision detection of interpolated motion is useful in motion planning and also as an improvement to the discrete collision detection algorithms. Canny [7] used quaternion-based parameterization of the objects' motion and collision constraints, and then extracted the time of collision by solving a set of polynomial equations of low degrees; however, his algorithm's high complexity makes it unwieldy and difficult to implement. Redon, Kheddar and Coquillart [20], Buss [4], Rossignac and Kim [22] parameterized the objects' motion with glide rotation (e.g., screw motion). This approach handles only limited motions, and approximating a general motion using screw motion is, therefore, not always completely accurate. Another continuous method, suggested by Cameron [6], transforms the collision detection problem to a space-time domain, and performs a 4D intersection test between 4D extrusions of the 3D objects through time (sweep). Nevertheless, due to the computational cost of generating the 4D extrusion, the objects are limited to translational motions only.

A more generic approach suggested by Redon, Kheddar and Coquillart [21] makes use of OBB hierarchies, and expands the discrete overlap test for OBBs described by Gottschalk et al. [11] to the continuous case, using interval arithmetics [18]. This method eliminates the need to extract a closed-form contact time from the interpolated motion. The method, however, is not simple to implement and is based on interval arithmetics and recursive subdivision of these intervals. It, therefore, depends on a subdivision threshold and cannot claim to be analytically precise.

CCD has also been studied for moving ellipses [9] and ellipsoids [8] under rational motion. Here the CCD problem is reduced to finding the zeros of a univariate or a system of bivariate equations, respectively.

The suggested algorithm requires computation of MS of two OBBs. Efficient algorithms for computing the MS of general polyhedral models exist (e.g., [10], [25]), but these do not take advantage of our simple case, where the two polyhedral objects are OBBs. Herein, the MS is defined by using the proper subset of all possible pairs of elements from both OBBs. This pair-matching operation is similar to the one used for approximating the MS of curves in the 2D case by Lee, Kim and Elber [16], and for polytopes by Fogel and Halperin [10]. There, the explicit equation of each face is extracted, and a test is performed when, if at all, the face contains the origin.

III. Overview

This work focuses on a fundamental step of many CCD algorithms. We strive to predict the collision time between two OBBs, under univariate rational motion. Denote by $B_1(t)$ and $B_2(t)$, the sets of points defining the boundaries of the two moving OBBs, where t is a time parameter s.t. $t \in [0,1]$. The two OBBs will collide if and only if there exists t_0 where the MS of $B_1(t_0)$ and $-B_2(t_0)$ contains the origin, $\vec{0} \in (B_1(t_0) \oplus (-B_2(t_0)))$. This t_0 will be found analytically.

In order to detect a collision, only times when one OBB touches the other are required, without considering penetrations. Since a continuous motion is analyzed, when $B_1(t)$ touches $B_2(t)$ for the first time, the origin will become part of the MS for the first time. Hence, only the boundary of the Minkowski sum (BMS) is of interest, and all the internal elements can be ignored.

A face on the BMS of two polytopes must be the sum of two elements (vertices, edges, faces), taken from the polytopes. However, not all these element pairs will constitute a face on the boundary; some of them may only form an internal element. In fact, the faces on the MS must be formed by pairs of elements that have matching normal vector ranges, which we find using the spherical Gauss map (SGM) of the OBBs' described in Section III-A. If the two polytopes touch each other, the face on the Minkowski sum formed by the touching elements must contain the origin.

The main steps of the presented algorithm are:

1) Extract the SGMs of $B_1(t)$ and $B_2(t)$. Every face, edge and vertex in each OBB has its matching point, arc and spherical octant [1] on the normal (Gaussian) sphere, respectively, according to their orientation (described in detail, in Section III-A);
2) Find the compatible pairs of components from each OBB by comparing the two SGMs (described in detail in Section IV-B);
3) Calculate the planes in which the BMS faces are contained using the compatible element pairs of the OBBs. Since the topology of the BMS is derived from the set of compatible elements, a change in these pairs accounts to a change in the topology of the BMS. The set of compatible pairs will not change as long as no spherical point in the SGMs switches octants. Therefore, the time line is divided into homogeneous intervals, in each of which no point moves between octants on the SGM of $B_1(t)$ and $B_2(t)$ (described in detail in Section IV-C);
4) Each homogeneous interval defines a unique MS's topology, and thus a unique set of planes that contain the faces of the BMS. The MS of two convex polytopes is convex [10]. Consequently, if there exists a t_0 s.t. one of the planes of the BMS contains the origin and further, the origin is on the inside half-space of all other planes, the origin is on the BMS due to the convexity of the BMS of two OBBs, which indicates a collision (described in detail in Section IV-D).

