Seventh International Conference on Wireless & Mobile Network (WiMo 2015) Sydney, Australia, (Eds : David C. Wyld, Natarajan Meghanathan, ISBN: 978-1-921987-38-0)Vol 5 (11), May 2015

A Triangle-Triangle Intersection Algorithm

Chaman L. Sabharwal and Jennifer L. Leopold

Missouri University of Science and Technology
Rolla, Missouri, USA – 65409
{chaman, leopoldj}@mst.edu

ABSTRACT

The intersection between 3D objects plays a prominent role in spatial reasoning, geometric modeling and computer vision. Detection of possible intersection between objects can be based on the objects' triangulated boundaries, leading to computing triangle-triangle intersection. Traditionally there are separate algorithms for cross intersection and coplanar intersection. There is no single algorithm that can intersect both types of triangles without resorting to special cases. Herein we present a complete design and implementation of a single algorithm independent of the type of intersection. Additionally, this algorithm first detects, then intersects and classifies the intersections using barycentric coordinates. This work is directly applicable to (1) Mobile Network Computing and Spatial Reasoning, and (2) CAD/CAM geometric modeling where curves of intersection between a pair of surfaces is required for numerical control (NC) machines. Three experiments of the algorithm implementation are presented as a proof this feasibility.

KEYWORDS

Intersection Detection, Geometric Modeling, Classification Predicates, Spatial Reasoning, Triangle-Triangle Intersection.

1. INTRODUCTION

The intersection between 3D objects plays a prominent role in several areas including spatial reasoning [1], real-time rendering [2], collision-detection, geology [3], geometric modeling [4], and computer vision. Traditionally there are specialized algorithms for cross intersection and coplanar intersection. The intersection detection is a byproduct of actual intersection computations. Most of the time intersection detection should be done before the actual intersection. For example, in qualitative spatial reasoning, intersection detection is sufficient, actual intersection is not even needed, whereas in geometric applications like surface-surface intersection, it is desirable to have actual intersections. For early detection of intersection, we present a complete uniform algorithm independent of whether the triangles have cross intersection or coplanar intersection. Herein we illustrate this with an algorithm, its Python implementation is supported with output displayed in tables and figures. Typically, the boundary of a 3D object is represented as a triangulated surface and a triangle-triangle intersection is the computational basis for determining intersection between objects. Since an object boundary may contain thousands of triangles, algorithms to speed up the intersection detection process are still being explored for various applications, sometimes with a focus on innovations in processor architecture [5].

In qualitative spatial reasoning, spatial relations between regions are defined axiomatically using first order logic [6] or the 9-Intersection model [1]. It has been shown in [7] that it is sufficient to define the spatial relations by computing 4-Intersection predicates, (namely, Interior–Interior

(IntInt), Boundary–Boundary (BndBnd), Interior–Boundary (IntBnd), and Boundary–Interior (BndInt)) instead of 9-Intersections. In order to implement these algorithms, we must first implement the triangle-triangle intersection determination.

In geometric modeling, the surface-surface intersection problem occurs frequently. Since a free-form surface is represented as triangulated mesh, intersection between 3D objects amounts to detecting and computing intersection between 3D triangles. The algorithm presented here is extremely useful for geometric modeling where intersection between objects occurs thousands of times. For geometric applications cross intersection is most often used to the determine curves of intersection between surfaces [4]. Most of the approaches to these problems are based on code optimization of same or similar cross intersection algorithms. No attempt is made to design a single algorithm that handles both intersections simultaneously. Herein we present a new algorithm that fills the gap.

This paper is organized as follows: Section 2 discusses the background of possible cross and coplanar intersection between a pair of triangles. It describes the cross intersection, coplanar area intersection algorithm, and composite algorithms for general triangles. Section 3 develops the new generic single algorithm for triangle-triangle intersection, and classifies the intersections. Section 4 describes implementation experiments and the timing results. Section 5 describes the two of the applications where this approach is directly applicable. Section 6 concludes and references are given in Section 7.

