Robustness of Spatial Relation Evaluation in Data Exchange

Alberto Belussi, Sara Migliorini
Dipartimento di Informatica
University of Verona
Strada Le Grazie, 15
37134 Verona (Italy)
alberto.belussi@univr.it

Mauro Negri, Giuseppe Pelagatti
Dipartimento di Elettronica e Informazione
Politecnico of Milan
Via Ponzio, 34/5
20133 Milan (Italy)
giuseppe.pelagatti@polimi.it

ABSTRACT

Topological relationships between geometric objects are important in several spatial applications, like spatial query evaluation, spatial integrity constraints checking, and spatial reasoning. Although the conceptual aspects of topological relationships between geometric objects embedded in the Euclidean space have been extensively studied, the problems arising when topological relationships are evaluated on real data have been much less explored. In particular, robustness problems arise in the evaluation of topological relationships between geometric objects implemented as vectors in a discrete space. A lack of robustness is characterized by the fact that different systems can produce different evaluations of topological relationships on the same data, and it is caused by the fact that coordinates are represented as finite numbers. The goal of this paper is to formally analyze some rules for increasing the robustness of a topological relationship evaluation and to give some examples w.r.t. a specific topological relationship.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management – Database applications – *Spatial databases and GIS*

General Terms

Algorithms, Theory, Verification.

Keywords

Spatial data, topological relations, robustness of computation.

1. INTRODUCTION

Topological relationships are a fundamental formal tool for describing the spatial properties of data in geographical applications: this occurs for example in schema definitions, where spatial integrity constraints have to be defined, but also in query specification, where spatial filters have to be applied, and in updates where relations are used to specify data quality [7,8].

Although many abstract models for defining the semantics of topological relationships between geometric objects embedded in an Euclidean space have been extensively studied [2,3], the problems arising when they are evaluated on real data have been much less explored. In particular, topological relations have been defined by using the 9-intersection matrix approach [2] or other approaches [5], while for their evaluation algorithms of computational geometry have been implemented in real systems working on real data implemented as vectors in a discrete space.

A consequence of this fact is that the evaluation of topological relations can be non robust, i.e. produce different results on the same data in different contexts. The existence of robustness problems in the execution of computational geometry algorithms using finite numbers, instead of the real numbers theoretically required, is well known [4]. In the context of this paper the problems due to the finite number representations used in the implementation of the algorithms is made even worse by the data perturbations due to their exchange between different systems. Such exchanges can introduce perturbations in geometric representation due to the conversions between different formats and precisions. The robustness problem in the evaluation of topological relations depends also from the dimension of the embedding space. For example, consider two curves which have a macroscopic intersection in 2D: in 2D it is assumed that every system evaluates the topological relation to be *Crosses*, but in 3D small differences in the z-value could cause some system to evaluate the relation to *Disjoint* while others would evaluate it to *Crosses*. Therefore, in many cases a distinct analysis of the robustness of topological relations in 2D and 3D is necessary.

Several *robustness rules* have been proposed in order to get rid of these problems, and they are to some extent applied by systems. The most important one is based on the determination of common geometric primitives between different objects. These common geometric primitives are either stored once and referred to by the objects (topological structures [6]) or are repeated identically in all objects which share them. This robustness rule can solve many of the mentioned robustness problems, but not all of them. A complementary robustness rule, which has been suggested for instance in [6], consists in assuring that a minimum distance is kept between all geometric primitives which are not identical.

The goal of this paper is to analyze which robustness rules are sufficient and necessary in order to make robust the evaluation of some topological relations; this analysis is done with respect to some topological relations applied to some the geometric types of the Simple Feature Model (SFM) published by OGC [1], moreover, it is done in 2D and in 3D.