A. The Spherical Gauss Map's (SGM) Dual Representation

In order to efficiently find the matching pairs among OBB elements (faces, edges and vertices), an SGM's dual representation is created for each OBB. Initially, the OBB is assumed to be aligned with the axes, so it can be described by its two extreme dimensions: $(MinX, MinY, MinZ)$ and $(MaxX, MaxY, MaxZ)$, denoted $(\underline{X}, \underline{Y}, \underline{Z})$ and $(\overline{X}, \overline{Y}, \overline{Z})$, respectively. Each face f is defined by its containing plane (e.g., $\underline{X}$ for the face contained in $x = MinX$ plane, etc.). The eight vertices of the OBB are then defined as the intersection of three faces $\{(f_0, f_1, f_2) \,|\, f_0 \in \{\underline{X}, \overline{X}\}, f_1 \in \{\underline{Y}, \overline{Y}\}, f_2 \in \{\underline{Z}, \overline{Z}\}\}$. Finally, denote an OBB edge e as the intersection of its two adjacent faces (e.g., $(\underline{X}, \overline{Y})$ etc.).

Define $\mathcal{A}_{\mathcal{P}}(f)$ to be the opposite (antipodal) face to f relative to the box's center. For example, given $f = \underline{X}$, $\mathcal{A}_{\mathcal{P}}(f) = \overline{X}$. Similarly, define $\mathcal{A}_{\mathcal{P}}(v) = (\mathcal{A}_{\mathcal{P}}(f_0), \mathcal{A}_{\mathcal{P}}(f_1), \mathcal{A}_{\mathcal{P}}(f_2))$ of vertex v as the opposite vertex of $v = (f_0, f_1, f_2)$ and

[1] Note we use the terms face, edge and vertex to denote geometric elements in the primal, Euclidean, space, and the terms point, arc and octant to denote geometric elements in the dual, SGM, space.

$\mathcal{A}_{\mathcal{P}}(e) = (\mathcal{A}_{\mathcal{P}}(f_0), \mathcal{A}_{\mathcal{P}}(f_1))$ of edge e as the opposite edge of $e = (f_0, f_1)$. Moreover, for any element E of the box (a face, an edge, or a vertex), $\mathcal{A}_{\mathcal{P}}(\mathcal{A}_{\mathcal{P}}(E)) = E$.

Each face in the original OBB is represented as a point on the SGM. Each of the OBB's edges is represented as a great, 90 degree arc on the SGM, with normals spanned by the affine combination of its two adjacent faces' normals. Finally, each of the OBB's vertices is represented as an octant on the SGM, with normals that are the affine combination of its three adjacent faces' normals. The SGM of an OBB, as shown in Fig. 1, has six spherical points (OBB faces), 12 spherical arcs (edges on the OBB), and eight sphere octants (vertices of the OBB). The OBB notation will be used to describe the equivalent elements in the SGM. As in the OBB case, $\mathcal{A}_{\mathcal{P}}(E)$, $E \in SGM$ will be used to denote the antipodal element to E in the SGM.

Boundary faces on the MS are created by summing either a face from one OBB with a vertex from the other or an edge from one OBB with an edge from the other. Herein, only pairs with compatible or matched orientation (when one's SGM normal range intersects with the other's SGM normal range) form a valid BMS face (see Fig. 2). Taking into account all face-vertex and edge-edge combinations for eight vertices, six faces, and 12 edges per OBB yield a total of 240 possible pairs. By comparing the SGMs of each OBB, and taking only pairs with compatible orientation, the number of pairs that will be processed in constructing the BMS will be greatly reduced to only 30. This pairing procedure will be discussed in detail in Section IV.

