Polibits, Research Journal on Computer Science and Computer Engineering with Applications
Issue 48 (July–December 2013), pp. 13–22, 2013

Triangle-Triangle Intersection Determination and Classification to Support Qualitative Spatial Reasoning

Chaman L Sabharwal[1], Jennifer L Leopold[2], Douglas McGeehan[3]
Missouri University of Science and Technology
Rolla, Missouri, USA – 65409
{[1]chaman, [2]leopoldj, [3]djmvfb}@mst.edu

Abstract. ***In CAD/CAM modeling, objects are represented using the Boundary Representation (ANSI Brep) model. Detection of possible intersection between objects can be based on the objects' boundaries (i.e., triangulated surfaces), and computed using triangle-triangle intersection. Usually only a cross intersection algorithm is needed; however, it is beneficial to have a single robust and fast intersection detection algorithm for both cross and coplanar intersections. For qualitative spatial reasoning, a general purpose algorithm is desirable for accurately differentiating the relations in a region connection calculus, a task which requires consideration of intersection between objects. Herein we present a complete uniform integrated algorithm for both cross and coplanar intersection. Additionally, we present parametric methods for classifying and computing intersection points. This work is applicable to most region connection calculi, particularly VRCC-3D+, which detects intersections 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.*

I. INTRODUCTION

There are relatively few software applications supporting qualitative spatial reasoning. In part, this may be due to the complexity in determining the intersection between 2D/3D objects. Yet the ability to detect the existence of a possible intersection between pairs of objects can be important in a variety of problem domains such as geographic information systems [1], CAD/CAM geometric modeling [2], real - time rendering [3], geology [4], 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 [5, 6, 7].

For pairs of triangles, there are three types of intersections: zero dimensional (single point), one dimensional (line segment), and two dimensional (area) intersection. In the past, almost all attention has been devoted to determining the cross intersections, which resulted in an absence of analysis in two dimensional intersections. Coplanar triangle intersections are unique because an intersection may be any of the aforementioned three types. If the triangles cross intersect, only zero or one dimensional intersection is possible. If the planes are parallel and distinct, the triangles do not intersect. If the triangles are coplanar, then there is a possibility of intersection. Even when the cost of intersecting a triangle pair is constant, the cost of intersecting a pair of objects A and B is order $O(T_A * T_B)$ where T_A is the number of triangles in object A, and T_B is the number of triangles in object B.

In qualitative spatial reasoning, spatial relations between regions are defined axiomatically using first order logic [8] or the 9-Intersection model [9]. 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. It has been shown in [10] 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. Since IntBnd and BndInt are the converse of each other, only three algorithms are necessary for these predicates. In order to implement these algorithms, we must first solve the triangle - triangle intersection determination, as it is a lower level problem that must be solved in order to determine the 4-Intersection predicates which, in turn, determine the qualitative spatial relation between two objects.

This paper is organized as follows: Section II briefly reviews the background and related cross intersection framework. Section III discusses motivation and conceptual classification of intersections, whereupon Section IV develops the overall main algorithm for triangle-triangle intersection. Section V describes the area intersection algorithm for general triangles, and predicates for classifying the intersection between pairs of triangles, after which Section VI discusses the applications to qualitative spatial reasoning. Section VII concludes, followed by references in Section VIII.

II. BACKGROUND

A. The Traditional Algorithm

Many papers have been written on the intersection between a pair of triangles [3, 11, 12, 13, 14, 15]. Interestingly, most of them simply reinvent the algorithm and implement it slightly differently and more efficiently, with no innovation. A recent paper [7] 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. Our approach is exhaustive, integrating both cross and coplanar intersection, and analytically more rigorous than the previous approaches [3, 11]. It is described in the next section where we follow the approach similar to the techniques used in [7] for cross intersection. The cross intersection standalone algorithm is described as follows:

boolean triTriCrossInt (tr1 = ABC, tr2 = PQR)
input: two triangles whose planes cross intersect
output: true if the triangles intersect, else false
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.
Let $N_1 = UxV$, $N_2 = SxT$ be normals to the planes supporting the triangles directed away from the objects. 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 $N_1xN_2 \neq 0$ 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

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

Then parameter values u, v are explicitly written as

$$u = AP \bullet N_2 \frac{U \bullet N_2}{U \bullet N_2^2 + V \bullet N_2^2} + \lambda V \bullet N_2$$

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

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

If there is a λ in these three equations such that $0 \le u, v, u + v \le 1$, the triangles are ensured to intersect. The range of values of λ is bounded by λ_m and λ_M. This detects whether the two triangles cross intersect only.

