Chaman L. Sabharwal and Jennifer L. Leopold, *An Implementation of Triangle-Triangle Intersection for Qualitative Spatial Reasoning,* Proceedings of the 29th International Conference on Computers and Their Applications (CATA–2014), March 24 – 26, 2014, Las Vegas, Nevada, USA, pp. 23-28, 2014.

An Implementation of Triangle-Triangle Intersection for Qualitative Spatial Reasoning

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 is ubiquitous in modeling. The objects are represented using the Boundary Representation (ANSI Brep) model in many applications such as CAD/CAM, collision detection, and spatial reasoning. Detection of possible intersection between objects can be based on the objects' boundaries (approximate triangulations), computing triangle-triangle intersection. Usually there are separate algorithms for cross and coplanar intersections. The intersection detection is a byproduct of actual intersection computations. For qualitative spatial reasoning, intersection detection is sufficient, actual intersection is not necessary. Herein we present an implementation of a complete uniform integrated algorithm independent of cross and coplanar intersection. Additionally, we use barycentric coordinates for detecting and classifying intersections prior to computing precise 3D coordinates. This work is applicable to most region connection calculi, particularly VRCC-3D+, which uses intersection detection between 3D objects as well as their projections in 2D that are essential for occlusion detection.

Keywords: *Intersection Detection, Classification Predicates, Spatial Reasoning, Triangle-Triangle Intersection*.

1 Introduction

The ability to detect the existence of possible intersection between pairs of objects is important in a variety of problem domains such as geographic information systems [1], real-time rendering [2], collision-detection, geology [3], networking and wireless computing. In qualitative reasoning, it is not necessary to know the precise intersection between pairs of objects; it is sufficient to detect and classify the intersection between objects. Typically, the boundary of each 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 [4].

In qualitative spatial reasoning, spatial relations between regions are defined axiomatically using first order logic [5] or the 9-Intersection model [6,7]. Using the latter model, the spatial relations are defined using the intersections of the interior, boundary, and exterior of one region with those of a second region. In order to implement these algorithms, we must first solve the triangle - triangle intersection problem, as it is a lower level problem that must be solved in order to determine the 4-Intersection predicates that, in turn, determine the qualitative spatial relation between two objects.

This paper is organized as follows: Section 2 briefly reviews the background and related inequalities resulting from intersection framework. Section 3 discusses the types of possible intersection between triangles Section 4 develops the overall main algorithm for triangle-triangle intersection, and classifies the intersections. Section 5 describes the applications to qualitative spatial reasoning. Section 6 concludes, followed by references in Section 7.

2 Background

2.1 The Traditional Algorithm

There is an abundance of papers devoted to the intersection between a pair of triangles [2,3,8,9,10,11]. Interestingly, most of them simply reinvent the algorithm and implement it slightly differently and more efficiently, with no innovation. The paper [11] 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. A recent paper [12] considered an approach for determining the intersection detection covering both cross and coplanar intersection simultaneously. Our approach follows [12] and is exhaustive and analytically more rigorous than the previous approaches [3, 8, 11]. It computes intersection barycentric coordinates that detect the intersection to be a single point, or a line or an area. The algorithm in [12] is not robust completely as it is missing two tests for (t_m(s), t_M(s)) yielding an approximate solution, which are also addressed here in the implementation for an accurate solution. Consequently, we show that the possible intersection can be detected before the precise intersection computations are performed.

This method of intersection solution involves three linear inequalities. The general principle for elimination of variables that works well for equations does not directly translate into solving inequalities. For example, the brute force method for solving these inequalities may lead to an extraneous solution, thus an erroneous solution as shown in the following example. Such approach gives an inconsistent solution to the inequalities:

$$-1 \le x + y \le 1 \quad (a)$$
$$-1 \le x - y \le 1 \quad (b)$$
$$-1 \le y - x \le 1 \quad (c)$$

Adding (a) and (b) yields $-1 \le x \le 1$, and adding (a) and (c) yields $-1 \le y \le 1$ which is the area enclosed by dotted boundary in Fig. 1. This is an inaccurate solution to the inequalities (a), (b), and (c). But the accurate solution is in the shaded area in Fig. 1, which is $|x| \le 1$, and $|y| \le (1 - |x|)$.

