ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,
Volume II-2/W1, ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, 27 – 29 November 2013, Istanbul, Turkey

AN APPROACH TO DEVELOP 3D GEO-DBMS TOPOLOGICAL OPERATORS BY RE-USING EXISTING 2D OPERATORS

Daniel Xu[a,*], Sisi Zlatanova[a]

[a] Jaffalaan 9, 2628 BX, Delft, Netherlands
xusifeng@gmail.com
S.Zlatanova@tudelft.nl

KEY WORDS: 3D, Geo-DBMS, 9IM, Oracle Spatial, topological function, city model

ABSTRACT:

Database systems are continuously extending their capabilities to store, process and analyse 3D data. Topological relationships which describe the interaction of objects in space is one of the important spatial issues. However, spatial operators for 3D objects are still insufficient. In this paper we present the development of a new 3D topological function to distinguish intersections of 3D planar polygons. The development uses existing 2D functions in the DBMS and two geometric transformations (rotation and projection). This function is tested for a real dataset to detect overlapping 3D city objects. The paper presents the algorithms and analyses the challenges. Suggestions for improvements of the current algorithm as well as possible extensions to handle more 3D topological cases are discussed at the end.

1 INTRODUCTION

Topology in 2D has been employed to describe the urban environment. It serves well when the third dimension (e.g. height) is not critical. However, there are situations in reality which are difficult to model only by 2D. For example, an innovative architecture is built in such way that 'a highway would go through the building' (figure 1(a)). Or a building has its part hanging above the other (figure 1(b)).

(a) A highway 'goes through' an office building.

(b) A building has its part hanging over the other.

Figure 1: Real world situations that are difficult to model only by 2D topology.

To model 3D topology, a number of 3D topological frameworks have been introduced from different research projects. Examples of such frameworks are the 9 Intersection Model (9IM) (Egenhofer, 1995), Dimensional Model (Billen et al., 2002), Dimension Extended Method (or DE-9IM) (Medeiros and Andrade, 1994) (Clementini et al., 1993). The most accepted framework is the 9IM. Much research has been completed on extending the framework in 3D, e.g. (Guo et al., 1998), (Zlatanova, 2000) or for dealing with polygons with holes (McKenney and Schneider, 2008). The 9IM has been adopted by OGC as the model to develop implementation specifications (Herring, 2010-08-04). The current implementation specification considers only 8 operators - equals, disjoint, touches, within, overlaps, crosses, intersects and contains - as they are the most commonly used. These are the possible relations between two objects of the same dimension embedded in space, which has the same dimension (e.g. polygons in 2D space or polyhedrons in 3D space) (Zlatanova, 2000).

9IM is realised in Geo-DBMSs such as PostGIS and Oracle, to calculate 2D topological relationships for point(s), line(s) and polygon(s). While significant improvement of such implementation in 2D is constantly being improved, the development toward 3D topological relationships is slow. The latest version of PostGIS (2.0) (PostGIS 2.0 online Manual, 2012) gives support to real 3D objects. But still the functions and object types are limited, e.g. to only points and line strings. Oracle Spatial 11g supports also native 3D object types and the largest set of 3D operators. However, most of the Spatial 3D operators are non-topological (distance calculation, validation, visibility, etc.) If one wants to identify basic topological relationships, e.g. 'overlap' as in 9IM, there is no existing 3D function ready to use. Oracle Spatial, for example, supports ANYINTERACT and INSIDE (for solids only) in 3D (Oracle, 2010).

The development of 3D operators is apparently a challenging issue and requires careful consideration of dimension of objects, their geometry and the space. Therefore we have experimented with development of 3D operators re-using existing 2D operators. 2D Geo-DBMS operators are already quite mature and can handle a variety of cases. Combining them will save efforts and time in extending the 3D functionally of Geo-DBMSs.

In this paper, we discuss an approach to realise two operators named 'overlap' and 'meet' between two 3D planar polygons, using Oracle Spatial operators SDO_ANYINTERACT (3D) and SDO_RELATE (2D), and applying rotation and projection on the polygons. The algorithm is further extended to determine the 'overlap' relation between two polyhedrons. And it is implemented as an operator in Oracle Spatial and tested against different polygons and one real world dataset.

The remaining part of the paper is organised as the following: section 2 discusses research related to 9IM and its implementations in 3D. Section 3 describes the algorithm in details. Section 4 discusses the tests and illustrates some performance issues in challenging situations. Section 5 concludes the findings and suggests future developments towards 3D functionality in Geo-DBMSs.

2 RELATED WORK

Topological relationships between two objects in the space are those that do not change with respect to geometric transformations, such as rotation and distortion. It has been a 'central and important concept in GIS since the mid 1970s' (Van Oosterom et al., 2006). Different formalisms have been developed to enumerate all the possible relationships that would take place in 2D and 3D space. The 9IM developed by Egenhofer and Herring ((Egenhofer and Franzosa, 1995) (Egenhofer et al., 1993)), is the most well-known topological formalism in geographic information science. And it has been approved by OpenGeospatial Consortium as a basic framework for implementation. It also forms part of ISO19107 standard (ISO, 2008-09-17). Many researchers have contributed to the extension and implementation of this framework (Chen et al., 1998), (Haarslev and Möller, 1997), (Zlatanova, 2000) and (Ellul and Haklay, 2009).