In fact, for precise intersection, using λ_m, λ_M, as parameter values, we compute (u_m, v_m) and (u_M, v_M) for the segment of intersection on ABC. Similarly the values (s_m, t_m) and (s_M, t_M) represent the segment of intersection on PQR. The precise intersection between the two triangles is the common segment of these two segments. If the segment degenerates into a single point, the parameter values also can be used to classify the intersection as a vertex, an edgeInterior point or triangleInterior point in the triangle ABC.

III. CLASSIFICATION OF TRIANGLE INTERSECTIONS

For spatial reasoning, we detect intersection between pairs of 2D/3D objects and classify pairwise intersection predicates IntInt, IntBnd, BndInt, and BndBnd, without computing the 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, TPPc). 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*), Area Intersection bounded by 3, 4, 5, 6 edges, (Fig. 4, Fig. 5(a, b, c)). A triangle may be entirely contained in the other triangle (Fig. 5(d)). In this paper, we present a detailed analytical study of the intersection of coplanar triangles, which has not been previously presented.

The intersection between a pair of triangles can be abstracted as: Cross (C) intersection or Parallel (P) coplanar triangles intersection. For taxonomy of cross and parallel coplanar triangles, the conceptual intersections are supported with figures presented here. The specific cases are as follows:

No intersection
- *disjoint* (C, P) (see Fig. 1)

Single Point Intersection
- *vertex-vertex* Intersection (C, P) (see Fig. 2(a))
- *vertex-edgeInterior* Intersection (C, P) (see Fig. 2(b))
- *vertex-triangleInterior* Intersection (C) (see Fig. 2(c))
- *edgeInterior-edgeInteriorCross* Intersection (C) (Fig. 2(d))

Line intersection
- *edge-edgeCollinear* Intersection (C, P) (see Fig. 3(a))
- *edge-triangleInterior* Intersection (C) (see Fig. 3(b))
- *triangleInterior-triangleInterior* Intersection (C) (Fig. 3(c))

Area Intersection
- *vertex-triangleInterior* Intersection (P) (see Fig. 4, Fig. 5(a, b, d))
- *edgeInterior-edgeInterorCross* Intersection (P) (Fig. 4, Fig. 5(a, b, c))
- *edge-triangleInterior* Intersection (P) (see Fig. 5(d))
- *triangleInterior-triangleInterior* Intersection (P) (see Fig. 4, Fig. 5(a, b, c, d))

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. 3(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 VI.

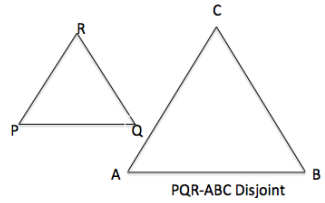


Fig. 1. Disjoint triangles: Planes supporting the triangles may be crossing or coplanar. The triangles do not have anything in common.

It should be noted that the vertex-edge intersection encompasses vertex-vertex, 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.

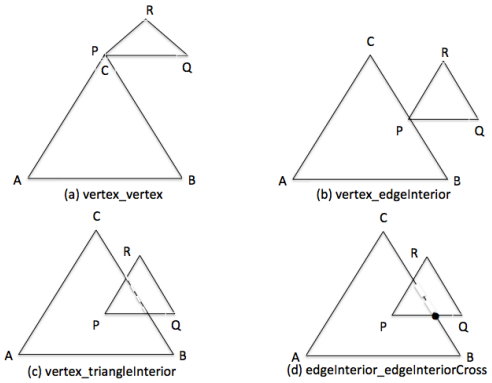


Fig. 2. Triangles intersect at a single point. The intersections between triangles ABC and PQR are JEPD (Jointly Exhaustive and Pairwise Distinct) cases of Single Point intersection between triangles. (a) vertex-vertex and (b) vertex-edgeInterior can occur in both cross and coplanar intersections. However, (c) vertex-triangleInterior and (d) edgeInterior-edgeInterior intersection point can occur in cross intersection only.

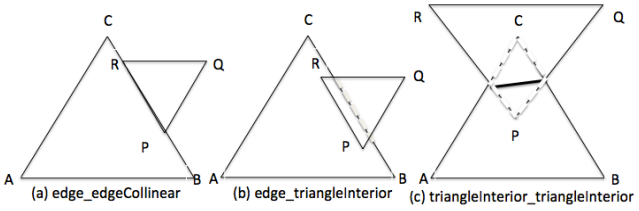


Fig. 3. Triangles intersect in a line segment. (a) edge-edgeCollinear intersection can occur in both cross and coplanar intersections. However, (b) edge-triangleInterior and (c) triangleInterior-triangleInterior intersection segment occur in cross intersection only.