If the coefficients in (c) are modified and (c) becomes (c'): $0 \le y - x \le 0$, adding (a) and (c') yields $-1/2 \le x \le 1/2$, and y=x from (c'), which is a straight line as seen in Fig. 2.

Further if the coefficients in (a) are modified and (a) becomes (a'): $0 \le x + y \le 0$, we have y=x from (c'), and y=x=0 from (a'), then result is a Single point as shown in in Fig. 3.

Thus to accurately solve these inequalities $-1 \le x + y \le 1$, $-1 \le y - x \le 1$and $-1 \le x - y \le 1$, we first solve these for one variable x, then use this value to solve for the other variable y otherwise we have to eliminate the extraneous part of the solution.

First we solve two inequalities in the most general form:

$$m \le ax + by \le n \quad (1)$$
$$M \le Ax + By \le N \quad (2)$$

The following algorithm determines x_m, x_M such that for each x in $[x_m, x_M]$, the inequalities hold.

boolean solve_x (m, a, b, n, M, A, B, N, x_m, x_M)

If a solution is found, it returns true, else it returns false. First assume b and B are non-negative. If not, multiply them by - 1 to make them non-negative. Multiplying (1) by B and (2) by b, subtraction leads to

$$(mB - Mb) \le (aB - Ab)x \le (nB - Nb)$$

which yields the range $[\mathbf{x_m, x_M}]$ for x values provided $aB - Ab \ne 0$. In the case of zero, $(mB - Mb) \le (nB - Nb)$ must be satisfied in order to have solution for y values.

Now once x_m, x_M have been determined, for each x in $[x_m, x_M]$ in the inequalities, we determine the range $[y_m(x), y_M(x)]$ for y. That is, after the range $[x_m, x_M]$ is determined, only then for each x in $[x_m, x_M]$, the range for y is determined; in other words, y is a function of x.

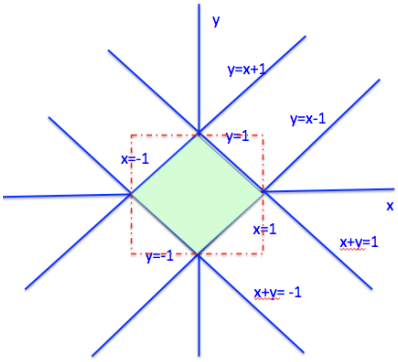

Fig. 1. Solution to inequalities: $-1 \le x - y \le 1$, $-1 \le x + y \le 1$ and $-1 \le y - x \le 1$. Using brute force method of elimination of variables yields the area enclosed by the dotted boundary, but the accurate solution is enclosed by the shaded area.

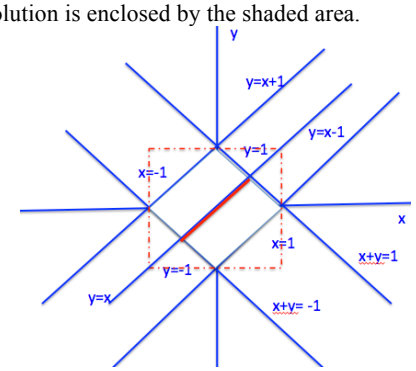

Fig. 2. Solution to inequalities: $-1 \le x - y \le 1$, $-1 \le x + y \le 1$, and $0 \le y - x \le 0$. The solution results in a shaded line segment.

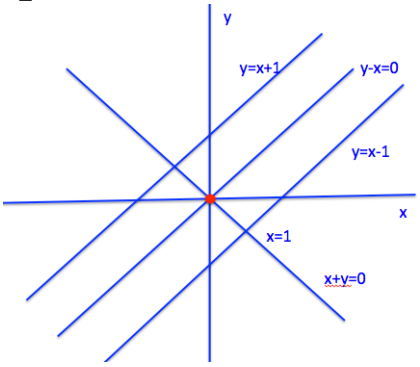

Fig. 3. Solution to inequalities: : $-1 \le x - y \le 1$, $0 \le x + y \le 0$ and $0 \le y - x \le 0$. The solution results in a shaded single point.

boolean solve_y (m, a, b, n, M, A, B, N, x, y_m, y_M)

Given that $x_m \le x \le x_M$ are known, it solves the inequalities for y_m, y_M . In the process it may update the values of x_m, x_M as needed.