While we discuss the implementation of 3D topological framework, two related issues need to be addressed: one is the support of native 3D data types and the other is the maturity of implementation on topology ((Ellul and Haklay, 2006), (Khuan et al., 2008) and (Arens et al., 2005)).

By now all mainstream DBMSs (Oracle, PostGIS, IBM DB2, Informix, Ingres, MySQL, SQL Server) support the OGC geometry model, but basically in 2D space, e.g. point, line and polygon. Some of them (Ingres) supports higher dimension (z value) but do not really consider it in calculation (ActianCommunity, 2011). PostGIS support 3D geometry types e.g. 3D polygons and polyhedral surfaces (PostGIS 2.0 online Manual, 2012). However, a support for solids is still missing. Oracle Spatial is most advanced in support of a 3D type (i.e. solid). Additionally, different 3D operators are realised on this data type. One example is a 3D geometry validation operator - SDO_Geom.Validate_Geometry_With_Context (Oracle, 2010) - which validates solid objects against GML 3.1.1 and ISO 19107 specifications. Queries concerning 3D visibility, volumetric analysis and spatial/semantics attribute are available in Spatial 11g (Kazar et al., 2008) as well.

The second issue, the maturity of topology implementation, refers to both: topological models and frameworks for detecting topological relationships (Ellul and Haklay, 2006). In the mainstream DBMSs, only PostGIS and Oracle Spatial maintain topological models, but only in 2D. Currently, there is no 3D topological model that is natively supported by DBMS. Therefore the operator for detecting topological relationships are developed on the geometry model of Geo-DBMSs. PostGIS and Oracle Spatial are the only spatial DMS that support topological operators. The former adopts DE-9IM (Clementini et al., 1994) with a consideration of dimension issue upon 9IM, while the later chooses the classical 9IM. As mentioned previously, these implementations are very limited in 3D. Oracle Spatial supports the topological operators ANYINTERACT (determining disjoint and non-disjoint) and Inside (9IM).

Many 3D topological models have been presented in the literature and many approaches have been developed to construct a topologically correct data sets (e.g. (Ledoux and Meijers, 2009), (Ghawana and Zlatanova, 2010), (Boguslawski et al., 2011)). However many of these models have not discussed topological frameworks yet.

The work described in (Brugman et al., 2011) establishes topological relationship between different geometric primitives (data types), e.g. node, edge, ring, face, shell and volume. The relationships are stored in user-defined tables. The structure implemented in Oracle Spatial and validated with pre-defined set of rules start from the (valid) geometric primitives and are further refined in different levels.

As is presented in (Ellul and Haklay, 2009), an extended implementation of 9IM topological relationships is carried out on a topological data structure (user-defined in Oracle). It adopts the boundary representation as proposed by (Hoffmann, 1989) (page 37) to define 3D objects and the set of possible relationships as presented in (Zlatanova, 2000). A 3D solid is represented as constituent Surface components (or primitives in other contexts) - Nodes (0D), Edges (1D) and Faces (2D). Nodes (or vertices) store the actual coordinates. The interior of the solid is defined by the space enclosed by the Faces. Each face is defined by edges which in turn are defined by nodes. Nodes, edges, faces and the relations between these primitives are modelled as database tables. With an optimisation of the traditional B-Rep structure - binary B-Rep (BB-rep) and modified binary B-Rep (MBB-rep) - the author presented an approach to model all the 9I relationships identified by (Zlatanova, 2000) in 3D space. PL/SQL (Procedural Language/SQL) is used to implement this concept and run to test queries.

Another interesting implementation is by (Borrmann and Rank, 2009) who applies Octree algorithm to topological relationships determination in the model of buildings. The prototype is implemented in Oracle 10g. Spatial (topological) operators, such as touch, are implemented as methods of spatial data types and can be used in the WHERE part of an SQL statement. An example was given in this paper:

```
SELECT *
FROM IFCColumn col, IFCSlab slab3
WHERE col.touch(slab3) AND slab3.id = '0id23089';
```

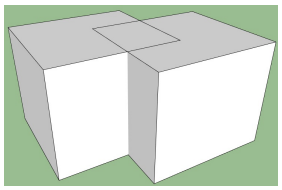
which extracts all columns that touch the slab whose ID is 0id23089. According to the author, the declaration of user-defined types is advantageous in that it provides a strong type safety. For example, declaration of the touch operator forces the operands to be of type SpatialObject or one of its subtypes. Thus type errors can be already detected by the query engine and a more specific error report can be generated for the end-user.

3 3D TOPOLOGICAL OPERATORS 'OVERLAP' AND 'MEET'

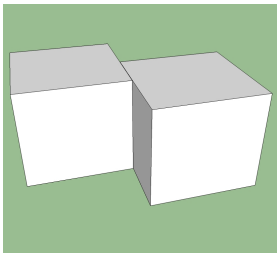
The algorithm is developed under the following considerations:

Firstly, we develop our algorithm explicitly for Oracle Spatial as it supports the most 3D topological operators. The similar development for PostGIS would require further investigations on which functions are the most appropriate.