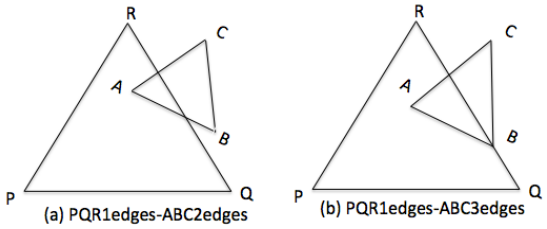


Fig. 4. Triangles intersect in an area. (a) One edge of triangle PQR and two edges AB and AC of triangle ABC intersect, vertex A is in the interior of PQR. (b) One edge of triangle PQR with three edges of ABC, and vertex A in the interior of PQR. The common area is bounded by three edges. The intersections vertex-triangleInterior, edge_triangle, edgeInterior-triangleInterior hold.

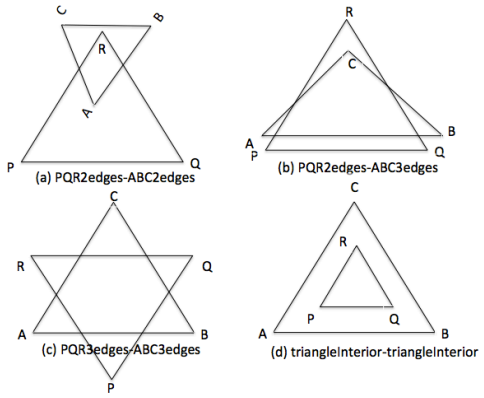


Fig. 5. Triangles intersect in an area (continued). The coplanar triangle intersections are bounded by four, five, and six edge segments. (a) Two edges of triangle PQR and two edges AB and AC of triangle ABC intersect, vertex A is in the interior of PQR, vertex R is in the interior of triangle ABC. The 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 area 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.

IV. THE OVERALL ALGORITHM (INTERSECTION BETWEEN TRIANGLES)

In this section, we describe the overall structure of the triangle - triangle intersection. In Section IV.A, we develop 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 3D intersection points, segment or area) which are necessary for some applications.

A. Description Of The Overall Algorithm

The general structure of the overall triangle-triangle intersection algorithm is presented here. The description is in Python style so that it can be easily transported to programmable code. Here is the traditional approach to the algorithm, whereas our approach is presented in Section V.

boolean triTriInt(tr1 = ABC, tr2 = PQR)

Input: two triangles ABC and PQR

Output: Boolean value whether the triangles intersect or not.

Let ABC and PQR be two triangles. The triangles are represented with parametric vector equations where u, v are parameters for triangle ABC, and s, t are parameters for triangle PQR.

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

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

where

$U = B - A$, $V = C - A$, are directions of the edges at A;

$S = Q - P$, $T = R - P$ are the directions of edges at P.

Let $N_1 = U \times V$, $N_2 = S \times T$ be the normals to planes supporting the triangles ABC and PQR.

if $N_1 \times N_2 \neq 0$ // planes supporting triangles are not parallel
  if triTriCrossInt (tr1, tr2) // cross intersect the triangles
    return true
  else
    return false
elseif $N_1 \times N_2 = 0$, // triangles planes are parallel
    if $AP \bullet N_1 = 0$, //the triangles are coplanar
      if triTriParInt (tr1, tr2)// implicit in Section V.
        return true
      else
        return false
    elseif $AP \bullet N_1 \neq 0$, //the triangles are not coplanar,
      no Intersection
      return false
    endif
endif
/*end of algorithm*/

Here we give all the supporting algorithms for implementation and classification of all special case intersections in the main algorithm. There are three broad categories for intersections of triangles: zero dimensional (single point), one dimensional (line segment), and two dimensional (area) intersection.

A. 1 Single Point Intersection (0D).

We first analyze the vertices of the triangle PQR with respect to triangle ABC to determine if a vertex P or Q or R is common to the ABC triangle and conversely.

vertex-triangleTest (X, tri = ABC)

Input: X is a vertex of one triangle and tri another triangle.

Output: boolean value determining whether X is a vertex, edgeInterior, triangleInterior point of the triangle.

To determine the relation of $X \in \{P, Q, R\}$ to the triangle ABC, we solve

$A + u\,U + v\,V = X$ for $0 \le u, v, u + v \le 1$,

Rearranging the equation, we get

$u\,U + v\,V = AX$

To eliminate one of the parameters u, v 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 $u = -\gamma \bullet V$ and $v = \gamma \bullet U$

if $0 \le u, v, u + v \le 1$,
  return true // X of PQR, intersects the triangle ABC.
else
  return false
/*end of algorithm*/

The vector $\dfrac{(U \times V)}{(U \times V) \bullet (U \times V)}$ is computed only once and used repeatedly. As a result $\gamma = \dfrac{AX \times (U \times V)}{(U \times V) \bullet (U \times V)}$ is calculated with one cross product, and u, v are calculated with one dot product. The parameters u, v naturally lend themselves to classification of intersections. Similarly $\gamma' = \dfrac{PX \times (S \times T)}{(S \times T) \bullet (S \times T)}$.