If a solution is found, it returns true else it returns false. Now for $x_m \le x \le x_M$, the inequalities become

$$m - ax \le by \le n - ax \text{ and}$$
$$M - Ax \le By \le N - Ax.$$

These inequalities give the range $[y_m(x), y_M(x)]$ of values for y as function of x.

Another point to note in using vector equations
Solve for u and v

uU+vV=W

we can multiply with (UxV)xU, and (UxV)xV,

to get

u=WX(UxV)•V/ UX(UxV)•V

and

v=WX(UxV)•U/ VX(UxV)•U

Looks like a prefect solution. If we plug it into the equation it may or may not be it.

Example: x i + y j=k will yield x=y=0, but it does not satisfy the equation. Some test has to be made to ascertain that solution does work.

This complicates overfitting. We want solution that satisfies the constraints of the two triangles.

3 Types of Triangle Intersections

For spatial reasoning, we classify pairwise intersection based on the predicates IntInt(A,B) (intersection of Interior of object A and Interior of object B), IntBnd(A,B) (intersection of Interior of object A and Boundary of object B), BndInt(A,B), and BndBnd(A,B), without computing the precise extent of intersections. The cross intersection can be characterized into seven categories [7]. When cross intersection is insufficient to determine tangential intersection, some applications such as RCC8 and VRCC-3D$^+$ [6] resort to coplanar intersection to support relations such as externally connected (EC) and tangentially connected (TPP, TPPconverse). The precise intersection of coplanar triangles is a little more complex because it can result in area intersection as well. The coplanar triangles intersection can be classified as: Single Point Intersection (*vertex-vertex, vertex-edgeInterior*), Line Segment Intersection (*edge-edgeCollinear, edge-triangleInterior, triangleInterior-triangleInterior*) in Fig. 6 (a,b,c), Area Intersection bounded by 3, 4, 5, 6 edges, (Fig. 7(a,b), Fig. 8(a, b, c)). A triangle may be entirely contained in the other triangle (Fig. 8(d)). In this paper, we present a detailed analytical study of the intersection of triangles.

It is possible that two triangles cross intersect in a line segment even when a triangle is on one side of the other triangle. In that case, it may be desirable to know which side of the other triangle is occupied. In Fig. 6(b), the triangle PQR (except QR which is in ABC) is on the positive side of triangle ABC. So PQR does not intersect the interior of object of triangle ABC. We will use this concept in Section 5.

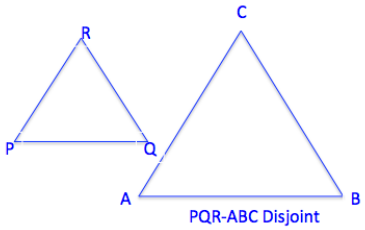

Fig. 4. Disjoint triangles: Planes supporting the triangles may be crossing or coplanar. The triangles do not have anything in common.

It should be noted that the vertex-edge intersection encompasses vertex-vertex and vertex-edgeInterior intersection, whereas the vertex-triangle intersection encompasses vertex-vertex, vertex-edgeInterior, and vertex-triangleInterior. Thus 1D JEPD cross intersection between ABC and PQR can be one of the three possibilities: (1) collinear along edges, (2) an edge of PQR lying in the plane of triangle ABC, or (3) triangles "pierce" through each other yielding an intersection segment.

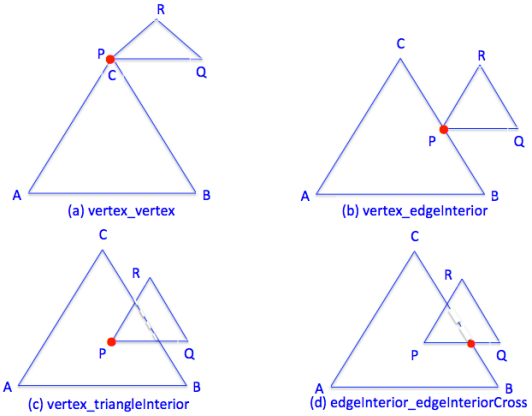

Fig. 5. Triangles intersect at a single point. The intersections between triangles ABC and PQR are JEPD (Jointly Exhaustive and Pairwise Distinct) cases of Single Point intersection between triangles. (a) vertex-vertex and (b) vertex-edgeInterior can occur in both cross and coplanar intersections. However, (c) vertex-triangleInterior and (d) edgeInterior-edgeInterior intersection point can occur in cross intersection only.