2. BACKGROUND: *TRIANGLE-TRIANGLE INTERSECTION*

There is an abundance of papers devoted to intersection between a pair of triangles [8,9,10,11,12]. Interestingly, most of the papers concentrate on cross intersection, they simply reinvent the algorithm and optimize the code to implement it slightly differently and more efficiently, with no innovation. The cross intersection may result in a no intersection, a single point intersection or a line intersection. For coplanar triangles, it can result in area intersection as well. For coplanar intersection, such cross intersection algorithms do not work, there are separate algorithms for area intersection [10,11,12]. In these approaches, they first try to compute the intersection. Whether intersection is found or not, then it leads to conclude on the existence of intersection. The paper [12] surveyed various approaches for determining the cross intersection detection, and developed a fast vector version of the cross intersection detection, as well as classification of the type of intersection. The papers [8] and [11] also compare hardware implementation of their algorithm based on the number of arithmetic operations: +, -, *, /. Another paper [10] also compares the optimized intersection times. The papers [10,11] considered an approach for determining the intersection detection covering coplanar intersection, using edge to edge intersections. These approaches [10,11,12] led to two separate algorithms one for cross and one for coplanar intersection. There is no single algorithm that can handle both cross and coplanar intersection. We present a new algorithm that is analytically different, exhaustive and more rigorous than the previous algorithms [8,9,10,11,12]. It computes intersection using barycentric coordinates and vector equations very judiciously. Logical rather than computational tests are used to detect whether the intersection is a single point, or a line or an area. This new algorithm encompasses all cases implicitly and is different from previous ones in that we use only one equation rather than different algorithm for each special case. We show that it is possible to detect the existence of intersection in a uniform way before the precise intersection computations are computed.

2.1 Specialized Intersection Methods and Algorithms

Here we describe the conventional approach to triangle-triangle intersection in terms of two algorithms, one for cross intersection and one for coplanar intersection. Then we derive the composite algorithm for triangle-triangle intersection. Here we give the line intersection for cross intersection and area intersection for coplanar triangles separately and combine the two into one algorithm as is conventionally done. Then in Section 3 we present our new algorithm, which is a single algorithm that employs a more singular, seamless approach.

2.1.1 The Triangle-Triangle Line Intersection Algorithm

The cross intersection algorithm encompasses the single point and edge intersection cases. Here we give a solution *different* from previous approaches.

Algorithm for cross intersection line Algorithm

Input: Two triangles ABC and PQR

Output: Determine if they cross intersect. Return true if they intersect, otherwise return false.

boolean crossInt (tr1 = ABC, tr2 = PQR)

The vector equations for two triangles ABC and PQR are

$$R_1(u, v) = A + u\,U + v\,V,\ 0 \le u, v, u + v \le 1$$

$$R_2(s, t) = P + s\,S + t\,T,\ 0 \le s, t, s + t \le 1$$

where U = B - A, V = C - A, and S = Q - P, T = R – P are direction vectors along the sides of triangles.

Let $N_1 = U \times V$, $N_2 = S \times T$ be normals to the planes supporting the triangles. The triangles intersect if there exist some barycentric coordinates (u, v) and (s, t) satisfying the equation

$$A + u\,U + v\,V = P + s\,S + t\,T$$

Since $N_1 \times N_2 \neq 0$ for *cross* intersecting triangles, and S and T are orthogonal to N_2, the dot product of this equation with N_2 eliminates S and T from the above equation to yield

$$u\,U \bullet N_2 + v\,V \bullet N_2 = AP \bullet N_2$$

This is the familiar equation of a line in the uv - plane for real variables u, v. The vector equation using real parameter λ becomes

$$(u,v) = AP \bullet N_2 \frac{(U \bullet N_2, V \bullet N_2)}{U \bullet N_2^2 + V \bullet N_2^2} + \lambda(V \bullet N_2, -U \bullet N_2)$$

If there is a λ satisfying three equations such that $0 \le u, v, u + v \le 1$, then the triangles are ensured to intersect. The solution in λ is the range of values $\lambda_m \le \lambda \le \lambda_M$ for some λ_m and λ_M.

This gives the line of intersection of uv-parameter triangle with the st-parameter plane. Similarly the line of intersection of st-triangle with the uv-plane is computed. Then the common segment if any is the line intersection between the two triangles, for details see [9,13]. This algorithm works only if the triangles *cross* intersect.