A. 2 Classification Of Intersection.

In order to determine whether the vertex X of triangle PQR 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 $(u, v) \in \{(0, 0), (0, 1), (1, 0)\}$, 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, as above we can classify vertex X of triangle ABC as *vertex, edgeInterior, or triangleInterior* point of triangle PQR. Single point intersection may result from cross intersection of edges as well. An edge point may be a vertex or an interior point of the edge.

A. 3 The Edge-edge Single Point Intersection.

If two triangles cross intersect across an edge, the edge-to-edge intersection results in a single point. The edge-edge cross intersection algorithm is presented below.

edge_edgeCrossIntersection (edge1, edge2)

Let the two edges be AB and PQ. Then the edges are represented with equations

$X = A + u\,U$ with $U = B - A$, $0 \le u \le 1$

$X = P + s\,S$ with $S = Q - P$, $0 \le s \le 1$

if $UxS \bullet AP \neq 0$, return false //non - coplanar lines
elseif $UxS = 0$, return false //lines are parallel
else $UxS \neq 0$, // lines cross
/* solve for u_P, s_A values for the intersection point*/

$A + u_P\,U = P + s_A\,S$

$$u_P = \frac{S \bullet PA \times (U \times S)}{(U \times S) \bullet (U \times S)}$$

if ($u_P < 0$) or ($u_P > 1$), return false //no cross intersection,

$$s_A = \frac{U \bullet AP \times (U \times S)}{(U \times S) \bullet (U \times S)}$$

if ($s_A < 0$) or ($s_A > 1$),
    return false //no cross intersection,
else
    return true //there is edge-edge cross intersection.
endif
/* end of algorithm*/

A. 4 Composite Classification Of Single Point Intersection.

Here we represent the intersection point uniformly for both the algorithms. Let A_m, P_m, be the pair of bilinear parametric coordinates of the 3D intersection points $R_1(u_m,v_m)$ and $R_2(s_m,t_m)$ with respect to triangles ABC and PQR respectively. When there is no confusion, we will refer to the points as A_m and P_m instead of 3D points $R_1(u_m,v_m)$ and $R_2(s_m,t_m)$. From vertex-triangle intersection (Section 3) we have

P_m is a vertex of PQR, and $A_m = (u_m, v_m)$, where u_m and v_m are $u_m = -\gamma \bullet V$, $v_m = \gamma \bullet U$

or A_m is a vertex of ABC, and $P_m = (s_m, t_m)$, where s_m and t_m are $s_m = -\gamma' \bullet T$, $t_m = \gamma' \bullet S$

From edge-edge intersection (Section B.3) we have

$A_m = (u_m, v_m) = (0, u_P)$ or $(u_P, 0)$ or $(u_P, 1 - u_P)$ or $(1 - u_P, u_P)$

$P_m = (s_m, t_m) = (0, s_A)$ or $(s_A, 0)$ or $(s_A, 1 - s_A)$ or $(1 - s_A, s_A)$

If ($u_m = 0$ or 1) and ($v_m = 0$ or 1) and ($s_m = 0$ or 1) and ($t_m = 0$ or 1), it is *vertex-vertex* intersection. If ($u_m = 0$ or 1) and not ($s_m = 0$ or 1), it is *vertex-edgeInterior* intersection. If not ($u_m = 0$ or 1) and ($s_m = 0$ or 1), it is *edgeInterior-vertex* intersection. If {not ($u_m = 0$ or 1) & $v_m=0$} or {not ($v_m = 0$ or 1) & $u_m=0$} or { not ($u_m = 0$ or 1) & not ($v_m = 0$ or 1) & $u_m + v_m = 1$} and {not ($s_m = 0$ or 1) & $t_m=0$} or {not ($t_m = 0$ or 1) & $s_m=0$} or {not ($s_m = 0$ or 1) & not ($t_m = 0$ or 1) & $s_m + t_m = 1$}, it is *edgeInterior-edgeInterior* intersection. This completes the discussion of single point intersection classification and parameters for the corresponding 3D points.

B. Line Intersection (1D)

Besides edge-edge cross intersection, the edge-edge collinear intersection is a possibility, independent of crossing or coplanar triangles. In this section we discuss algorithms that result in a segment (1D) intersection; see Fig. 3.