2. REFERENCE MODEL

This paper considers the geometric model recently defined by the OGC as version 1.2 of the Simple Feature Access architecture (SFA) [1]. This model contains classes for describing geometries in the Euclidean spaces R^2 and R^3. The most significant evolution with respect to the previous version is represented by the type *PolyhedralSurface* for representing 3D surfaces as sets of polygon patches with some constraints [1]. The types considered in this paper are: *Point, LineString, Polygon,* and *PolyhedralSurface* together with all their subtypes. Anyway, the obtained results can be easily extended to *MultiGeometries* types.

Topological relations of the Simple Feature Access architecture are not mutually exclusive; hence, for simplifying the study of robustness property a slightly different set of mutually exclusive relations is adopted, called REL_{topo}. The extension of this study to other sets of relations is possible provided that the relations can be expressed as a set of matrices of the 9-intersection model [2]. The set REL_{topo} is made up of the relations: *Disjoint* (DJ), *Touches* (TC), *In* (IN), *Contains* (CT), *Equals* (EQ), *Overlaps* (OV) and *Crosses* (CR) as defined in [3].

REL_{topo} is defined by using the concepts of internal part, boundary and external part of a geometric object. Given a geometric object *a* of the abstract type *Geometry* and the operations: *a.PS*() returning the point set represented by *a*, *a.bnd*() returning the geometric object representing the boundary of *a*, and *a.dim*() returning the dimension of *a*, the following point sets are defined. The *internal part* of *a*, denoted as *I*(*a*), is the point set *a.PS*() - *a.bnd*().*PS*(), the *boundary* of *a*, denoted as *B*(*a*), is the point set *a.bnd*().*PS*, and the *external part* of *a*, denoted as E(*a*), is the point set *a.space*() - *a.PS*().

In a discrete vector model each geometry is described as a set of vertices embedded in a discrete space. A vertex is represented as a tuple of coordinates, namely by two or three real numbers encoded using a discrete approach, like the floating point model. In the sequel, these numbers are denoted as *finite numbers*. The discrete representation of a geometry *g* is denoted as *DR*(*g*).

Definition 2. [Vertex] A *vertex ver* is a tuple of finite numbers representing a 2D or 3D coordinate: *ver* = (*x*,*y*), or *ver* =(*x*,*y*,*z*).

The discrete representation of a *Point* corresponds to a single vertex, while *LineString*, *Polygon* and *PolyhedralSurface* are described as sets of vertices among which a linear interpolation method is applied between consecutive vertices.

The definition of the discrete representation of a *LineString*, *Polygon*, and *PolyhedralSurface* can be simplified by using the concepts of *segment*, *ring*, and *patch* defined below.

Definition 3. [Segment, boundary ring and patch] Let (ver_1, ver_2) a pair of vertices, a *segment* is the linear interpolation between them. Let (ver_1,...,ver_n) be a list of vertices, its linear interpolation is a *ring* if and only if ver_1=ver_n, namely it is a cycle. A *patch* is a planar polygon whose boundary is defined by a planar ring, i.e (ver_1,...,ver_n).

The following operations are used in the remainder of this paper:

- *real a.dist*(*b*): it returns the distance between *a* and *b*.
- *boolean a.eq*(*b*): it tests for equality two geometries of the same type represented in a discrete form. In particular, two vertices are equal if they are bitwise identical..
- *vertex seg.s*()(e()): it returns the start(end) point of *seg*.
- *set*(*vertex*) *seg.bnd*(): it returns the set of end points of the segment s. (*seg.bnd*() ≡ { *seg.s*(), *seg.e*() })
- *segment* $s_1 \cup \ldots \cup s_n$: it joins two overlapping (or touching) segments that lie on the same straight line, if the segments are disjoint or not collinear, it returns the empty geometry.
- *patch* $p_1 \cup \ldots \cup p_n$: it joins two overlapping (or touching) patches that lie on the same plane, if the patches are disjoint or not coplanar, it returns the empty geometry.
- *boolean a.int*(*b*): it tests the intersection between the interior of two geometries: $I(a) \cap I(b) \neq \emptyset$. When *a* and *b* are segments, it requires that the intersection is a point ($dim(I(a) \cap I(b))=0$); if the intersection has dimension 1, it returns false.
- *boolean a.cnt*(*b*): it is a test of containment between two geometries interiors: $I(b) \subset I(a)$.
- *set*(*segment*) *DR*(*psur*).*bnd*(): it returns the set of segments representing the boundary of *psur*.
- *set*(*segment*) *DR*(*psur*).*intSeg*(): it returns the set of segments belonging to a patch of *psur* that are not in *DR*(*psur*).*bnd*().