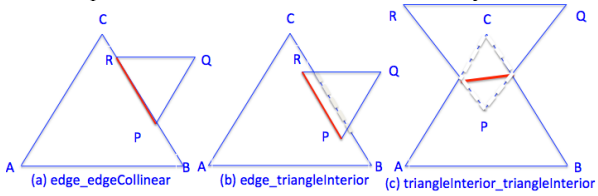

Fig. 6. Triangles intersect in a line segment. (a) edge-edgeCollinear intersection can occur in both cross and coplanar intersections. However, (b) edge-triangleInterior and (c) triangleInterior-triangleInterior intersection segment occur in cross intersection only.

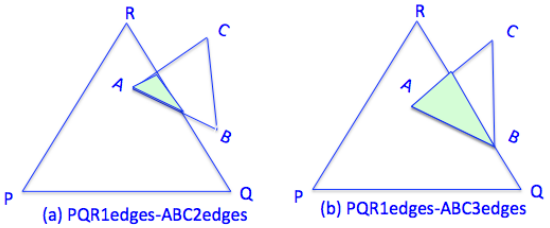

Fig. 7. Triangles intersect in an area (shaded). (a) One edge of triangle PQR and two edges AB and AC of triangle ABC

intersect, vertex A is in the interior of PQR. (b) One edge of triangle PQR with three edges of ABC, and vertex A in the interior of PQR. The common area is bounded by three edges. The intersections vertex-triangleInterior, edge_triangle, edgeInterior-triangleInterior hold.

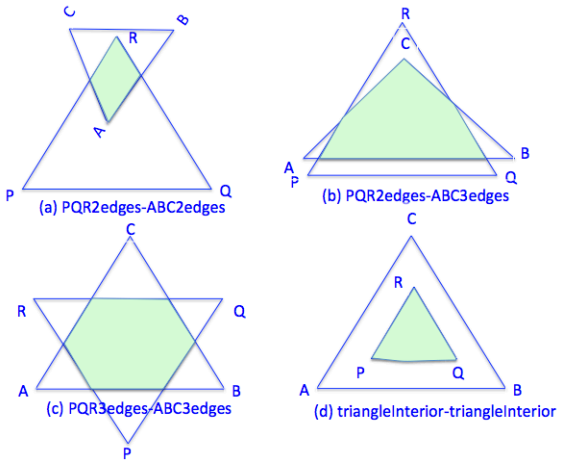

Fig. 8. Triangles intersect in an area (continued). The coplanar triangle intersections are bounded by four, five, and six edge segments. (a) Two edges of triangle PQR and two edges AB and AC of triangle ABC intersect, vertex A is in the interior of PQR, vertex R is in the interior of triangle ABC. The intersection area is bounded by four edges. (b) Two edges of triangle PQR and three edges of triangle ABC intersect; vertex C is in the interior of PQR. The intersection area is bounded by five edges. (c) Three edges of triangle PQR and three edges of triangle ABC intersect; every vertex of one triangle is outside the other triangle. The intersection is bounded by six edges. (d) No edge of triangle PQR intersects any edge of triangle ABC; vertices P, Q, R are in the interior of triangle ABC. The intersection area is the triangle PQR.

4 Intersection Algorithm

The examples Fig. 1-4 in section 3 indicate that there can be various types of intersections for both cross intersection and coplanar intersection between the triangles. All other configurations are homeomorphic to the figures presented in this paper. For qualitative spatial reasoning, in some cases (when the knowledge of cross intersection is insufficient), we resort to coplanar intersection to distinguish the externally or tangentially connected objects. The algorithm is implemented in Python 3.3.3.

4.1 General Purpose Algorithm

If a vertex of PQR is in the interior of ABC (or the converse is true), then an area intersection occurs, (Fig. 4(a, b), Fig. 5(a,b,d)). If no two edges intersect and *vertex_triangleInterior (*vertex, triangle = tr2) for every vertex of a triangle tr1, then the triangle tr1 is contained in tr2 and conversely. If no *edge-edge* intersection takes place and no vertex of one triangle is inside the other triangle (or the converse is true), then they are disjoint.