B. 1 Intersection Algorithm And Parametric Coordinates.

Here we derive an edge-edgeCollinear intersection algorithm. This algorithm is seamlessly applicable to both cross intersecting and coplanar triangles. The following algorithm implements intersection of edges of the triangles ABC and PQR.

boolean edge-edgeCollinearTest (edge1, edge2)

input: two line segments

output: true if the segments have a common intersection, else false. First we compute the linear parameter coordinates u_P, u_Q, s_A, s_B for intersection of X = A + u (B - A), for X = P, Q and X = P + s (Q - P), for X = A, B. Similarly we can compute the intersection of other edges of triangle ABC with any edge of triangle PQR. Then we update the parameters for the common segment. This algorithm is standard, straight forward and is omitted for the sake of limited space.

B. 2 Uniform Representation Of Edge-edge Intersection Of Edge-edge Intersection.

Again we first represent the intersection parameters uniformly as $A_m = (u_m, v_m)$, $A_M=(u_M, v_M)$ and $P_m = (s_m, t_m)$, $P_M = (s_M, t_M)$. Now we have the linear coordinates for intersection points u_P, u_Q and s_A, s_B. We map the linear parameters for intersection points to bilinear parameter coordinates (u, v) and (s, t). If u_P, u_Q are known along an edge and the edge is AB, let $u_m = u_P$, $u_M = u_Q$, $v_m = 0$, $v_M = 0$;

Similarly for AC, let $v_m = u_P$, $v_M = u_Q$, $u_m = 0$, $u_M = 0$;

and for BC, let $u_m = u_P$, $u_M = u_Q$, $v_m = 1 - u_P$, $v_M = 1 - u_Q$;

Thus ABC triangle bilinear coordinates for the intersection points are:

$A_m=(u_m, v_m)$, $A_M=(u_M, v_M)$

where $v_m=v_M = 0$ or $u_m=u_M =0$ or $u_m+v_m=u_M + v_M = 1$

Similarly for the triangle PQR, the linear coordinates s_A, s_B of intersection translate into bilinear coordinates

$P_m = (s_m, t_m)$, $P_M = (s_M, t_M)$

where $t_m = t_M = 0$ or $s_m = s_M = 0$ or $s_m+t_m = s_M+t_M = 1$

.

Now we have the bilinear parametric coordinates u, v, s, t for the intersection segment. The common 3D segment is denoted by $[R_1(A_m), R_1(A_M)]$ which is $[R_2(P_m), R_2(P_M)]$ or $[R_2(P_M), R_2(P_m)]$. It is possible that the intersection segment is equal to both edges, or it overlaps both edges, or it is entirely contained in one edge. Since the intersection is a part of the edges, it cannot properly contain any edge.

B.3. Composite Classification Of Line Intersection. For collinear edge intersection A_m, A_M are normally distinct and similarly P_m, P_M may be distinct. Though the intersection segment is given by $[R_1(A_m), R_1(A_M)] = [R_2(P_m), R_2(P_M))$ or $[R_1(A_m), R_1(A_M)] = [R_2(P_m), R_2(P_M)]$, it is not necessary that parameter coordinates $[A_m, A_M] = [P_m, P_M]$ or $[A_m, A_M] = [P_M, P_m]$. The predicate for edge-edge collinear intersection segment becomes

*edge-edgeCollinear (*edge1, edge2*) = edge* ($[A_m, A_M]$) and *edge* ($[P_m, P_M]$) and $[R_1(A_m), R_1(A_M)] == [R_2(P_m), R_2(P_M)]$ or $[R_1(A_m), R_1(A_M)] == [R_2(P_M), R_2(P_m)]$

Also it may be noted that for a cross intersection triangle, an *edge-triangleInterior* intersection may result in a segment intersection (Fig. 3(b)). For cross intersecting planes we have (cf. 3.A for vertex to triangle intersection and [7]) .

*edge-triangle (*edge, triangle*) = edge* ($[A_m, A_M]$) and *triangle* ($[P_m, P_M]$) and $[R_1(A_m), R_1(A_M)] == [R_2(P_m), R_2(P_M)]$ or $[R_1(A_m), R_1(A_M)] == [R_2(P_M), R_2(P_m)]$

This completes the discussion of segment intersection (1D), classification, 3D points for both *cross and coplanar* triangle intersections.