2.1.2 The Triangle-Triangle Area Intersection Algorithm

The area intersection is possible for coplanar triangles only. If the triangles ABC and PQR are coplanar and a vertex of PQR is in the interior of ABC (or the converse is true), then an area intersection occurs. If there is no *edge-edge* intersection and no vertex of one triangle is inside the other triangle (or the converse is true), then they do not intersect, hence they are disjoint. The input, output, method prototype and pseudo code are given below.

Algorithm for area intersection
Input: Two triangles ABC and PQR coplanar
Output: If they do not intersect, then return false otherwise return true and the intersection area.
boolean coplanarInt (tr1 = ABC, tr2 = PQR)
The derivation of the algorithm is based on extensive use of the intersections of edges of ABC triangle with the edges of the PQR triangle and collecting the relevant interactions to form the area if any, for details see [11]. This algorithm can solve for intersection of *coplanar* triangles only.

The classical approach is to use crossInt (tr1, tr2), algorithm from section 2.1.1, when the triangle cross, and to use the second algorithm coplanarInt (tr1, tr2), algorithm from section 2.1.2, for coplanar triangles.
Thus separate algorithms are used to determine intersections on case-by-case bases. There is no single algorithm that detects intersection. We present a new single algorithm that does not depend on case-by-case intersections.

3. ALGORITHM FOR TRIANGLE-TRIANGLE INTERSECTION

Here we present a single algorithm that is analytically different, more comprehensive, robust and rigorous; it is implicitly capable of handling any specific type of intersection, which may be no intersection, a single point, a segment or an area. The Triangles may be coplanar or crossing. This single algorithm is not a modification of any previous algorithm, it is a new approach different from the other strategies.

The Main Algorithm
Input: Two triangles ABC and PQR in 3D, triangles may be coplanar or crossing
Output: If the triangles do not intersect, return false otherwise return true and the intersection, which may be single point, a line segment or an area.
boolean triTriIntersection (tr1 = ABC, tr2 = PQR)
The triangles ABC and PQR are
$X = A + u\,U + v\,V$ with $U = B - A$, $V = C - A$, $0 \le u, v, u + v \le 1$
$X = P + s\,S + t\,T$ with $S = Q - P$, $T = R - P$, $0 \le s, t, s + t \le 1$
To detect the intersection between the triangles ABC and PQR, we must attempt to solve
$A + u\,U + v\,V = P + s\,S + t\,T$ for some values $0 \le u, v, u + v, s, t, s + t \le 1$,
Rearranging this equation, we have
$u\,U + v\,V = AP + s\,S + t\,T$ (1)
where $AP = P - A$ is a vector
To solve the equation (1) for u, v, we dot equation (1) with $(U \times V) \times V$ and $(U \times V) \times U$, we quickly get u and v as
$u = -(\gamma \bullet V + s\,\alpha \bullet V + t\,\beta \bullet V)$
$v = \gamma \bullet U + s\,\alpha \bullet U + t\,\beta \bullet U$
$u + v = \gamma \bullet (U - V) + s\,\alpha \bullet (U - V) + t\,\beta \bullet (U - V)$
where $\alpha, \beta, \gamma, \delta$ are:
$\delta = (U \times V) \bullet (U \times V)$, $\alpha = \dfrac{S \times (U \times V)}{\delta}, \beta = \dfrac{T \times (U \times V)}{\delta}, \gamma = \dfrac{AP \times (U \times V)}{\delta}$
In order to satisfy the constraints $0 \le u, v, u + v \le 1$, we get three inequalities in parameters s and t

(a) $-\gamma \bullet U \le \alpha \bullet U\, s + \beta \bullet U\, t \le 1 - \gamma \bullet U$

(b) $-1 - \gamma \bullet V \le \alpha \bullet V\, s + \beta \bullet V\, t \le -\gamma \bullet V$

(c) $-\gamma \bullet (U - V) \le \alpha \bullet (U - V)\, s + \beta \bullet (U - V)\, t \le 1 - \gamma \bullet (U - V)$