3. System and Implementation Assumptions

Let us consider two systems, denoted here as *S* (source) and *D* (destination), which exchange spatial data and evaluate topological relations on the exchanged data. During the transfer a transformation may occur on each vertex, let ver_S be a vertex in *S* and ver_D the same vertex in *D*, the transformation from ver_S to ver_D is captured by a mapping *F*(), such that $ver_D = F(ver_S)$.

The behavior of current systems displays a set of problems with respect to robustness. In the sequel they are listed and for each one some minimal assumptions on the systems behavior are stated which are necessary preconditions for building robustness rules.

Problem 1. [Perturbation in data exchange] The data transfer between two systems *S* and *D* may cause a perturbation.

In order to deal with this problem, the following minimal assumption has to be stated.

Assumption 1. [Locality of perturbation in data exchange] The perturbation introduced in a vertex representation by the mapping *F*() has an upper bound, called PUB (Perturbation Upper Bound). In other words, given a vertex *ver*, the distance between *ver* and *F*(*ver*) is always less than PUB:

$$\forall ver \in \text{vertex}\ (ver.dist(F(ver)) < \text{PUB})$$

If a sequence of *n* data transfers is considered instead of a single one, then it is also assumed that the combined effect of all perturbations cannot diverge:

$$\forall ver \in \text{vertex}\ (ver.dist(F_n(\ldots(F_1(ver)))) < \text{PUB})$$

Problem 2. [Uncertainty in the evaluation of relative positions] Another aspect that introduces ambiguity in topological relation evaluation is due to the necessity of implementing a system of linear equations in order to evaluate some *elementary geometric operations*. These elementary geometric operations are the basis of all vector predicates that are needed in topological relations evaluations. They are ambiguous due to the finite precision of discrete geometric representation and algorithm implementation.

In particular, the elementary operations which test the following spatial relationships can cause ambiguous results:

Problem 2.1 [Vertex-Vertex equality/disjunction] In some cases different systems return different results if the two vertexes are close to each other.

Problem 2.2 [Vertex-Segment relative position] If the distance between a vertex *ver* and a segment *seg* is very small, then different systems can return different results. The relative position between an (oriented) segment *seg* and a closed vertex *ver* in a 2D/3D space can be: (i) *seg* contains *ver*; (ii) *ver* is on the left of *seg*; (iii) *ver* is on the right of *seg*. Left and right refers to the unique plane that contains both *seg* and *ver*. For example, *seg* can contain *ver* and *F*(*ver*) might be on the left of *F*(*seg*) or *ver* can be on the right of *seg* and *F*(*ver*) on the left of *F*(*seg*).

Problem 2.3 [Segment-Segment intersection/disjunction in 3D] In some cases different systems return different results if the two segments are close to each other.

Segment-segment intersection in 2D is not an independent elementary problem, since it can be reduced to problem 2.2.

Problem 2.4 [Vertex-Patch relative position] If the distance between a vertex *ver* and a patch *pat* is very small, then different systems can return different results. The relative position between a vertex *ver* and an (oriented) patch *pat* in the 3D space can be: (i) *pat* contains the vertex; (ii) *ver* is above *pat*; (iii) *ver* is below *pat*

In order to introduce robustness in topological relations evaluation, the following assumptions are introduced.