Some of the existing methods may use alternate edge-oriented techniques to determine the area of intersection, however those will be limited [8] lacking classification capability. Our algorithm is more comprehensive and analytically rigorous; it is implicitly capable of handling any specific type of intersection simultaneously, which may be a single point, a segment or an area.

THE ALGORITHM – A GENERAL APPROACH

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 $S = Q - P$, $T = R - P$, $0 \le s, t, s + t \le 1$

The general set up for detecting intersections is to solve the equation

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

with $0 \le u, v, u + v, s, t, s + t \le 1$,

Rearranging the equation, we have

$u\,U + v\,V = AP + s\,S + t\,T$ (1)

This is an underdetermined system of equations involving four parameters u, v, s, t and three equations in x, y, and z- coordinates. This means one of the parameters can be arbitrarily assigned. The other three parameters can be determined in terms one parameter provided the system is consistent otherwise not solution can be found due to inconsistency. We need additional constraint to have valid solution. Here the parameters u, v, s, t are bound such that $0 \le u, v, u+v \le 1$ and $0 \le s, t, s+t \le 1$. In addition, the solution is s, t must be consistent. This may be confirmed by checking the (s,t)-intersection must correspond to (u,v). This can be quickly determined that the point $X = P + s\,S + t\,T$ lies in the uv-plane by checking the dot product $AX \bullet UxV = 0$

For simplicity in solving (1), we use the following notation.

Let $AP = P - A$ be a vector, $\delta = (UxV) \bullet (UxV)$, and α, β, γ be vectors

$$\alpha = \frac{S \times (U \times V)}{\delta}, \beta = \frac{T \times (U \times V)}{\delta}, \gamma = \frac{AP \times (U \times V)}{\delta}$$

For intersection between triangles ABC and PQR, dot equation (1) with (UxV)xU and (UxV)xV, we quickly get

$u = -(\gamma \bullet V + s\,\alpha \bullet V + t\,\beta \bullet V)$

$v = \gamma \bullet U + s\,\alpha \bullet U + t\,\beta \bullet U$

Adding the two equations,

$u + v = \gamma \bullet (U - V) + s\,\alpha \bullet (U - V) + t\,\beta \bullet (U - V)$

In order that $0 \le u, v, u + v \le 1$, we get the following inequalities for possible range of values for 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 linear inequalities (a) - (c) are of the form

$m \le ax + by \le n$

The solution to this system of inequalities is derived in Section 3. We apply the results of those methods here in solving (a) - (c) pairwise.

$s_m = 0, s_M = 1$

If solve_x (- γ•U, α•U, β•U, 1 - γ•U, -1 - γ•V, α•V, β•V, - γ•V, x_m, x_M) // (a), (b)

$s_m = \max(s_m, x_m)$, $s_M = \min(s_M, x_M)$

If solve_x (- γ•U, α•U, β•U, 1 - γ•U, - γ•(U - V), α• (U - V), β•(U - V), 1 - γ•(U - V), x_m, x_M) // (a), (c)

$s_m = \max(s_m, x_m)$, $s_M = \min(x_M, s_M)$

If solve_x (- 1 - γ•V, α•V, β•V, - γ•V, - γ•(U - V), α•(U - V), β•(U - V), 1 - γ•(U - V), x_m, x_M) // (a), (c)

$s_m = \max(s_m, x_m)$, $s_M = \min(x_M, s_M)$

if $s_m > s_M$

return false

else

for $s \in [s_m, s_M]$ // we solve the (a)-(c) inequalities for t

$t_m(s) = 0$; $t_M(s) = 1$;

if solve_y(- γ•U, α•U, β•U, 1 - γ•U, -1- γ•V, α•V, β•V, - γ•V, s, y_m, y_M) //(a), (b)

$t_m(s) = \max(t_m(s), y_m)$, $t_M(s) = \max(t_M(s), y_M)$

if solve_y(- γ•U, α•U, β•U, 1 - γ•U, - γ•(U-V), α•(U-V), β•(U-V), 1 - γ•(U-V), s, y_m, y_M) //(a), (c)