These inequalities (a) - (c) are linear in s and t and are of the general form

$$m \le a\, s + b\, t \le n$$

To solve these inequalities we first eliminate t to get the possible range for s values, then solve for the range for possible t values as t(s). If there is no solution, then the algorithm returns false. If there is a solution, then it return s, t values as

$$s_m \le s \le s_M\ ,\ t_m(s) \le t \le t_M(s).$$

This discussion is summarized and the intersection points are classified as follows:

if the algorithm returns false,

    No Intersection

elseif ($s_m = s_M$ and ($t_m(s) = t_M(s)$ for $s_m \le s \le s_M$)

    Single Point Intersection

elseif ($s_m = s_M$ or ($t_m(s) = t_M(s)$ for $s_m \le s \le s_M$)

    Line segment intersection common to two triangles

else

    Area Intersection common to two triangles

This completes the discussion of our algorithm for intersection between triangles.

4. EXPERIMENTAL RESULTS

The algorithm is implemented in MacPython 3.3.3. The test time results are obtained via Python timeit utility. Time tests were performed on Apple Macintosh OS X processor (2.2 GHz intel Core i7). The average time for no intersection, single point intersection (0D), line intersection (1D) and area intersection (2D) are measured in seconds. The program is executed 100 times on each of 22 sample triangle pairs. The intersection times shown in Table 1 are neither optimized nor hardware embedded, they include classification of intersections also. Times are shown as proof of concept that this single integrated algorithm works reliably on all triangle pairs. The test data domain consists of synthetic triangles that have every possible type of intersection. For example, the times for no intersection are averaged over 100 runs with three samples. Similarly for single point intersection six sample pairs are averaged over 100 runs, then four samples are used for line segment and averaged over 100 runs, and finally nine sample triangle pairs are used and averaged. The composite average intersection time is computed. The test time statistics are displayed in Table 1. We also give three sample run output figures for examples of single point intersection, a line intersection and an area intersection. The Matlab software is used to draw the figures, see Fig. 1-3. The user interface allows user to select any type of the possible triangle-triangle pair for intersections and it displays the corresponding triangle and the intersection. For the sake of space consideration, one of the 0D (single point) intersections, see Fig. 1, one of the 1D line segment intersections, see Fig. 2; and one of 2D area intersections, see Fig. 3, are displayed here.

TABLE 1. EXECUTION AVERAGE TIMES OF ALGORITHM IN SECONDS

Type of Intersection	Time in Seconds
No Intersection	0.000073
Single Point	0.000531
Line Segment	0.001483
Area	0.001962
Overall	0.001353