Assumption 2. [Equality preservation in data exchange] The mapping *F*() representing the exchange of geometry between a source system *S* and a destination system *D* is a function, namely: $\forall ver_1, ver_2 \in \text{vertex}\ (ver_1.eq(ver_2) \Rightarrow F(ver_1).eq(F(ver_2)))$

Assumption 3. [Minimum distance for preserving relative positions] Given two geometries transferred from system *S* to system *D*, the evaluation of their relative position in the two systems is consistent if their distance is greater than a lower bound, called threshold distance TD. In particular:

Assumption 3.1 [Vertex-Vertex minimum distance]
$\forall ver \in \text{vertex}(\forall ver_1 \in \text{vertexes}, (ver.dist(ver_1) > \text{TD} \Rightarrow$
$(\neg ver.eq(ver_1) \Rightarrow \neg F(ver).eq(F(ver_1)))$

Assumption 3.2 [Vertex-Segment minimum distance]
$\forall ver \in \text{vertex}\ (\forall seg \in \text{segment}\ (ver.dist(seg) > \text{TD} \Rightarrow$
$((ver \text{ is on the left of } seg) \wedge (F(ver) \text{is on the left of } F(seg)\ ) \vee$
$(ver \text{ is on the right of } seg) \wedge (F(ver) \text{ is on the right of } F(seg)))))$

Assumption 3.3 [Segment-Segment 3D minimum distance]
$\forall seg \in \text{segment3D}\ (\forall seg_1 \in \text{segment3D}\ (seg.dist(seg_1) > \text{TD} \Rightarrow$
$(\neg seg.int(seg_1) \Rightarrow \neg F(seg).int(F(seg_1)))$

Assumption 3.4 [Vertex-Patch minimum distance]
$\forall ver \in \text{vertex}\ (\forall pat \in \text{patches}\ (ver.dist(pat) > \text{TD} \Rightarrow$
$((ver \text{ is above } pat) \wedge (F(ver) \text{ is above } F(pat)) \vee$
$(ver \text{ is below } pat) \wedge (F(ver) \text{ is below } F(pat)))))$

Notice that with the precision levels of current systems, TD is significantly smaller than the average error in the coordinate representation with respect to real positions of points: TD << Absolute-Positional-Error. Normally, if no specific robustness rules are applied, in real datasets it is not correct to assume that TD is the minimum distance among the represented vertexes, segments and patches. In the sequel robustness rules are defined based on the above assumptions.

4. Robustness of Vector Predicates

The implementation of topological relations on the presented discrete vector model makes use of a set of vector predicates, which include in their implementation the elementary operations described in Sec. 3. For the reasons highlighted above, some of these predicates can return different results in different systems, and are called here *critical vector predicates*. The following definition introduces the main six critical vector predicates which are necessary in the implementation of topological relations (the proof of this claim cannot be shown here due to space restriction).

Definition 7 [Basic predicates] The following six basic vector predicates are used in topological relations implementation:

P1. $ver_1.eq(ver_2)$: *equal* predicate applied to a pair of vertexes.
P2. $seg.cnt(ver)$: *cnt* predicate applied to a segment and a vertex.
P3. $seg_1.int(seg_2)$: *int* predicate applied to a pair of segments.
P4. $pat.cnt(ver)$: *cnt* predicate applied to a patch and a vertex.
P5. $pat.int(seg)$: *int* predicate applied to a patch and a segment.
P6. $pat_1.int(pat_2)$: *int* predicate applied to a pair of patches.

4.1 Robustness Rules for Critical Predicates

This subsection introduces a set of rules that, applied on the source system *S*, make the identified critical vector predicates (Def. 7) non ambiguous. These rules are classified into two categories: identity rules and minimum distance rules. The following *identity rules* are used to ensure the robustness of topological relation evaluation and are based on Asmp. 2.