$t_m(s) = \max(t_m(s), y_m)$, $t_M(s) = \min(t_M(s), y_M)$

if solve_y(-1- γ•V, α•V,β•V, - γ•V, - γ•(U-V), α•(U-V), β•(U-V),1 - γ•(U-V), s, y_m, y_M) //(b), (c)

$t_m(s) = \max(t_m(s), y_m)$, $t_M(s) = \min(t_M(s), y_M)$,

if $t_m(s) > t_M(s)$

return false

else

$t_m(s) \le t \le t_M(s)$

return true

/* end of algorithm */

We first solved the three inequalities pairwise for a range of values for s, so that $s_m \le s \le s_M$ holds good simultaneously with three inequalities. Then from this range of s values, we solved for t(s) as a function of s such that $t_m(s) \le t(s) \le t_M(s)$. There is no closed form function as such, It is a numerical solution to t(s) for each s. Thus if $s_m = s_M$ and t(s) is constant, it result in a single point else it is a line segment. Also if $s_m < s_M$ and $t_m(s) = t_M(s)$ for each s, it is a line segment. If $s_m < s_M$ and $t_m(s) < t_M(s)$ for some s, the it is an area intersetion. The parmetric bounding box for overall intersection is $[s_m, s_M] \times [t_m, t_M]$ where $t_m = \min\{ t_m(s): s_m \le s \le s_M\}$, and $t_M = \max\{ t_M(s): s_m \le s \le s_M\}$. The bilinear parameter coordinates for range of parameters are denoted by $p_m = (s_m, t_m)$, $p_M = (s_M, t_M)$ the corresponding 3D points are denoted by $P_m = P + s_m S + t_m(s_m) T$, and $P_M = P + s_M S + t_M(s_M) T$. In general $P([s_m, s_M], t(s)) = P + s S + t(s) T$.

This discussion may be summarized and the intersection points can be classified as follows:

If the algorithm returns false,

No Intersection

Elseif ($P_m = P_M$)

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 will implicitly cover the case when a triangle is inside the other triangle as well. If triangles do not intersect, then the triangles are declared *disjoint*. This completes the discussion of general algorithm for intersection between triangles.

Similarly, if required, we can determine (u, v) - parameter values corresponding to triangle ABC. This algorithm detects whether the triangles intersect regardless of crossing or coplanar triangles, and we classify the intersection as a *vertex, edge-interior or triangle-interior* point in Section 4.2.

This completes the general purpose algorithm discussion for determining the triangle - triangle intersection algorithm completely.

4.2 Classification Of Intersection

In Section 3, we developed sub-algorithms that support the main algorithm at its intermediate steps. In addition to existence or nonexistence of an intersection, this algorithm also supports other auxiliary computations, (e.g. classification of intersection and the calculation of actual 3D intersection points, segment or area) which are necessary for some applications.

If an application requires the (u,v) parameters for triangle ABC corresponding to (s,t) of triangle PQR, they can be quickly determined by using the computed (s,t) values as follows.

Input: X is a point of triangle PQR corresponding to parameters (s,t), X = P + s S + t T.

Output: (u,v) coordinates corresponding to (s,t).

Let

A + u U + v V = X

Rearranging the equation, we get

u U + v V = AX

To solve this, we dot product the equation with vectors (UxV)xU and (UxV)xV.

Let $\gamma = \dfrac{AX \times (U \times V)}{(U \times V) \bullet (U \times V)}$

then u = - γ•V and v = γ•U

The parameter values (s,t) are constrained by the barycentric constraints on (u,v). However solving this equation there may be extraneous (u,v) values which are pruned to stay within the (U,V) constraints.

This algorithm may be used with any application (e.g., qualitative spatial reasoning, surface modeling, image processing etc.). The algorithm determines whether intersection exists or not (i.e., it returns true or false). If true, the parameter coordinates of intersection are readily available. Now we can derive all the classification information from the parametric coordinates; only logical tests are sufficient for classification of the intersections. It is not the intent of this algorithm to determine whether the triangles are crossing or coplanar. This can be quickly determined as follows: if UxV•SxT ≠0, then triangles cross else triangle planes are parallel. If AP•UxV = 0 or AP•SxT = 0, then the triangles are coplanar.