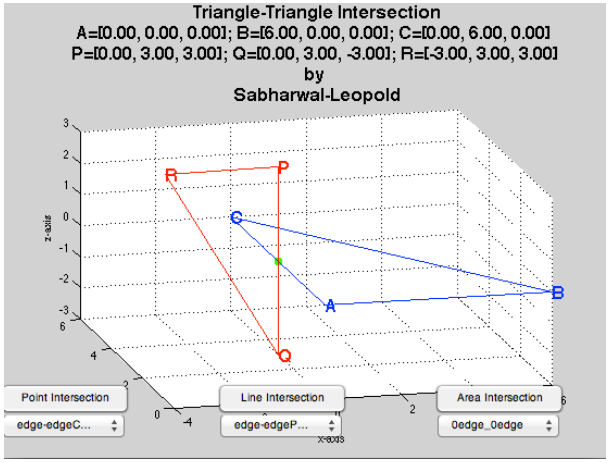

Fig. 1. Single Point Intersection

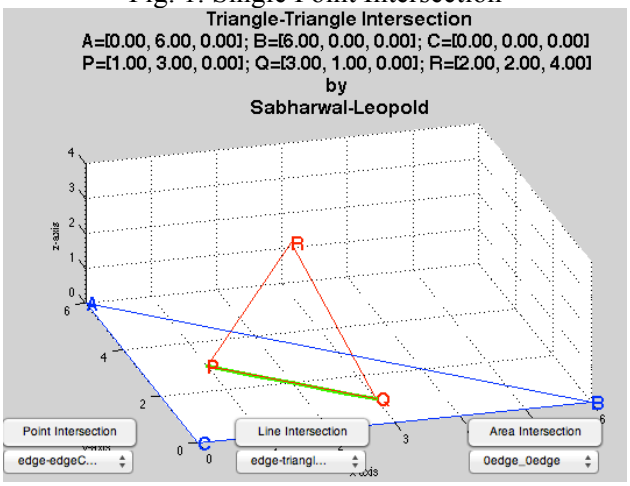

Fig. 2. Line Segment Intersection

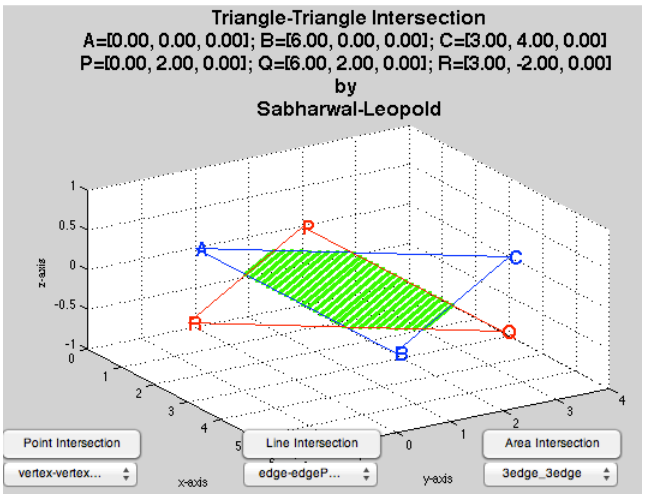

Fig. 3. Area Intersection

5. APPLICATIONS OF TRIANGLE-TRIANGLE INTERSECTION

Here we describe two of the applications where triangle-triangle intersection is directly applicable and is used extensively: qualitative spatial reasoning and geometric modeling. It is not limited to these two applications, other applications include virtual reality, and computer vision.

5.1 Qualitative Spatial Reasoning

In Qualitative Spatial Reasoning, the spatial relations are determined by the 9-Intersection/4-Intersection model [7, 8]. That is, for any pair of objects A and B, the interior-interior intersection predicate, IntInt(A, B), has true or false value depending on whether the interior of A and the interior of B intersect without regard to precise intersection. Similarly IntBnd(A, B) represents the truth value for the intersection of the interior of A and the boundary of B, and BndBnd(A,B) represents the predicate for the intersection of the boundaries of A and B. These four qualitative spatial reasoning predicates are sufficient to define RCC8 spatial relations [7].

In the application VRCC-3D+[13], the boundary of an object is already triangulated; that is, we will need to intersect pairs of only triangles. To reduce the computational complexity, the algorithm uses axis aligned bounding boxes (AABB) to determine the closest triangles which may possibly intersect [13]. Table 2 is a characterization of the intersection predicates, which subsequently can be used to resolve the eight RCC8 relations. Here we assume all normals are oriented towards the outside of the object. Each characterization in Table 2 describes when the associated predicate is true. If the truth test fails, then other triangles need to be tested. If no pair of triangles results in a true value, then the result is false.

TABLE 2. CHARACTERIZATION OF INTERSECTION PREDICATES

BndBnd	At least one pair of triangles (cross or coplanar) intersects.
BndInt	At least one pair tr1 and tr2 intersect, at least one vertex of tr2 is on the negative side of triangles of object 1. Or object 2 is contained inside object1, i.e. every vertex of object2 is on the negative side of triangles of object 1.
IntBnd	At least one pair tr1 and tr2 intersect, at least one vertex of tr1 is on the negative side of triangles of object 2. Or object 1 is contained inside object2, i.e. every vertex of object1 is on the negative side of triangles of object 2.
IntInt	At least one pair of triangles cross intersects (triangleInterior-triangleInterior) Or an object is contained in the other.

This characterizes the intersection predicates that help in resolving the RCC8 relations.