Rule 1. [IR1: Vertex-Segment] For each vertex *ver* that has to lie in a segment $seg = (ver_1, ver_2)$, a new vertex ver_h bitwise identical to *ver* has to be introduced in the *seg* representation splitting it into two new segments $seg_1 = (ver_1, ver_h)$ and $seg_2 = (ver_h, ver_2)$. Therefore, after the rule IR1 has been applied, the following condition is true: $\forall ver \in S\ (\forall seg \in S\ (\neg seg.cnt(ver))))$

Rule 2. [IR2: Segment-Segment] All intersections between two segments seg_1 and seg_2 must be represented by splitting them in four through the insertion of a new common vertex, which has to be represented only once or has to be represented by means of several identical instances, namely instances that have bitwise identical coordinates. Therefore, after the rule IR2 has been applied, the following condition is true: $\forall seg_1 \in S\ (\forall seg_2 \in S$ $(\neg seg_1.eq(seg_2) \Rightarrow \neg seg_1.int(seg_2)))$

Rule 3. [IR3: Vertex-Patch] All vertex-patch intersections must be represented by a vertex *ver* contained in the patch definition. Therefore, the patch representation has to be split in two and the vertex *ver* has to be inserted as a new start/end point of a new segment composing one of the patch boundaries. Therefore, after the rule IR3 has been applied, the following condition is true: $\forall ver \in S\ (\forall pat \in S\ (\neg pat.cnt(ver)))$

These rules are not sufficient alone to guarantee robustness for all critical predicates. In particular, in order to solve the situation highlighted in Problems 2.1, 2.2, 2.3 and 2.4 and given Asmp. 1 and 3, the following other rules are introduced.

Rule 4. [DR0: Vertex-Vertex] For all pairs of vertices ver_1, ver_2 of *S*, the distance between them is either zero or is greater than a Minimum Granted Distance (MGD); MGD is chosen in order to satisfy Asmp. 3 in all involved systems (for example, it could be the maximum TD): $\forall ver_1 \in S\ (\forall ver_2 \in S\ (ver_1.dist(ver_2) = 0 \vee ver_1.dist(ver_2) > \text{MGD}))$

Rule 5. [DR1: Vertex-Segment] For all vertices *ver* and for all segments *seg* of *S*, the distance between them is either zero or is greater than MGD: $\forall ver \in S\ (\forall seg \in S\ (seg.dist(ver) = 0 \vee seg.dist(ver) > \text{MGD}))$.

Rule 6. [DR2: Segment-Segment] For all pairs of distinct segments seg_1, seg_2 of *S*, the distance between them is either zero or is greater than MGD: $\forall seg_1 \in S\ (\forall seg_2 \in S\ (seg_1.dist(seg_2) = 0 \vee seg_2.dist(seg_2) > \text{MGD}))$

Rule 7. [DR3: Vertex-Patch] For all vertices *ver* (representing isolated points or segment end points) and for all patches *pat* of *S*, the distance between them is either zero or is greater than MGD: $\forall ver \in S\ (\forall pat \in S\ (pat.dist(ver) = 0 \vee pat.dist(ver) > \text{MGD}))$

Notice that in 2D spaces DR2 is implied by DR1 and that the minimum granted distance between patches and segments is implied by DR2 and DR3.

Theorem 1. Given Asmp. 1-3 the necessary and sufficient rules to be applied on *S* in order to guarantee the robustness of the critical vector predicates evaluation on *D* are shown in Tab. 1.

Notice that for the mentioned assumptions, when IR1, IR2 and IR3 are applied on *S* the predicates $seg.cnt(ver)$, $seg_1.int(seg_2)$, and $pat.cnt(ver)$ respectively are always false.

Table 1. Robustness conditions for critical vector predicates.

Predicate	in 2D	in 3D
$ver_1.eq(ver_2)$	DR0	DR0
$seg.cnt(ver)$	IR1+DR1	IR1+DR1
$seg_1.int(seg_2)$	IR1+DR1	IR1+IR2+DR2
$pat.cnt(ver)$	IR1+DR1	IR1+IR3+DR3
$pat.int(seg)$ $pat_1.int(pat_2)$	IR1+DR1	IR1+IR2+IR3+DR2+DR3

A derived critical predicate is a predicate which can be expressed in terms of other predicates, at least one of which is critical. A derived critical predicate becomes robust if and only if its constituent predicates become robust. Among all the possible derived predicates, the ones used in Sec. 5 are considered here.