V. AREA INTERSECTION

For coplanar triangles, there may be no intersection (Fig. 1), a single point (Fig. 2(a, b)), a segment (Fig. 3(a)) or an area (Fig. 4, Fig. 5(a, b, c)), including one triangle contained in another, (Fig. 5(d)). An area can result from two edges of one triangle and one, two, or three edges of another triangle, or three edges from both triangles creating a star shaped figure. The resulting area is bounded by 3, 4, 5, or 6 edges. 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.

A. 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.

Although this algorithm may look simple, it is a new approach compared to previous approaches cited in the background section. The existing methods may use alternate edge-oriented techniques to determine the area of intersection, however those will be limited [11]. 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 NOVEL 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

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

for u, v, s, t. If a solution exists satisfying the constraints $0 \le u, v, u + v, s, t, s + t \le 1$, then there is an intersection, else there is no intersection.

Rearranging the equation, we have

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

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

Let α, β, γ be vectors and δ be a positive real number. Then for triangle ABC, let

$AP = P - A$ be a vector, $\delta = (UxV)\bullet(UxV)$,

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

Similarly, let α', β', γ' be vectors and δ' be a positive real number. Then for triangle PQR, let

$PA = A - P$ be a vector, $\delta' = (SxT)\bullet(SxT)$

$$\alpha' = \frac{U \times (S \times T)}{\delta}, \beta' = \frac{V \times (S \times T)}{\delta}, \gamma' = \frac{PA \times (S \times T)}{\delta}$$

For intersection between triangles ABC and PQR,

on dotting 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 at the end of this section. We apply the results of the algorithms here in solving (a) - (c).

If solve_x ($-\gamma\bullet U, \alpha\bullet U, \beta\bullet U, 1 - \gamma\bullet U, -\gamma\bullet V, \alpha\bullet V, \beta\bullet V, 1 - \gamma\bullet V, x_m, x_M$)

$s_m = \max(0, x_m)$, $s_M = \min(1, x_M)$

If solve_x ($-\gamma\bullet U, \alpha\bullet U, \beta\bullet U, 1 - \gamma\bullet U, -\gamma\bullet(U - V), \alpha\bullet(U - V), \beta\bullet(U - V), 1 - \gamma\bullet(U - V), x_m, x_M$)

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

If solve_x ($-1 - \gamma\bullet V, \alpha\bullet V, \beta\bullet V, -\gamma\bullet V, -\gamma\bullet(U - V), \alpha\bullet(U - V), \beta\bullet(U - V), 1 - \gamma\bullet(U - V), x_m, x_M$)

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

if $s_m > s_M$

return false

else

$t_M = 0$; $t_m = 1$

for $s \in [s_m, s_M]$ // we solve the inequalities for t

if solve_y ($-\gamma\bullet U, \alpha\bullet U, \beta\bullet U, 1 - \gamma\bullet U, -\gamma\bullet V, \alpha\bullet V, \beta\bullet V, 1 - \gamma\bullet V, s, y_m, y_M$)

$t_m(s) = \max(0, y_m)$, $t_M(s) = \min(1, y_M)$,

$t_m = \min(t_m(s), t_m)$, $t_M = \max(t_M(s), t_M)$ // extent of overall t values

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 as a function of s such that t_m (s) ≤ s ≤ t_M (s), and overall t_m ≤ t_M. If it succeeds, it ensures that there is a solution. Similarly we determine for u - parameter and v - parameter values in terms of u to obtain the area enclosed by the two triangles. This algorithm detects whether coplanar triangles intersect, and we classify the intersection as in Section V.B. Here we describe the two algorithms we applied in the general purpose algorithm. An auxiliary algorithm solves inequalities of the form

$m \le ax + by \le n$ and

$M \le Ax + By \le N$

The brute force method for solving these inequalities may lead to an erroneous solution as shown in the following example. The general elimination of variables principle that works well for equations does not directly translate into solving inequalities. Such approach gives an inconsistent solution to the two inequalities

(a) $-1 \le x + y \le 1$ and

(b) $-1 \le x - y \le 1$

Since $-1 \le x - y \le 1$ is equivalent to $-1 \le -x + y \le 1$, adding and subtracting the two inequalities (a) and (b), yields an inaccurate answer $-1 \le x \le 1$, and $-1 \le y \le 1$ which is the area enclosed by dotted boundary in Fig. 6. But the accurate solution is in the shaded area in Fig. 6, which is $|x| \le 1$, and $|y| \le (1 - |x|)$.

Thus to accurately solve these two inequalities $-1 \le x + y \le 1$ and $-1 \le x - y \le 1$, we first solve these for one variable x, then use this variable value to solve for the other variable y as $-(1 - |x|) \le y \le (1 - |x|)$.