Secondly, only two 3D operators named here 'overlap' and 'meet' are realised, since they have been found most appealing for 3D real objects during two student projects (Boufidou et al., 2011) and (Xu, 2011). As it was mentioned in previous section, ANYINTERACT(3D), Relate(2D) and INSIDE are the only topological operators according to the 9IM. ANYINTERACT tells if two objects are disjoint or not. It does not discern further what happens in the non-disjoint situation. However, this missing information is of great interest in many real world cases. Depending on the semantic of object, different situations might be considered possible. For example, an 'overlap' of two real world buildings must be an error, while two neighbour buildings can have a 'meet' relation (which is also non-disjoint) (see figure 2(b)). Buildings and tress may not 'overlap', but 'two trees overlap' is allowed (see figure 3). These relationships can be used as constraints to

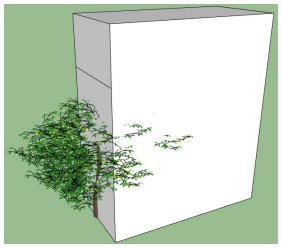


(a) 'Two buildings overlap' may not be allowed in 3D models.

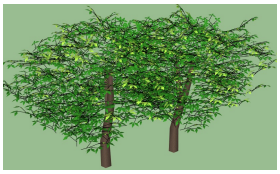


(b) 'Two buildings meet' may be allowed in 3D models since it is realistic.

Figure 2: Details within non-disjoint can lead to serious discussion about validity of 3D model.



(a) A tree and a building 'overlap'. This may not be allowed in 3D models.



(b) 'Two trees overlap' is common in real life and thus may be allowed in 3D models.

Figure 3: Objects with the same geometry but different semantics may be treated differently in 3D models.

check validity of 3D data sets especially when they are obtained from different reconstruction methods (Xu, 2011).

Thirdly, the algorithm assumes that the 3D objects are constructed by planar polygons. According to (OGC, 2008), many 3D topographic objects are commonly modelled by faces (3D polygons) or a collection of faces when stored in databases. Therefore, in our approach we first develop a method to determine the relationships between two 3D planar polygons, then between a 3D polygon and a polyhedron, and finally we propose a further development of this method to tell what happens between two polyhedral objects. At this point we assume that 3D polygons and polyhedrons are valid, according to rules of Oracle Spatial (Oracle, 2010). This implies that the 3D polygons are planar. However they can be concave. A polygon with a hole is also investigated.

3.1 Identify Topological Relations Between Two 3D Planar Polygons

As listed in (Zlatanova, 2000), there are in total 14 possible relationships between two planar polygons in 3D. These are disjoint, meet, cover, coveredBy, inside, contains, equal, overlap and 6 other which do not have names. As mentioned previously, we are interested of two cases named meet and overlap, which however have slightly difference in the meaning. With meet we want to indicate that interiors of the objects do not intersect (see figure 4). Overlap stands for all cases in which the interiors intersect (see figure 5(a)). This means that 5 of above mentioned possible operations will be classified as 'meet' and 8 as 'overlap'. We do not consider disjoint as it is taken care by the 3D operator ANYINTERACT. The operation 'equal' is not taken into consideration either. The 9IM matrices, which correspond to our case 'meet' are (i-interior, b-boundary, e-exterior):

$$\begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 1 & 1 \\ bA & 0 & 0 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 0 & 1 \\ bA & 1 & 0 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 1 & 1 \\ bA & 0 & 1 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 0 & 1 \\ bA & 1 & 1 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$

$$\text{('meet')} = \begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 0 & 1 \\ bA & 0 & 1 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$

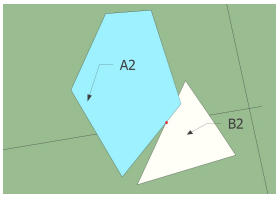
The 9IM matrices, which correspond to our case 'overlap' are (i-interior, b-boundary, e-exterior):

$$\begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 0 & 1 \\ bA & 1 & 0 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 1 & 1 \\ bA & 0 & 0 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$

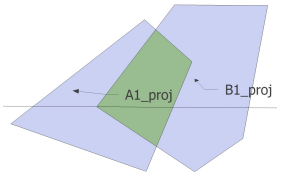
$$\begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 0 & 0 \\ bA & 1 & 0 & 0 \\ eA & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 1 & 1 \\ bA & 0 & 0 & 1 \\ eA & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 1 & 1 \\ bA & 0 & 1 & 1 \\ eA & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 0 & 0 \\ bA & 1 & 1 & 0 \\ eA & 1 & 1 & 1 \end{bmatrix}$$

$$\text{('overlap')} = \begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 1 & 1 \\ bA & 1 & 1 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$



(a) In 3D, two touching polygons share a part of boundary (in this case it is a point), which is contained by the line of intersection.

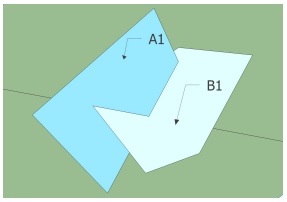


(b) In 2D, the shared geometry (the overlapping triangle) only touches the line of intersection.