In order to determine whether an intersection point X=(u,v) is a *vertex* of ABC, or on the *edge* of ABC, or an *interior point* of triangle ABC, no extra computational effort is required now. Logical tests are sufficient to establish the classification of this intersection. Since $0 \le u, v, u + v \le 1$, we can classify X relative to ABC in terms of the following predicates:

vertex ((u, v))**:** If (u, v) ∈ { (0, 0), (0, 1), (1, 0)}, then X is one of the vertices of ABC.

edgeInterior ((u, v))**:** If (u = 0, 0 < v < 1) or (v = 0, 0 < u < 1) or (u + v = 1, 0 < u < 1)), then X is on an edge of ABC, excluding vertices.

triangleInterior ((u, v))**:** If (0 < u < 1 and 0 < v < 1 and 0 < u + v < 1), X is an interior point (excluding boundary) of the triangle ABC.

Similarly for triangle PQR, ***vertex***((s,t)), ***edgeInterior*** ((s,t))**,** and ***triangleInterior*** ((s, t)) are determined.

5 Application

Qualitative Spatial Reasoning relies on intersections between objects whose boundaries are triangulated. The spatial relations are determined by the 9-Intersection/4-Intersection model [6,7]. 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 (see Table 1).

In the application VRCC-3D+, 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. For example, for objects A and B, if bounding boxes for triangles of A are disjoint from bounding boxes for triangles of B, either A is contained in B (BndInt, IntInt is true) or B is contained in A (IntBnd, IntInt is true) or A is disjoint from B. The test for such containment of objects can be designed by casting an infinite ray through the centroid of A. If the ray intersects B an odd number of times, then B is contained in A or A is contained in B. If A is not contained in B and B is not contained in A, then A and B are disjoint (i.e., IntInt(A, B), IntBnd(A, B), BndInt(A, B), and BndBnd(A, B) are all false).

Without the knowledge of BndBnd(A,B), BndInt(A,B), IntBnd(A,B), calculation of IntInt(A,B) is too costly. On the other hand, with this prior knowledge, it becomes quite inexpensive as the odd parity can be used for quick IntInt detection.

If the triangles cross intersect (e.g., *triangleInterior–triangleInterior* is true), then IntInt, IntBnd, BndInt, BndBnd will be true. However if the triangles are coplanar and intersect, only BndBnd(A, B) is true and IntInt(A, B), IntBnd(A, B), BndInt(A, B) are false for the objects; otherwise, BndBnd(A, B) is also false.

It is possible that two triangles cross intersect in a line segment even when a triangle is on one side of the other triangle, so *edgeInterior–triangleInterior* is true. In that case, it may be desirable to know which side of the other triangle is occupied. In Fig. 3(b), the triangle PQR is on the positive side of triangle ABC. For example, if triangle1 of object A cross intersects the negative side of triangle2 of object B, then BndInt(A, B) is true.

Table 1 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 1 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 1. 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 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.
IntBnd	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.
IntInt	At least one pair of triangles cross intersects (triangleInterior-triangleInterior) Or an object is contained in the other.

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

6 Conclusion

For the 9-Intersection model used in qualitative spatial reasoning, triangle - triangle intersection plays a prominent role. Herein we presented an implementation of a complete framework for determining and characterizing the intersection of geometric objects. In contrast to other algorithms, this approach is a general technique to detect any type of intersection. It creates classifications by applying logical tests rather than computational arithmetic tests. Thus our algorithm not only detects whether or not an intersection exists, but also classifies intersections as a single point, a line segment, or an area. The algorithm provides more information than required by spatial reasoning systems. Consequently, we hope the new ideas and additional information including classification of 3D intersection presented herein will be useful in other related applications.

7 References

[1] Max J. Egenhofer, R.G. Golledge, *Spatial and Temporal Reasoning in Geographic Information Systems*, Oxford University Press, USA, 1998.

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

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

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

[7] Chaman L Sabharwal and Jennifer L Leopold, *"Reducing 9-Intersection to 4-Intersection for identifying relations in region connection calculus*, " in The 24th International Conference on Computer Applications in Industry and Engineering, 2011, pp. 118–123, 2011.

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

[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] Chaman Sabharwal, and Jennifer Leopold, "*A Fast Intersection Detection Algorithm for Qualitative Spatial Reasoning*", Proceedings of the 19h International Conference on Distributed Multimedia Systems (DMS'13), Brighton, UK, Aug. 8 - 10, pp. 145-149, 2013.

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