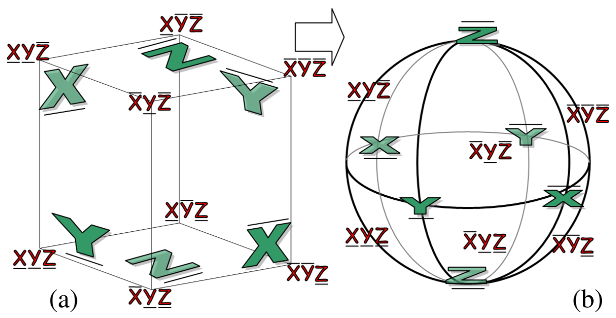


Fig. 1. An OBB with its vertex, edge and face notation (a), and the matching (dual) SGM with its octant, arc and point notation (b).

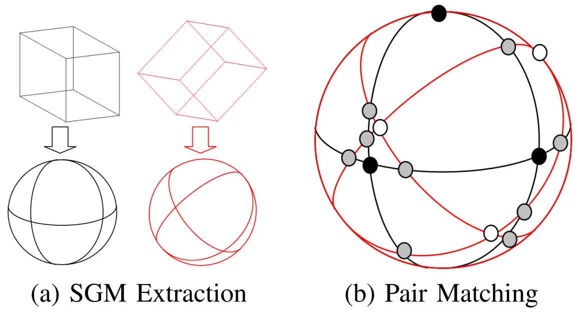


Fig. 2. Finding the pairs defining the MS. First, extract the SGM for each OBB (a). Then find pairs with compatible orientations (b). In white are the vertex-face matches, in gray edge-edge matches and in black, face-vertex matches.

IV. The Collision Prediction Algorithm

Denote the rigid motion transformation matrices of boxes $B_1(t)$ and $B_2(t)$ by $M_1(t)$ and $M_2(t)$, respectively. Both matrices include rigid motion and non-uniform scale. Denote the inverse motion matrix of $B_i(t)$ by $M_i^{-1}(t)$. Therefore, the relative motion of $B_2(t)$ in $B_1(t)$'s coordinate system is $M(t) := M_2(t) M_1^{-1}(t)$ [2]. Hence, and without loss of generality, we hereafter consider $B_1 = B_1(t)$ as static, with its center at the origin and edges aligned with the axes. We may now define the relative motion of $B_2(t)$, $\tilde{B}_2(t)$, as $M(t)$.

A. Dividing the Motion Path into Homogeneous Intervals

In order to examine the topology of the BMS and find the time(s) when it includes the origin, the path of $\tilde{B}_2(t)$ is divided into intervals that contain homogeneous BMS topologies. The times when these topologies change are found as the zeros of all the normal components of $\tilde{B}_2(t)$. These zeros are sufficient conditions for homogeneity, since the BMS is composed of point-octant and arc-arc pairs, one from B_1, which is now static and axis aligned, and one from $\tilde{B}_2(t)$. As long as none of the components of the normals of $\tilde{B}_2(t)$ switch sign, all its normals remain in the same octant of B_1 and thus the topology of the BMS remains consistent. The three normals of $\tilde{B}_2(t)$ are computed by applying the transformation $M(t)$ to the canonical normals $\overrightarrow{N_0^x} = (1,0,0,0)$, $\overrightarrow{N_0^y} = (0,1,0,0)$ and $\overrightarrow{N_0^z} = (0,0,1,0)$,

$$\begin{aligned}
\overrightarrow{N^x}(t) &:= \overrightarrow{N_0^x} M(t) = \left(N_x^x(t), N_y^x(t), N_z^x(t)\right), \\
\overrightarrow{N^y}(t) &:= \overrightarrow{N_0^y} M(t) = \left(N_x^y(t), N_y^y(t), N_z^y(t)\right), \qquad (1) \\
\overrightarrow{N^z}(t) &:= \overrightarrow{N_0^z} M(t) = \left(N_x^z(t), N_y^z(t), N_z^z(t)\right).
\end{aligned}$$

Consider the sorted list of the times the BMS's topology changes, $T = \left\{t_i \mid \exists N_a^b(t_i) = 0, a, b \in \{x, y, z\}\right\}$, as time(s) when one of the components of the transformed normals of $\tilde{B}_2(t)$ switches an octant in the SGM.