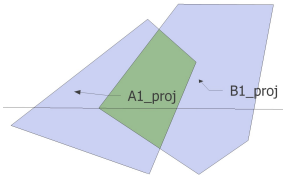
Figure 4: Touching polygons in 3D, their projections in 2D and the line of intersection.

The principle of the developed algorithm is that each 3D polygon belongs to a plane (infinite boundary). When two planes are not parallel, they meet in a line of intersection. If any polygon from one plane should have any spatial contact with a polygon from the other plane, the contact can only occur somewhere along the line of intersection. If two polygons overlap, this line will cross the interior of both. If they do not disjoin neither overlap, then they meet (see figure 4(a)).

To implement this concept, an algorithm was developed using Oracle 9I operators available in 2D, simple geometry transformation (rotation) as well as operator that returns shared 2D geometry (SDO_Intersection). (The original PL/SQL script can be accessed



(a) In 3D, the line of intersection from two planes crosses both A1 and B1's interiors.



(b) In 2D, the projected intersection line crosses the shared geometry of both polygons. In this case A1_proj and B1_proj share a polygon (see the green hole).

Figure 5: Overlapping polygons in 3D, their projections in 2D and the line of intersection. The hollow in 2D is the shared geometry.

via http://www.sourcepod.com/lyqxhg32-19780). Algorithm details are explained below:

Given any valid (in Oracle) polygon A and polygon B, if they have any contact, operator SDO_ANYINTERACT() returns True (step 0). (Although the ring test from (Xu, 2011) shows that ANYINTERACT may see a disjoint case as non-disjoint, e.g. polygon going through a ring without any contact, it does not miss any non-disjoint case. So if ANYINTERACT gives result FALSE then two polygons must have no spatial contact.) Then these two objects are passed to the further check (see flowchart in figure 6):

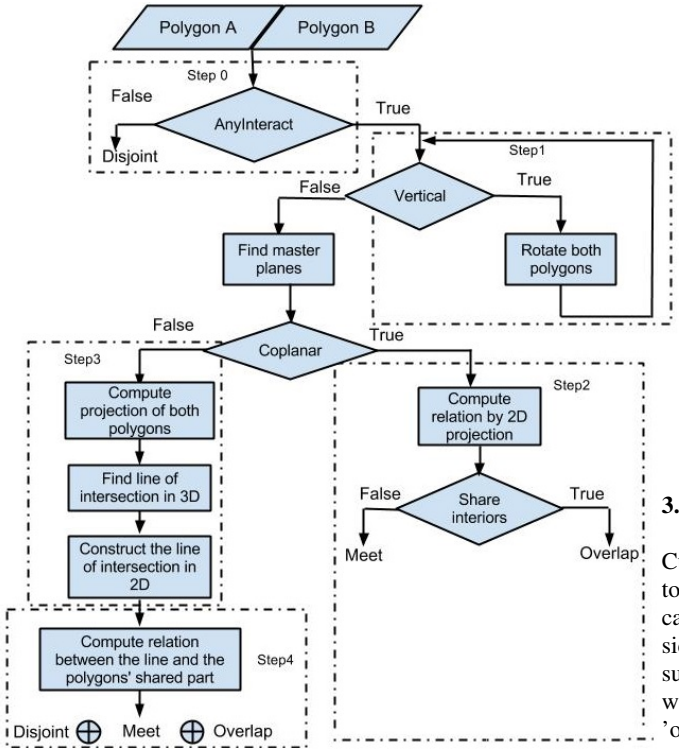


Figure 6: Flowchart of the algorithm determining topological relationships of interests between two 3D planar polygons.

- Step 1.

  Check if either polygon is vertical.

  If yes, rotate both in 3D space until neither is vertical. (The rotational matrix can be accessed through http://inside.mines.edu/ gmurray/ArbitraryAxisRotation/, section 6.2. In our research, value of the rotational angle θ is 10°).

- Step 2.

  Calculate the parameters of plane each polygon belongs to and see if planes are parallel. If yes, then calculate the relation their projections in 2D (x,y) plane have (using SDO_GEOM.RELATE 2D available masks). If the result string indicates that two interiors have common part, be it EQUAL or CONTAINS or COVERS or COVEREDBY or INSIDE or OVERLAPBDYINTERSECT, then two polygons overlap.

- Step 3.

  If two polygons are not parallel or vertical, calculate the parameters of line of intersection where the master planes meet. Then choose two points on this line to make up a 3D line. If the end point of the line would be intersected, there will be more topological relationship strings returned by SDO_GEOM.RELATE that do distract the discernment.

  To keep the computation simple, the line is made to be infinite to the polygons. Thus the end points are selected from a broader extend, that is, double-sized bounding rectangle.

- Step 4.

  Project polygon A and polygon B in 3D, and line of intersection from two master planes to 2D (x,y) plane. Then we get 2D polygons A_proj, B_proj and line Li2D. A_proj and B_proj will share a common geometry, which can be polygon(s) or line(s) or point(s).

  Use Spatial operator SDO_Intersection to retrieve the common geometry (for instance, if A_proj and B_proj overlap, it returns the overlapping region; if they meet, it returns the shared line(s)/point(s); if they disjoin, it returns empty geometry 'NULL').

  Check the relation Li2D has with the common geometry.

  If the common geometry shares a part on the interiors of both A_proj and B_proj, and is crossed by Li2D, then two polygons in 3D 'overlap' (see example figure 3.1).