Definition 8 [Derived predicates] The following derived vector predicates are used in topological relations implementation:

DP1. $seg_1.eq(seg_2) \equiv (seg_1.s().eq(seg_2.s()) \wedge seg_1.e().eq(seg_2.e()))$
$\vee (seg_1.s().eq(seg_2.e()) \wedge seg_1.e().eq(seg_2.s()))$.

DP2. $seg_1.in(seg_2) \equiv (seg_1.s().eq(seg_2.s()) \wedge seg_2.cnt(seg_1.e())) \vee$
$(seg_1.s().eq(seg_2.e()) \wedge seg_2.cnt(seg_1.e())) \vee (seg_1.e().eq(seg_2.s())$
$\wedge\, seg_2.cnt(seg_1.s())) \vee (seg_1.e().eq(seg_2.e()) \wedge seg_2.cnt(seg_1.s()))$
$\vee (seg_2.cnt(seg_1.s()) \wedge seg_2.cnt(seg_1.e()))$

DP3. $seg_1.ov(seg_2) \equiv (seg_1.cnt(seg_2.s()) \wedge (seg_2.cnt(seg_1.e()) \vee$
$seg_2.cnt(seg_1.s()))) \vee (seg_1.cnt(seg_2.e()) \wedge (seg_2.cnt(seg_1.e()) \vee$
$seg_2.cnt(seg_1.s())))$

DP4. $pat.cnt(seg) \equiv pat.int(seg) \wedge \forall v \in seg.bnd()(pat.cnt(v)$
$\vee \exists s_i \in pat.bnd()(s_i.cnt(v) \vee \{v\} \cap s_i.bnd() \neq \varnothing))$
$\wedge (\forall s_k \in pat.bnd()(\neg seg.int(s_k) \wedge \neg seg.ov(s_k)) \vee$
$\exists s_{i,1}, \ldots, s_{i,m} \in Segments(seg.eq(s_{2,1} \cup \ldots \cup s_{2,m}) \wedge$
$\forall j \in [1,m](pat.cnt^*(s_{i,j}) \vee \exists s_k \in pat.bnd()(s_{i,j}.eq(s_k) \vee s_{i,j}.in(s_k)))))$

where $pat.cnt^*(seg) \equiv pat.int(seg) \wedge \forall v \in seg.bnd()($
$pat.cnt(v) \vee \exists s_i \in pat.bnd()(s_i.cnt(v) \vee \{v\} \cap s_i.bnd() \neq \varnothing))$
$\wedge\, \forall s_k \in pat.bnd()(\neg seg.int(s_k) \wedge \neg seg.ov(s_k))$

Lemma 1 [Robustness of Derived Critical Vector Predicates] Under the hypothesis of Th. 1, the derived vector predicates of Def. 8 become robust with the application of the rules in Tab. 2.

Table 2. Rules for derived critical predicates robustness.

Derived Predicate	2D space	in 3D space	Basic Predicates
DP1	DR0	DR0	$ver_1.eq(ver_2)$
DP2	IR1+DR1	IR1+DR1	$seg.cnt(ver)$
DP3	IR1+DR1	IR1+IR2+IR3+ DR2+DR3	$pat.int(seg)$, $pat.cnt(ver)$ $seg.cnt(ver)$, $seg.int(seg)$

5. Robustness of Topological Relations

Considering the reference set of topological relations REL_{topo} presented in Sec. 2, this section shows how the corresponding evaluation can be implemented using the predicates of the vector model presented in the Sec. 2. Moreover, according to Th. 1 and Lm. 1 regarding the robustness of the vector predicates, the robustness rules of topological relations evaluation is derived.

A complete treatment of all topological relations cannot be performed here for space restriction. Therefore, this section concentrates only on the *in* relation between a line and a polyhedral surface.