B. Efficient Computation of the BMS

Assume that the topology of the BMS was found to change $|T| = m$, or m times, $t_i, i = 1..m$. Further, denote by I_i the homogeneous time interval $[t_{i-1}, t_i)$, and let $t_0 = -\infty$ and $t_{m+1} = \infty$. For each interval I_i, $1 \le i \le m+1$, we find the compatible components of the two boxes using their SGMs. Let $\tilde{t}_i \in I_i$ be an arbitrary interior time in I_i. Then, the vertex-face, face-vertex, and edge-edge pairs are found with the aid of the following property:

Lemma 1: For two OBBs in general position, exactly six face-vertex (FV) instances, six vertex-face (VF) instances and 18 edge-edge (EE) instances contribute and define the BMS at each interior time $\tilde{t}_i$.

Proof: Consider all compatible pairs. Each of the six points in each SGM resides in one octant on the other SGM. Therefore, the point and octant together comprise a compatible

[2] We follow the convention for post-order matrix multiplications. The composition of a transformation M_1 followed by M_2 is, therefore, given by $M = M_1 M_2$.

pair (giving us a total of six VF and six FV pairs). There are no other VF or FV pairs, because each point can only reside in one octant. There are three great circles on each SGM. Each great circle from B_1's SGM intersects each of the great circles of B_2 exactly twice. Altogether, there are nine possible pairs of great circles, each defining two intersections between two arcs (a total of 18 EE pairs). ■

Denote by o an octant in the SGM, $o \in \left\{(x,y,z)\,|x \in \left\{\underline{X},\overline{X}\right\}, y \in \left\{\underline{Y},\overline{Y}\right\}, z \in \left\{\underline{Z},\overline{Z}\right\}\right\}$ (recall Fig. 1). Further, denote by p a point and a an arc in an SGM. Elements $p_j^i, a_j^i, o_j^i,\ i \in \{1,2\}, j \in \{1,2,3\}$ correspond to the j^{th} point, arc, or octant of B_i's SGM. The OBB B_1 is static, aligned with the axes and has its center at the origin. Therefore, its SGM octants correspond to its coordinate system octants. Let $o2p$ be the function finding a point shared by three neighboring octants. As described above, this is done simply by finding the common point in the definition of these three neighboring octants. For example, as seen in Fig. 1, the octants $\left(\underline{X},\underline{Y},\overline{Z}\right)$, $\left(\underline{X},\underline{Y},\underline{Z}\right)$ and $\left(\overline{X},\underline{Y},\underline{Z}\right)$ define the point $\underline{Y}$. Similarly, function $p2o$ finds the octant on the SGM that is defined by its three neighboring points.