  If Li2D does not have any contact with either of the interiors, but has contact with the boundary of A_proj and/or B_proj, then A and B meet (example figure 4(b)).

  If Li2D disjoints the common geometry then A and B disjoin (this is designed specifically to avoid the error resulted by rings as mentioned earlier).

3.2 Identify Relations Between Two Polyhedrons

Currently only SDO_ANYINTERACT and SDO_INSIDE are able to check binary topological relationship for polyhedrons. They can determine if two polyhedrons are disjoint, or if one is inside the other. But they cannot distinguish between relationships such as meet, cover/coveredBy, overlap/intersect and equal. Here we propose our own approach to distinguish between 'meet' and 'overlap' in binary topological relationship of polyhedrons.

The topological relationship between two polyhedrons, in our research, is determined on the polygon level. An individual polyhedron is treated as an composite of a set of planar polygons. Two polyhedrons thus have two sets of planar polygons. Assuming that a polygon pair includes a polygon from polyhedron A, and the other from polyhedron B, the operator 3D_InteriorInteract, as we described in last section, is run recursively against all such pairs of polygons. Therefore if polyhedron A is made up of N polygons and polyhedron B M polygons, the total amount of running the operator will be N x M times and thus produce N x M

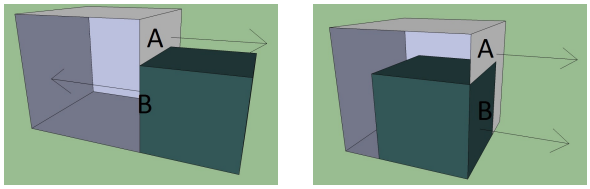
polygonal relationships. Finally, a summary of these relationships will help us determine the actual topological relationship between polyhedron A and B.

An adjustment to apply the operator to the polyhedron-relationship check is added to the method in step 2. Two overlapping polygons must share a part of or the whole face. Depending on polygons' orientations, their master polyhedrons may have different topological relations that are of our interests (see figure 7 for the illustration in 3D) One is marked as 'overlap':

$$\text{R(A, B)} = \begin{bmatrix} - & iB & bB & eB \\ iA & 1 & 1 & 1 \\ bA & 0 & 1 & 1 \\ eA & 0 & 0 & 1 \end{bmatrix}$$

and the other as 'meet':

$$\text{R(A, B)} = \begin{bmatrix} - & iB & bB & eB \\ iA & 0 & 0 & 1 \\ bA & 0 & 1 & 1 \\ eA & 1 & 1 & 1 \end{bmatrix}$$



(a) Orientations of polygon A and polygon B being opposite indicates that two master polyhedrons 'meet'.

(b) Orientations of polygon A and polygon B being the same indicates that two master polyhedrons 'overlap'.

Figure 7: The orientation of two overlapping polygons is used to tell relationships of their master polyhedrons.

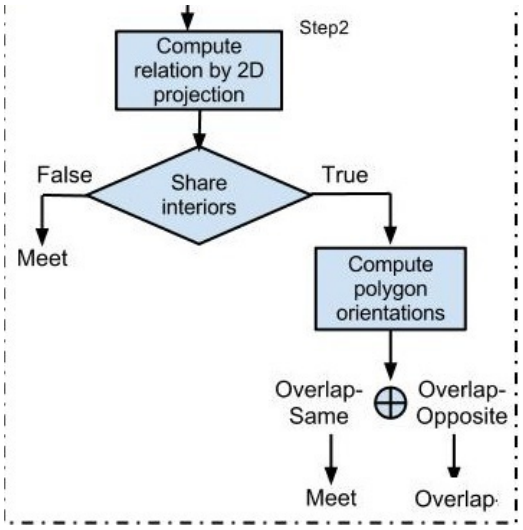


Figure 8: Extend step 2 of figure 6 to check the orientation of two overlapping polygons.

The detail of the extended method *PolyhedronInteract* from figure 8 is explained below (the original script is accessible via http://www.sourcepod.com/vupmsy73-19781). This method may also be employed to look for the shared face between adjacent 3D buildings that are discussed in (Ellul, 2013).

- Step 1. Select a random pair of polygons with one from polyhedron A and the other from polyhedron B as input.
- Step 2. Check the pair with 3D_InteriorInteract.
- Step 3. Count the number of times a relation (disjoint, overlap or meet) occurs and add the occurrence into three different (counters) variables.
- Step 4. Loop from Step 1 or, if all pairs are calculated, go to Step 5.
- Step 5. If counter_intersect is 1 or more than 1, then two polyhedrons overlap. If counter_ disjoint = number of all pairs (i.e. all pairs are disjoint), then go to step 6. If no pair intersects, and not all pairs disjoint, then the relation is meet.
- Step 6. Use SDO_INSIDE to check relations between these two polyhedrons, if neither is inside the other, then these two polyhedrons are disjoint. If either is inside the other, then these two polyhedrons overlap.

A situation that this method will have problem with is when one polyhedron 'contains' the other, and the 'meet' occurs on the boundary (see figure 9).