First we solve two most general inequalities

$m \le ax + by \le n$ (1)

$M \le Ax + By \le N$ (2)

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

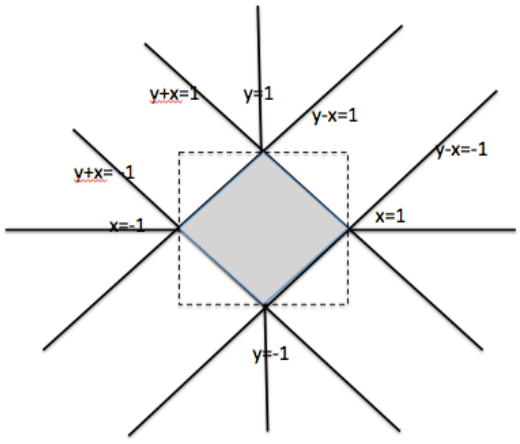


Fig. 6. Solution to a pair of inequalities: $-1 \le x + y \le 1$ and $-1 \le x - y \le 1$. Using brute force method of elimination of variables yields the area enclosed by the dotted boundary, but the accurate solution is enclosed by the shaded area.

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 $[\mathbf{x_m}, \mathbf{x_M}]$ for x values in addition to true or false value for the algorithm.

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.

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

$m - ax \le by \le n - ax$ 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.

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

B. Composite classification for area intersection

In this section, we summarize the algorithms in Section V.A. The equations of the triangles ABC and PQR are

$R_1(u, v) = A + u\,U + v\,V$,
where $U = B - A$, $V = C - A$, $0 \le u, v, u + v \le 1$
$R_2(s, t) = P + s\,S + t\,T$,
where $S = Q - P$, $T = R - P$, $0 \le s, t, s + t \le 1$

These equations are independent of whether they are supported by crossing planes or coplanar planes. The cross intersecting triangles discussion is well researched, see Section II. Here we consider the general case, including crossing or coplanar triangles. In this case the intersection may be an area in addition to a possible single point and a line segment. We first determined $[s_m, s_M]$ the range of s values, then used the range on s to solve for $[t_m(s), t_M(s)]$, the range of t. If such a solution exists, it is ensured that the two triangles intersect, which is sufficient for some qualitative spatial reasoning applications. The uv values can be similarly derived for the triangle ABC (e.g., first u_m, u_M then $v_m(u)$, $v_M(u)$). This algorithm may be used with any application (e.g., qualitative spatial reasoning, surface modeling, image processing etc.). As described in Section III, an intersection can arise from crossing or coplanar triangles. For example, vertex-vertex or edge-edge intersection can occur regardless of triangles being coplanar or crossing. 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. We can derive all the auxiliary 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. The bilinear parameter coordinates are denoted by $A_m = (u_m, v_m)$, $A_M = (u_M, v_M)$, $P_m = (s_m, t_m)$, $P_M = (s_M, t_M)$. The intersection points can be differentiated as follows.

If the algorithm returns false,
  No Intersection
Elseif ($A_m = A_M$) or ($P_m = P_M$)
  Single Point Intersection
Elseif ($s_m = s_M$ or $t_m = t_M$ or $u_m = u_M$ or $v_m = v_M$)
  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 overall intersection between triangles.

VI. APPLICATION TO QUALITATIVE SPATIAL REASONING

Qualitative Spatial Reasoning relies on intersections between objects whose boundaries are triangulated. The spatial relations are determined by the 9-Intersection/4-Intersection model [9, 10]. 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 the 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 (IntInt, BndInt is true) or B is contained in A (IntInt, IntBnd 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. Similarly, the test can be made if 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).

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 2 enumerates the outcome for triangle - triangle intersection with respect to 3D objects. This 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 1. RCC8 RELATIONS AND INTERSECTION PREDICATES, ONLY SHADED ENTRIES ARE NECESSARY.

RCC8	IntInt	BndBnd	IntBnd	BndInt
DC	F	F	F	F
EC	F	T	F	F
PO	T	T	T	T
EQ	T	T	F	F
TPP	T	T	F	T
NTPP	T	F	F	T
TPPc	T	T	T	F
NTPPc	T	F	T	F

TABLE 2. CHARACTERIZATION OF INTERSECTION PREDICATES

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

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

VII. CONCLUSION

For the 9-Intersection model used in qualitative spatial reasoning, triangle - triangle intersection plays a prominent role. Herein we presented a complete framework for determining and characterizing the intersection of geometric objects. In contrast to other algorithms, our 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.

VIII. REFERENCES

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

[2] 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.

[3] 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*.

[4] 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.

[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] 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.

[7] Chaman L Sabharwal, and Jennifer L 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, 2013, (accepted to appear).

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

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

[10] Sabharwal C. L. and J. 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.