1) Computing VF, FV pairs. The three orthonormal normals $\overrightarrow{N^x}(t), \overrightarrow{N^y}(t), \overrightarrow{N^z}(t)$ of B_2 can be obtained, as shown in (1), and they correspond to p_1^2, p_2^2, p_3^2, respectively, in B_2's SGM (as described in III-A). Determining the sign of each component of $\overrightarrow{N^x}(\tilde{t}), \overrightarrow{N^y}(\tilde{t}), \overrightarrow{N^z}(\tilde{t})$ provides us with three VF pairs, one pair for each point. For example, if $N_x^x > 0, N_y^x > 0, N_z^x > 0$, then point p_1^2 corresponds to the octant $o_1^1 = \left(\overline{X},\overline{Y},\overline{Z}\right)$, and as explained above, the point and octant together comprise the pair $\left(o_1^1, p_1^2\right)$. Denote the three VF pairs as $\left(o_j^1, p_j^2\right), j \in \{1,2,3\}$. The three remaining VF pairs are the antipodal pairs $\left(\mathcal{A}_{\mathcal{P}}\left(o_j^1\right), \mathcal{A}_{\mathcal{P}}\left(p_j^2\right)\right), j \in \{1,2,3\}$, giving us a total of six VF pairs.
From every three points of the SGM of B_2 that form an octant, we conclude which of the SGM points of B_1 reside in it using the VF instances found above. For example, for the three pairs $\left\{\left(o_1^1,p_1^2\right),\left(o_2^1,p_2^2\right),\left(o_3^1,p_3^2\right)\right\}$, the prescribed FV instance is:
$$\left(p_1^1, o_1^2\right) = \left(o2p\left(o_1^1,o_2^1,o_3^1\right), p2o\left(p_1^2,p_2^2,p_3^2\right)\right). \quad (2)$$
Note that after finding the first three VF pairs, there is no need to solve or evaluate any equations, and calculating the rest of the VF and FV pairs can be done symbolically using the notations defined above.
2) Computing EE pairs. Since there are eight octants and six points, and geometric constraints require that an octant contains no more than one point, there are exactly two antipodal point-free octants on each B_1's and B_2's SGMs. Consider the point-free octants of B_1 and denote them o_f^1 and $\mathcal{A}_{\mathcal{P}}\left(o_f^1\right)$. The point-free octants will be analyzed shortly to extract the edge-edge (EE) pairs.
In the first step of calculating EE pairs, simple intersections are considered, where each arc intersects exactly one other arc. For each arc in B_1, the arc it intersects in B_2's SGM is found as follows (see Fig. 3 (a)): For each arc a_1^1 in B_1 defined by the points p_1^1 and p_2^1, find the two adjacent octants o_1^1, o_2^1. If both octants contain a point of B_2 (found using the VF pairs), these two points (p_1^2, p_2^2) must be adjacent, and are, therefore, connected to an arc a_1^2 of B_2. Arc a_1^2 necessarily intersects a_1^1, because its edge points, p_1^2 and p_2^2, are on two different sides of a^1. This generates the pair (a_1^1, a_1^2). Since there are two antipodal octants that are point-free, the remaining octants form a ring with six internal arcs whose neighbouring octants must not be point-free. As a result, this step generates a total of six pairs.
The arcs of the point-free octant o_f^1 are then handled differently. For the point-free octant o_f^1, we find the octant's three border arcs a_1^1, a_2^1 and a_3^1 and the matching adjacent octants o_1^1, o_2^1 and o_3^1 (see Fig. 3 (b)). We then find the B_2 points, p_1^2, p_2^2 and p_3^2, in each of these octants (there must be a point in each, since the other point-free octant is antipodal, and not adjacent to o_f^1). Now, six new EE pairs are derived, which are defined by: $\left(a_i^1, p_i^2 p_j^2\right), i,j \in \{1,2,3\}, i \neq j$. Six additional arc-arc pairs come from the antipodal point-free octant $\mathcal{A}_{\mathcal{P}}\left(o_f^1\right)$, which gives us a total of twelve pairs generated by this step.

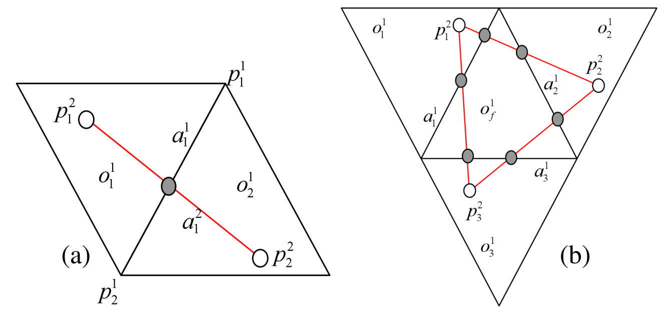


Fig. 3. EE Pairs: (a) shows the single intersection arcs. (b) shows the double intersection arcs.

C. *Calculating the BMS Planes*

With the six VF, six FV, and the 18 EE pairs, the 30 corresponding BMS plane equations are constructed. The followings are the explicit equations of these 30 planes.