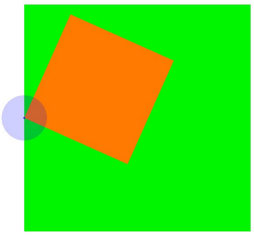
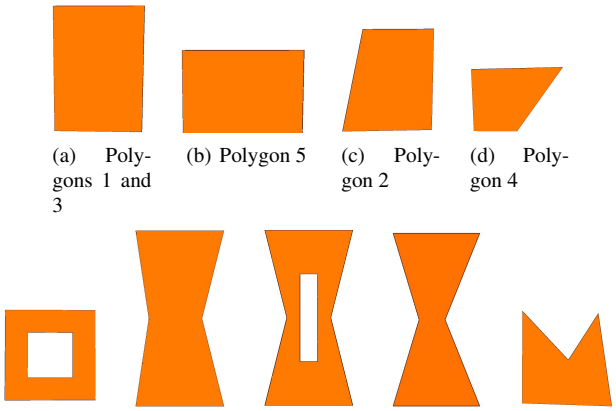


Figure 9: A 2D view of a 3D problematic case where one polyhedron contains the other. And the touching occurs on the boundary from within. The method described in section 3.2 will detect this case as 'meet'.

4 RESULT AND TEST

A group of 3D planar polygons are used to test the performance of 3D_InteriorInteract, including special cases as concave polygons and rings (see figure 10).



(a) Polygons 1 and 3

(b) Polygon 5

(c) Polygon 2

(d) Polygon 4

(e) Polygons 7 and 9

(f) Polygon 8

(g) Polygon 10

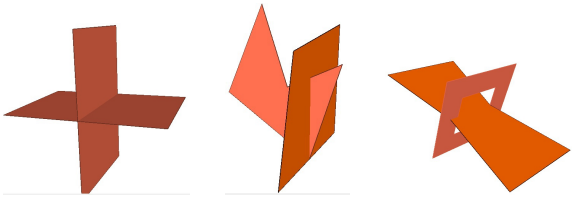
(h) Polygon 11

(i) Polygon 15

Figure 10: 3D planar polygons used in the test. Some polygons are of the same shape but at different locations of the space.

A collection of polygon pairs that are calculated as 'overlap' is illustrated in figure 11. And 'meet' cases are given in figure 12. An example SQL query is given below (see also figure 11(c)):

SELECT polygons3d relation(p1.geometry, p2.geometry)
FROM polygons3d p1, polygons3d p2
WHERE p1.id=7 AND p2.id=8

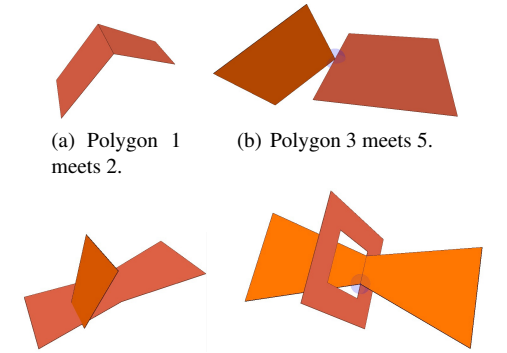


(a) Polygon 1 overlaps 3.

(b) Polygon 4 overlaps 15.

(c) Polygon 7 overlaps 8. Overlapping occurs on inner boundary of the ring.

Figure 11: Polygon pairs whose topological relationships are calculated as 'overlap'.



(a) Polygon 1 meets 2.

(b) Polygon 3 meets 5.

(c) Polygon 4 meets 8.

(d) Polygon 7 meets 10.

(e) Polygon 10 meets 13.

(f) Polygon 13 meets 14.

(g) Polygon 13 meets 15.

Figure 12: Polygon pairs whose topological relationships are calculated as 'meet'.

According to matrices from section 3.1, ANYINTERACT of Oracle Spatial can distinguish disjoint from the non-disjoint (the rest) on polygons without any hole. As a result, our 3D_InteriorInteract can distinguish between 'meet' and 'overlap' as presented in section 3.1. As was presented in section 3.2, the orientation of overlapping polygons indicate whether or not two master polyhedrons overlap.

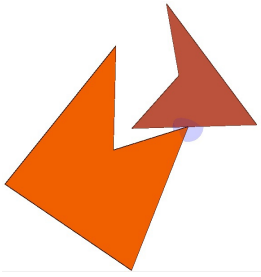


Figure 13: A problematic case where two polygons meet at a point that is on the boundary of both. However their relation is calculated as 'overlap' according to our algorithm.

Test of polygon 14 and polygon 15 (see figure 13) shows that the two meeting polygons are mistaken as 'overlap'.

SELECT polygons3d_relation(p1.geometry, p2.geometry)
FROM polygons3d p1, polygons3d p2
WHERE p1.id=14 AND p2.id=15;

POLYGONS3D 3D_InteriorInteract(P1.GEOMETRY, P2.GEOMETRY)

Overlap

With a zoom-in examination, this inappropriate result appears to be affected by the collection geometry type and specification of Oracle regarding topological relationships. The common geometry (geometric intersection) in 2D of these two polygons is a heterogeneous geometry - a collection of a point and a polygon (see figure 14). Actually the point part is on the intersection line (in 2D), but the polygon part is disjoint from the line. The Oracle operator SDO_GEOM.RELATE sees the topological relationship between the line and the heterogeneous geometry as 'OVERLAPBDYDISJOINT' which according to (Oracle, 2010) means 'The interior of one object intersects the boundary and interior of the other object, but two boundaries do not intersect'.