Proposition 2 [*In* L/S]. Given a *LineString* l and a *PolyhedralSurface* s and their discrete representation denoted as ln = DR(l) and ps = DR(s), respectively, the relation $l.in(s)$ = $(l.PS() \cap s.PS() = l.PS()) \wedge (I(l) \cap I(s) \neq \varnothing)$ can be implemented as follows:

$\exists seg \in ln(\exists pat_1, \ldots, pat_m \in ps((pat_1 \cup \ldots \cup pat_m).cnt(seg)) \vee$
$\exists s_1 \ldots, s_m \in ps.intSeg()(seg.in(s_1 \cup \ldots \cup s_m) \vee seg.eq(s_1 \cup \ldots \cup s_m))) \wedge$
$\forall seg \in ln(\exists pat_1, \ldots, pat_m \in ps((pat_1 \cup \ldots \cup pat_m).cnt(seg)) \vee$
$\exists s_1 \ldots, s_m \in ps.bnd()(seg.in(s_1 \cup \ldots \cup s_m) \vee seg.eq(s_1 \cup \ldots \cup s_m)) \vee$
$\exists s_1 \ldots, s_m \in ps.intSeg()(seg.in(s_1 \cup \ldots \cup s_m) \vee seg.eq(s_1 \cup \ldots \cup s_m))))$

Theorem 2 [Robustness of *In* L/S]. The evaluation of the topological relation *in* between a line and a polyhedral surface presented in Prop. 2 is robust if the rules IR1+DR1 are applied in a 2D space, or the rules IR1+IR2+IR3+DR2+DR3 are applied in a 3D space.

6. Conclusion and Future Work

The approach to robustness that has been proposed in this paper is based on a general analysis method that can be applied also for proving the robustness of other topological relations and in different contexts. In particular, the interesting contexts are those which reduce the amount of vertices that have to be created in order to represent a given situation.The first relaxation that can be performed regards the identity rules which can be substituted by maximal distance rules (the distance between two points that have to be snapped is smaller than a certain value D_{max}). The second relaxation regards the planarity of patches, which is a strong requirement. In practice, the only admissible planar patches are triangular, vertical or horizontal patches, or a few other very particular cases. Thus, an almost planar patch can be introduced, that is a patch such that the distance D of all its vertices from a given plane is very small, for instance less than the internal resolution. This will eliminate the need of triangulating all non horizontal or vertical patches.

7. REFERENCES

[1] OGC. OpenGIS Implementation Standard for Geographic Information – Simple Feature Access – Part 1: Common Architecture, version 1.2.1, 2011.

[2] M. J. Egenhofer and R. Franzosa. Point-set Topological Spatial Relations. Int. Journal of GIS, 5(2):161–174, 1991.

[3] E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings of 3rd Int. Symp. on Advances in Spatial Databases*, pages. 277–295, 1993.

[4] J. Hobby. Practical segment intersection with finite precision output. *Comp. Geometry Th. and App.*, 13:199-214, 1999.

[5] Randell, D. A., Cui, Z. and Cohn, A. G. A spatial logic based on regions and connection. *In: Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, Morgan Kaufmann, San Mateo, pages 165–176, 1992.

[6] Thompson, R. and van Oosterom, P. Interchange of Spatial Data-Inhibiting Factors, Agile 2006, Hungary. 2006.

[7] Rodríguez, M.A., Brisaboa, N., Meza, J., and Luaces, M.R. Measuring consistency with respect to topological dependency constraints. In *Proceedings of the 18th ACM SIGSPATIAL Inter. Conf. on Advances in Geographic Information Systems* (GIS '10), pages 182-191, 2010.

[8] Pelagatti, G., Negri, M., Belussi, A., and Migliorini, S. From the conceptual design of spatial constraints to their implementation in real systems. In *Proceedings of the 17th ACM SIGSPATIAL Inter. Conf. on Advances in Geographic Information Systems* (GIS '09), pages 448-451, 2009.