1) Face-Vertex(t) (Point-Octant(t)). Let the face's plane equation be $Ax + By + Cz + D = 0$, where $A^2 + B^2 + C^2 = 1$, and let $v(t) = (x(t), y(t), z(t))$ denote the vertex. Then, the BMS plane of this pair is:
$$Ax + By + Cz + D + \langle (A,B,C), v(t)\rangle = 0. \quad (3)$$
2) Face(t)-Vertex (Point(t)-Octant). This is a symmetric case - the vertex is now stationary and the plane equation is a function of t. The BMS plane of this pair is:
$$A(t)x + B(t)y + C(t)z + D(t) + \langle (A(t), B(t), C(t)), v\rangle = 0. \quad (4)$$
3) Edge - Edge(t) (Arc - Arc(t)). Let $e^1 = \left(v_1^1, v_2^1\right)$ denote the static edge in B_1, and $e^2(t) = \left(v_1^2(t), v_2^2(t)\right)$ denote the moving edge. The matching BMS plane is:

$$\left\langle e_1^1 \times e_1^2(t), (x,y,z)\right\rangle + \left\langle e_1^1 \times e_1^2(t), v_1^1 + v_1^2(t)\right\rangle = 0. \quad (5)$$

D. Finding the Collision Time

The last step of the algorithm finds the time $\tilde{t}$ for which $\vec{0} \in BMS\left(\tilde{t}\right)$. For each of the 30 plane equations, given by $P_i(t) = \left\langle \overrightarrow{N_i}(t), (x,y,z)\right\rangle - D_i(t) = 0,\ 1 \le i \le 30$, there is $\vec{0} \in P_i\left(\tilde{t}\right)$ iff $D_i\left(\tilde{t}\right) = 0$. Further, all BMS planes are constructed so that they face outside (i.e., $D_i(t) > 0$). From the convexity of the BMS (as a MS of two convex shapes), we have

$$\vec{0} \in BMS(\tilde{t}) \Leftrightarrow \exists i,\ D_i(\tilde{t}) = 0\ \wedge\ \forall j \neq i,\ D_j(\tilde{t}) \le 0.$$

E. Optimization

A simple and effective bounding sphere (BS) test over the OBBs is used to eliminate the intervals along the relative motion where the OBBs are too far to collide. The BS test is performed by finding all intervals along the motion when the center of the moving OBB, $\tilde{B}_2$, is located at a distance of at least r from the origin (which is the center of B_1), where r is the sum of the radii of the two bounding spheres.

V. Results

The presented algorithm was implemented in C++ and the tests below were executed on a Pentium 4 with 1 GB RAM.

As stated in Section IV, addressing the case in which both boxes are moving is unnecessary, because one can always change the coordinate system to be that of one of the boxes, and thus one box is static and the other moving. Sections V-A to V-C, present linear, 2nd and 4th degree motions, respectively. The following examples are also presented in details in http://www.cs.technion.ac.il/~gershon/OBBCol/.

A. Linear Motion

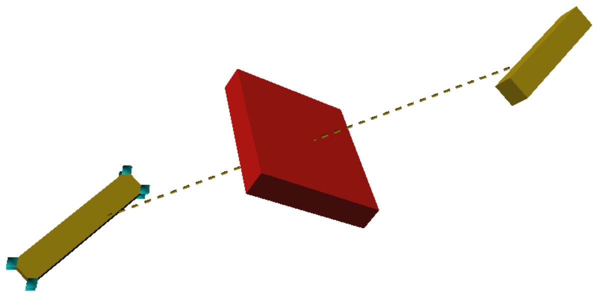
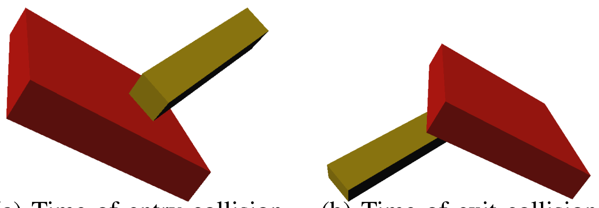


Fig. 4. Linear Motion Path



(a) Time of entry collision (b) Time of exit collision

Fig. 5. Linear Motion Collisions - (a) The time of entry collision between the two boxes. (b) The time of exit collision between the boxes, if they were to continue their initial motion, as if no physical collision occurred.

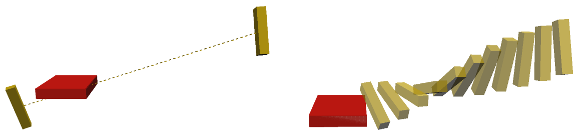
Fig. 4 depicts the motion path and Fig. 5 the collision points of the linear motion example. This linear motion path is given by the motion matrix:

$$M(t) = \begin{pmatrix} -0.88 & -0.16 & -0.46 & 0 \\ 0.07 & -0.90 & -0.43 & 0 \\ 0.48 & 0.41 & -0.78 & 0 \\ -28.35 + 45.11t & -36.86 + 60.69t & 0 & 1 \end{pmatrix} \quad (6)$$

where the 3×3 upper left component is a fixed rotation, and the lower 1×3 component defines the linear translation. This motion has only one homogeneous interval – i.e., the BMS topology does not change throughout the entire motion.