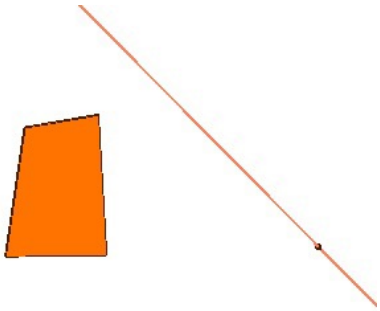


Figure 14: The 2D common geometry derived from two polygons in figure 13 after rotation. It has a polygon part, and a point part which is on the line.

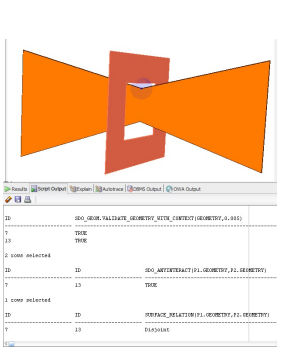
In other words, the point that is on the line object in figure 14 is considered to be both boundary and interior of the collection geometry by Oracle. However, to our understanding, this point is only a boundary. Therefore the line and the collection object are expected to hold the relation 'meet'. The definition of topological relationships concerning the collection geometry type is neither clear in the Oracle document, nor in 9IM.

Since concave polygons and holey polygons often give unexpected result in calculation of topological relationships, we also test our 3D operator against such polygons (given that their geometries are valid according to the definition of Oracle Spatial). In two cases our operator works properly while ANYINTERACT of Oracle Spatial does not (see figure 15).

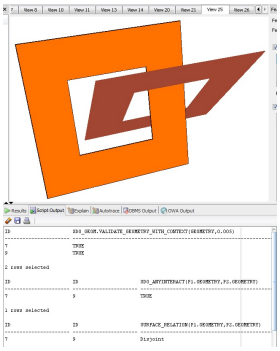
One has a concave polygon going through a ring without any contact. The other has two rings binding each other and is also without any contact. Both cases should be recognised as 'disjoint' as our operator shows. But ANYINTERACT returns 'non-disjoint'.

5 CONCLUSION AND FUTURE WORK

In this paper we present an approach to distinguish 9I topological relationships between two 3D planar polygons, 'overlap' and 'meet'. The concept is then implemented with the help of existed 2D operators in Oracle Spatial 11g. And with small extension, the approach can be applied to distinguish the same relationships between two polyhedrons. We believe the approach of using 2D



(a) A concave polygon goes through the hole of the other polygon without any contact.



(b) Two holey polygons (i.e. rings) 'bind' each other without any contact.

Figure 15: Two tests that ANYINTERACT cannot function properly, while our operator returns the the expected value.

functionality and computational geometry will immediately extend 3D functionality of current Geo-DBMSs.

An Oracle Spatial operator was developed as a result of implementing our concept. Both self-made data and real world datasets / objects, as was presented in (Boufidou et al., 2011), are used to test against the performance. We also discover the lacking of clear definition of the topological relationship for collection geometry, in our discussion about concave polygon (see section 3.1). A future work that can exam the applicability of our approach is to implement the operator in other Geo-DBMSs, e.g. PostGIS, since its recent version also supports a check for 3D geometry intersection.

An advantage to detect the topological relationship between polyhedrons by the composing polygons is that in many 3D models, e.g. CityGML, a polyhedron (or body/solid) are stored as a collection of polygons (or faces). Therefore the existing records (polygons) can be immediately used to do the calculation. But the composing of polyhedron (needed in step 6), which may have thousands of polygons/faces involved, would be rather complicated.

An immediate workaround is to use the aggregate minimum bounding box (MBR) which contains all polygons to represent a polyhedron. And the relationship between two polyhedrons will be simplified to relationship between two aggregate MBRs. However, this workaround may give wrong interpretation if one polyhedron is concave.

To conclude, more 3D operators for can be built on the basis of existing 2D and 3D operators. This approach, as we have discussed through this paper, would reduce many efforts and speed up the development of 3D functionality in Geo-DBMS.

REFERENCES

ActianCommunity, 2011. Ingres Geospatial - spatial types online manual. Technical report, http://community.actian.com/wiki/SpatialTypes (last access on August 2013).

Arens, C., Stoter, J. and van Oosterom, P., 2005. Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. Computers & Geosciences 31(2), pp. 165–177.

Billen, R., Zlatanova, S., Mathonet, P. and Boniver, F., 2002. The dimensional model: a framework to distinguish spatial relationships. In: Advances in Spatial Data Handling, Springer, pp. 285–298.

Boguslawski, P., Gold, C. M. and Ledoux, H., 2011. Modelling and analysing 3d buildings with a primal/dual data structure. ISPRS Journal of Photogrammetry and Remote Sensing 66(2), pp. 188–197.

Borrmann, A. and Rank, E., 2009. Topological analysis of 3D building models using a spatial query language. Advanced Engineering Informatics 23(4), pp. 370–385.