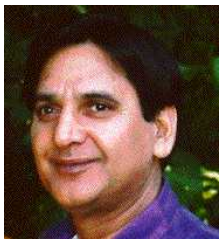
[11] Guigue P, Devillers O. *Fast and robust triangle - triangle overlap test using orientation predicates*. Journal of GraphicsTools 2003; **8** (1): pp. 25–42, 2003.

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

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

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

[15] C. L. Sabharwal, *Survey of implementations of cross intersection between triangular surfaces*, MDC Report Q0909 (Now Boeing at St. Louis, MO - USA), 1987.



Chaman L. Sabharwal was born at Ludhiana, Punjab, India, in 1937. He received his B.A.(Hons) in 1959, M.A.(Math) in 1961 from Punjab University, Chandigarh, India. He received his M.S.(Math) in 1966 and Ph.D.(Math) from the University of Illinois, Urbana, Champaign, Illinois, USA, in 1967.

He is professor of Computer Science at Missouri University of Science and Technology (1986-). He was assistant professor (1967-971), associate professor (1971-9175), full professor (1975-1982) at Saint Louis University. He was Senior Programmer Analyst(1982), Specialist(1983), Senior Specialist(1984) Lead Engineer(1985) at Boeing Corporation. He published several technical reports and journal articles on CAD/CAM. He was consultant at Boeing (1986-1990). He was National Science Foundation fellow (1979) at Boeing and NSF Image Databases Panelist (1996).

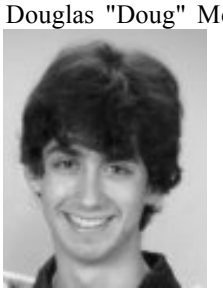
Dr. Sabharwal has been member of American Mathematical Society, Mathematical Society of America, IEEE Computer Society, ACM, and ISCA. He has been on editorial board of International Journal of Zhejiang University Science (JZUS), Editorial Board CAD(Computer Aided Design), Progress In Computer Graphics Series, Modeling and Simulation, Instrument Society of America. He has been a reviewer for numerous books, journals and conferences. He was awarded service awards by NSF Young Scholars George Engelmann Institute, and ACM Symposium on Applied Computing for Multimedia and Visualization track.

Jennifer L. Leopold was born in Kansas City, Missouri, USA, in 1959. She received her B.S. (Math) in 1981, M.S. (Computer Science) in 1986, and Ph.D. (Computer Science) in 1999, all from the University of Kansas, USA.

She is graduate coordinator (2013-) and associate professor (2008-) of Computer Science at Missouri University of Science and Technology. She started as assistant professor (2002-2008) at Missouri University of Science and Technology. Previous to that appointment she was a research associate

at the University of Kansas Biodiversity Research Institute (1999-2002). She also has worked in industry for over ten years as a systems analyst. She has several publications in end-user programming, with particular focus on database accessibility and scientific visualization. She has received over $2.5M in NSF funding, predominantly for bioinformatics research to develop and study software tools that allow end-users to use powerful information technology to enhance their research without the need for traditional programming training.

Dr. Leopold has been a member of IEEE Computer Society and ACM. She has been a reviewer for numerous books, journals, and conferences. She has served as conference co-chair for the International Conference on Distributed Multimedia Systems (2012) and the International Workshop on Visual Languages and Computing (2013), and proceedings chair for the IEEE Symposium on Bioinformatics and Computational Biology (2014). She has served as an invited participant in several NSF-funded workshops, particularly in bioinformatics.



Douglas "Doug" McGeehan was born in Jefferson City, Missouri in 1991, and began his college career at the age of 16 when he was admitted to the Missouri Academy of Science and Mathematics, Maryville, MO. He received his B.S. in Computer Science (Cum Laude) from the Missouri University of Science and Technology, Rolla, MO, in 2013, and is currently working on his Ph.D. in Computer Science at the same institution.

From 2011 until 2013, he has been a GIS Software Engineer for the U.S. Geological Survey, where he became interested in research into spatial reasoning and computational geometry. He is currently working as a Graduate Research Assistant in the Web & Wireless Computing Laboratory at Missouri S&T, Rolla, MO, investigating delay tolerant networks, spatiotemporal optimization, cryptology, and open security and privacy problems in cloud computing.

Mr. McGeehan's awards include an Outstanding Service Award from the Computer Science department at Missouri S&T for his work as a volunteer tutor from 2011 to 2012, and five consecutive semesters on the Academic Scholar's List of Missouri S&T. He is currently a U.S. Department of Education GAANN Fellow under Dr. Sanjay Madria and Dr. Dan Lin, his Ph.D. advisors.