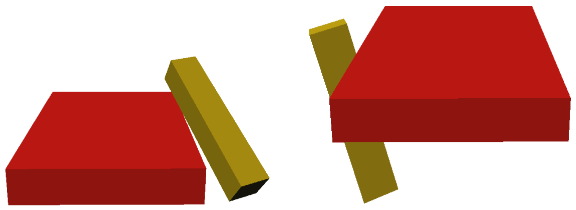
In this example, two FV pairs produce a collision – one for the entry collision (see Fig. 5(a)) and the other for the exit collision (see Fig. 5(b)). The derived polynomial, after substituting $(0,0,0)$ for (x,y,z) in (3), is $p(t) := 28.68 - 60.68t$, or $t = 0.47$. It took 0.0128 seconds to execute the entire algorithm, in this example.

B. 2nd Degree Motion



(a) Motion path (b) Motion samples

Fig. 6. 2nd Degree Motion Path - (a) The motion planned path, and initial and end positions; (b) Samples of the box's motion.



(a) Time of entry (b) Time of exit

Fig. 7. 2nd Degree Motion Collisions - (a) The time of entry collision between the two boxes. (b) The time of exit collision between the boxes.

An edge-edge collision example in presented in Fig. 6 (the motion path), and Fig. 7 (collisions' times). To produce this motion, rotation components that consist of 2nd degree rationals were used. The translation component is linear.

In this example, dividing the motion path to homogeneous intervals produced 15 intervals, of which three intervals were out of our $[0,1]$ domain, and eight intervals could be eliminated using the bounding sphere test. This leaves four relevant intervals, of which only one is of interest – between $t_{start} = 0.55$ and $t_{end} = 0.92$. In this interval, two EE pairs are found, which provide two collision times – one for the entry collision time (see Fig. 7(a)), and the other for the exit collision time (see Fig. 7(b)). The solutions were found ($t_{enter} = 0.654$, and $t_{exit} = 0.889$) from a degree 5 constraint, which is derived from (4) by substituting

$$D(t) = -\left\langle (A(t), B(t), C(t)), p(t)\right\rangle \quad (7)$$

where $p(t)$ is some point on the discussed plane. Recalling, that $A(t), B(t), C(t), p(t)$ are all generated by applying the motion matrix on static vectors representing the corresponding face/vertex, and recalling that this motion is consisted of a 2nd degree rational motion, and a linear translation - $A(t), B(t), C(t)$ become 2nd degree polynomials, and $p(t)$ becomes a 3rd degree polynomial. Therefore the dot product in (4) yields a degree-5 constraint. It took 0.052 seconds to execute the entire algorithm, in this example.

C. 4th Degree Motion

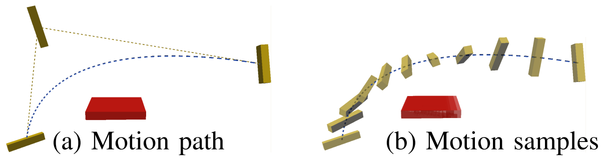


Fig. 8. 4th Degree Motion Path - (a) The motion planned path, and initial and end positions; (b) Samples of the box's motion.

The last example is without collision and is presented in Fig. 8 (a) (the motion path), and Fig. 8 (b) (samples of the motion). In this example, dividing the motion path to homogeneous intervals produced 17 intervals, of which two intervals were out of the $[0, 1]$ domain, and 11 intervals could be eliminated using the bounding sphere test. In the remaining intervals no collision was found. It took 0.103 seconds to execute the entire algorithm in the above example.

VI. Conclusion and Future Work

The proposed CD algorithm provides an efficient way to analytically calculate collisions under univariate rational rigid motion. Since continuous motions can be approximated arbitrarily precisely using rationals, this algorithm provides a practical way to perform a purely analytic CCD. Further, any complex motions could be arbitrarily precisely approximated using piecewise polynomial functions, avoiding higher degree representations.