Boufidou, E., Commandeur, T., Nedkov, S. and Zlatanova, S., 2011. Measure the Climate, Model the City. In: in: Proceedings of the 28th UDMS, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-4/C21, pp. 59–66.

Brugman, B., Tijssen, T. and van Oosterom, P., 2011. Validating a 3D topological structure of a 3D space partition. In: Advancing Geoinformation Science for a Changing World, Springer, pp. 359–378.

Chen, J., Li, Z., Li, C. and Gold, C., 1998. Describing topological relations with voronoi-based 9-intersection model. International Archives of Photogrammetry and Remote Sensing 32, pp. 99–104.

Clementini, E., Di Felice, P. and van Oosterom, P., 1993. A small set of formal topological relationships suitable for end-user interaction. In: The 3rd International Symposium on Large Spatial, Springer, pp. 277–295.

Clementini, E., Sharma, J. and Egenhofer, M. J., 1994. Modelling topological spatial relations: Strategies for query processing. Computers & graphics 18(6), pp. 815–822.

Egenhofer, M. J., 1995. Topological relations in 3-D. Technical report, National Center for Geographic Information and Analysis and Deparment of Spatial Information Science and Engineering Department of Computer Science university of Maine.

Egenhofer, M. J. and Franzosa, R. D., 1995. On the equivalence of topological relations. International Journal of Geographical Information Systems 9(2), pp. 133–152.

Egenhofer, M. J., Sharma, J. and Mark, D. M., 1993. A critical comparison of the 4-intersection and 9-intersection models for spatial relations: formal analysis. In: AUTOCARTO, Vol. 11, pp. 1–12.

Ellul, C., 2013. Can topological pre-culling of faces improve rendering performance of city models in google earth? In: Progress and New Trends in 3D Geoinformation Sciences, Springer, pp. 133–154.

Ellul, C. and Haklay, M., 2006. Requirements for topology in 3D GIS. Transactions in GIS 10(2), pp. 157–175.

Ellul, C. and Haklay, M. M., 2009. Using a B-rep structure to query 9-intersection topological relationships in 3D GIS–reviewing the approach and improving performance. In: 3D Geoinformation Sciences, Springer, pp. 127–151.

Ghawana, T. and Zlatanova, S., 2010. Data consistency checks for building a 3D model: a case study of Technical University, Delft Campus, The Netherlands. Geospatial World (4).

Guo, W., Zhan, P. and Chen, J., 1998. Topological data modelling for 3D GIS. IAPRS, ISPRS Commission IV Symposium on GIS - Between Visions and Applications 32/4, pp. 657–661.

Haarslev, V. and Möller, R., 1997. Sbox: A qualitative spatial reasonerprogress report. In: 11th International Workshop on Qualitative Reasoning, Cortona, Tuscany, Italy, Citeseer, pp. 105–113.

Herring, J. R., 2010-08-04. OpenGIS Implementation Standard for Geographic information-Simple feature access-Part 2: SQL option. OGC Document.

Hoffmann, C. M., 1989. Geometric and solid modeling: an introduction. Morgan Kaufmann Publishers Inc.

ISO, 2008-09-17. 19107:2003. In: Geographic information – Spatial schema, ISO.

Kazar, B. M., Kothuri, R., van Oosterom, P. and Ravada, S., 2008. On valid and Invalid Three-Dimensional Geometries. Springer, chapter 2, pp. 19–46. Advances in 3D Geoinformation Systems, Lectures Notes in Geoinformation and Cartography.

Khuan, C. T., Abdul-Rahman, A. and Zlatanova, S., 2008. New 3D data type and topological operations for geo-DBMS. Urban and Regional Data Management UDMS 2007 Annual pp. 211–222.

Ledoux, H. and Meijers, M., 2009. Extruding building footprints to create topologically consistent 3d city models. Urban and Regional Data Management, UDMS Annuals pp. 39–48.

McKenney, M. and Schneider, M., 2008. Topological relationships between map geometries. In: Database Systems for Advanced Applications, Springer, pp. 110–125.

Medeiros, C. B. and Andrade, M. J., 1994. Implementing integrity control in active data bases. Journal of Systems Software 27, pp. 171–181.

OGC, 2008. OpenGIS City Geography Markup Language (CityGML) Implementation Specification. 1.0.0 edn. 232 pages.

Oracle, 2010. Oracle Spatial Developer's Guide 11g Release 2 (11.2). 916 pages.

PostGIS 2.0 online Manual, 2012. available at http://postgis.net/2012/12/03/postgis-2-0-2 (last accessed August 2013).

Van Oosterom, P., Ploeger, H., Stoter, J., Thompson, R., Lemmen, C. et al., 2006. Aspects of a 4D cadastre: a first exploration. In: In Proceedings of the XXXII Internatinoal FIG Congress, 23 p.

Xu, D., 2011. MSc thesis in Geomatics, Design and Implementation of Constraints for 3D Spatial Database - Using Climate City Campus Database as an Example. Master's thesis, Delft University of Technology, available at: http://www.gdmc.nl/publications/2011/Constraints_3D_Spatial_Database.pdf (last accessed August 2013).

Zlatanova, S., 2000. On 3D topological relationships. In: In Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA 2000), Greenwich, London, UK, pp. 913–919.