5.2 Geometric Modeling

In geometric modeling, the surface-surface intersection problem occurs frequently. In CAD/CAM, the objects are represented with surface boundaries using the ANSI (Brep) model. Intersection between 3D surfaces amounts to detecting and computing intersection between 3D triangles. Briefly, a pair of surfaces is subdivided using axis aligned bounding boxes (AABB) until the surfaces are reasonably planar and bounding boxes intersect. Then the plane surfaces are triangulated and the triangles are tested for cross-intersection and the intersection computed. The intersection segments are linked together to form curves of surface-surface intersection. The curves may be open curves, closed curves, or even a combination of both [4]. The algorithm presented here is extremely useful for geometric modeling where intersection between objects occurs thousands of times. For geometric applications cross intersection is most often used to obtain the line segment of intersection. Detailed analysis and implementation of the most comprehensive surface/surface intersection algorithm may be found in [4].

6. CONCLUSION

Herein we presented a single algorithm for the complete design and a robust implementation of a complete framework for determining and characterizing the triangle-triangle intersections. In contrast to other single track algorithms, this approach is a generic technique to detect any type cross or coplanar intersection using only one algorithm. The classification of intersections is based on logical tests on parametric coordinates rather than computational arithmetic tests in Cartesian coordinates. Thus our algorithm not only detects the existence but also classifies and computes the intersection as a single point, a line segment, or an area whichever the case may be. The algorithm provides more information than required by spatial reasoning systems. Consequently, we hope the additional information including classification of 3D intersection presented herein will be useful in other related applications.

7. REFERENCES

[1] Max J. Egenhofer, R. Franzosa, *Point-Set topological Relations,* International Journal of Geographical Information Systems 5(2), pp. 161 - 174, 1991.

[2] Oren Tropp, Ayellet Tal, Ilan Shimshoni, *A fast triangle to triangle intersection test for collision detection, Computer Animation and Virtual Worlds*, *Vol17 (50), pp.527 - 535, 2006.*

[3] G. Caumon, Collon - Drouaillet P, Le Carlier de Veslud C, Viseur S, Sausse J (2009) Surface - based 3D modeling of geological structures. Math Geosci 41:927–945, 2009.

[4] E.G. Houghton, Emnett R.F., Factor J.D. and Sabharwal C.L., *Implementation of A Divide and Conquer Method For Surface Intersections*; Computer Aided Geometric Design Vol.2, pp. 173 - 183, 1985.

[5] Ahmed H. Elsheikh, Mustafa Elsheikh, *A reliable triangular mesh intersection algorithm and its application in geological modeling*, Engineering with Computers, pp.1 - 15, 2012.

[6] D. A. Randell, Z. Cui, and A.G. Cohn, A *Spatial Logic Based on Regions and Connection*., *KR*, 92, pp. 165–176, 1992.

[7] C. Sabharwal and J. Leopold, "Reducing 9-Intersection to 4-Intersection for Identifying Relations in Region Connection Calculus", Proceedings of the 24th International Conference on Computer Applications in Industry and Engineering (CAINE 2011), Honolulu, Hawaii, Nov. 16-18, 2011, pp. 118-123, 2011.

[8] Mo¨ller T. *A fast triangle - triangle intersection test*. Journal of Graphics Tools, 1997; 2(2): 25–30.

[9] Held M. ERIT *a collection of efficient and reliable intersection tests*. Journal of Graphics Tools 1997; 2(4): pp. 25–44, 1997.

[10] Badouel Didier, *An Efficient Ray - Polygon Intersection,* Graphics Gems (Andrew S. Glassner, ed.), Academic Press, pp. 390 - 393, 1990.

[11] Guigue P, Devillers O. *Fast and robust triangle - triangle overlap test using orientation predicates*. Journal of GraphicsTools 2003; **8** (1): pp. 25–42, 2003.

[12] Chaman Sabharwal, Jennifer Leopold, and Douglas McGeehan,: *Triangle-Triangle Intersection Determination and Classification to Support Qualitative Spatial Reasoning,* Polibits, Research Journal of Computer Science and Computer Engineering with Applications Issue 48 (July–December 2013), pp. 13–22, 2013.

[13] N. Eloe, J. Leopold, C. Sabharwal, and Z. Yin, "*Efficient Computation of Boundary Intersection and Error Tolerance in VRCC-3D+* ", Proceedings of the 18h International Conference on Distributed Multimedia Systems (DMS'12), Miami, FL, Aug. 9 - 11, 2012, pp. 67 – 70, 2012.