We may further consider OBBs moving under affine motions, which can be useful for collision detection of deformed objects. However, it will take more work in the pair-matching stage as the SGMs of the OBBs are no longer regular.

The presented algorithm can be used for more complex objects than boxes, by approximating relevant parts of the objects with several OBBs, or, as suggested in Gottschalk et al [11], by using a hierarchy of OBBs. The basic calculation of OBB-OBB collision would remain the same – purely analytic.

Another possible extension to this algorithm would be to apply the SGM concept for discrete oriented polytopes (K-DOPS) or any convex polytopes. With some modification to the pair matching phase of our algorithm, it could provide a much better approximation of the bounded objects, and consequently, could reduce the overall collision computation time in cases where the motion is a-priori known and is (piecewise) rational.

References

[1] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 19–28, New York, NY, USA, 1990. ACM Press.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 322–331, New York, NY, USA, 1990. ACM Press.

[3] John W. Boyse. Interference detection among solids and surfaces. *Commun. ACM*, 22(1):3–9, 1979.

[4] Samuel R. Buss. Collision detection with relative screw motion. *The Visual Computer*, 21(1-2):41–58, 2005.

[5] S. Cameron. A study of the clash detection problem in robotics. In *Int. Conf. Robotics & Automation*, pages 488–493, St. Louis, March 1985.

[6] S. Cameron. Collision detection by four–dimensional intersectin testing. *IEEE Transaction on Robotics and Automation*, 6(3):291–302, 1990.

[7] John Canny. Collision detection for moving polyhedra. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(2):200–209, 1986.

[8] Yi-King Choi, Wenping Wang, and Myung-Soo Kim. Exact collision detection of two moving ellipsoids under rational motions. In *ICRA*, pages 349–354. IEEE, 2003.

[9] Yi-King Choi, Wenping Wang, Yang Liu, and Myung-Soo Kim. Continuous collision detection for elliptic disks. *To appear in IEEE Transactions on Robotics.*

[10] E. Fogel and D. Halperin. Exact and efficient construction of minkowski sums of convex polyhedra with applications, 2005. Manuscript, Tel Aviv University. http://www.cs.tau.ac.il/∼efif/CD/exact_mink_3d.pdf.

[11] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, New York, NY, USA, 1996. ACM Press.

[12] Martin Held, James T. Klosowski, and Joseph S. B. Mitchell. Collision detection for fly-throughs in virtual environments. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry*, pages 513–514, New York, NY, USA, 1996. ACM Press.

[13] Philip M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

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

[15] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[16] In-Kwon Lee, Myung-Soo Kim, and Gershon Elber. Polynomial/rational approximation of minkowski sum boundary curves. *Graph. Models Image Process.*, 60(2):136–165, 1998.

[17] M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

[18] Ramon E. Moore. *Methods and Applications of lnterval Analysis*. Society for Industrial and Applied Mathematics (SIAM), 1979.

[19] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees., 1999. In 15th Spring Conference on Computer Graphics, pages 83–92. Budmerice, Slovakia, April 1999. ISBN 80-223-1357-2.

[20] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. An algebraic solution to the problem of collision detection for rigid polyhedral objects. In *ICRA*, pages 3733–3738. IEEE, 2000.

[21] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Fast continuous collision detection between rigid bodies. *Comput. Graph. Forum*, 21(3), 2002.

[22] J. J. Rossignac and J. J. Kim. Computing and visualizing pose-interpolating 3d motions. *Computer Aided Design*, 33:279–291, 2001.

[23] Subhash Suri, Philip M. Hubbard, and John F. Hughes. Analyzing bounding boxes for object intersection. *ACM Trans. Graph.*, 18(3):257–277, 1999.

[24] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *J. Graph. Tools*, 2(4):1–13, 1997.

[25] Gokul Varadhan and Dinesh Manocha. Accurate minkowski sum approximation of polyhedral models. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 392–401, Seoul, Korea, October 06 - 08 2004. IEEE Computer Society.

[26] Eric W. Weisstein. Minkowski sum. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/MinkowskiSum.html.