Chapter 14
Modelling and Managing Topology in 3D Geoinformation Systems[1]

Andreas Thomsen, Martin Breunig, Edgar Butwilowski, and Björn Broscheit

Abstract

Modelling and managing topology in 3D GIS is a non-trivial task. The traditional approaches for modelling topological data in 2D GIS cannot be easily extended into higher dimensions. In fact, the topology of real 3D models is much more complex than that of the 2D and 2.5D models used in classical GIS; in consequence there is a great number of different 3D spatial models ranging from constructive solid geometry to boundary representations. The choice of a particular representation is generally driven by the requirements of a given application. Nevertheless, from a data management point of view, it would be useful to provide a general topological model handling 2D, 2.5D and 3D models in a uniform way. In this paper we describe concepts and the realisation of a general approach to modelling and managing topology in a 3D GIS based on oriented d-Generalised Maps and the closely related cell-tuple structures. As an example of the applicability of the approach, the combination of a group of buildings from a 3D city model with the corresponding part of a 2D city is presented. Finally, an outlook to ongoing research is given in the context of topological abstraction for objects represented in multi-representation databases.

Institute for Geoinformatics and Remote Sensing, University of Osnabrück, Seminarstr. 19 a/b, 49069 Osnabrück, Germany
{martin.breunig,andreas.thomsen,edgar.butwilowski,bjoern.broscheit}@uni-osnabrueck.de

[1] This work is funded by the German Research Foundation (DFG) in the project 'MAT' within the DFG joint project 'Abstraction of Geoinformation', grant no. BR 2128/6-1.

14.1 Introduction

Topology and GIS belong together since the development of GIS. Already first GIS like GRASS and Arc/Info provided a topological data model storing relationships between points, lines, and areas of an area network. These traditional approaches for modelling topology in in 2D GIS were implemented by explicit links between geometric objects, e.g. from a line segment to its neighbouring left and right area. The more topologogical relationships the user required, the more complex the topological model became.

Unfortunately, there are no straightforward extensions into 3D space of the 2D topological data models used in traditional GIS. Instead, there is a number of different 3D spatial models ranging from constructive solid geometry to boundary representations, the choice of a particular representation being driven by the requirements of a particular application - from architecture and urbanism to numerical modelling, engineering and underground mining.

In a 3D Geoinformation System, objects of different dimension $d \leq 3$ are processed. The geometry of a geoscientific object in 3D GIS can be composed of sets of points, curves, surfaces and volumes, respectively. In a topology model of a 3D GIS, the components of the objects can be interpreted e.g. as a mesh of nodes, edges, faces and solids that describes both the interior structure of the geoscientific objects and their mutual neighbourhood relationships in 3D space.

To describe topology uniformely in 3D solid modeling [1] and 3D GIS [2], a general framework for topological data models has to be provided that abstracts from the dimension of the objects. Furthermore, it should be usable as a data integration platform for 2D, 3D and time-dependent 3D (sometimes termed "4D") topology.

In this paper, we investigate how oriented G-Maps and cell-tuple structures can be used to handle the topology of a digital spatial model in a more generic way, in order to support 2- and 3-dimensional and spatio-temporal models. For many 3D geo-applications not only the modelling, but also the management of topology in database management systems is relevant. Inspired by the work of GeoToolKit [3], that provides a geometric 3D library, we here present topological data structures and operations needed in 3D GIS.

After a short overview on related work, we recall Lienhardt's d-G-Map in section 3, and discuss basic topological operations in section 4. An object-relational representation based on Brisson's cell-tuple structures is introduced in section 5. The integration of triangular meshes is discussed in section 6, while the application to time-dependent topology is briefly presented in section 7. We conclude with an application example in section 8, and an outlook on future work.

14.2 Related work

Whereas basic relationships of point set topology, and in particular Egenhofer's *nine intersection model* have become standard in GIS, and lend themselves to a 3D generalisation [4, 5], 3D discrete topological structures stemming from algebraic topology have not gained comparable popularity, despite considerable development during the last decades. While [6] discussed the application of simplicial complexes to spatial databases, general approaches to representing topology in the context of 3D modelling have been examined by [7] and by other authors. [8] developed d-dimensional cell-tuple structures, and in parallel [9] developed d-dimensional Generalised Maps (d-G-Maps), to represent and manage the topological properties of cellular partitions of d-dimensional manifolds (d-CPM), cf. also [10]. [11] has shown that 3-G-Maps have comparable space and time behaviour as the DCEL and radial edge structures, but can be used for a wider range of applications, allowing a more concise and robust code. [12] used G-Maps to model architectural complexes in a hierarchy of multi-partitions. G-Maps and cell-tuple structures have been used to represent the topology of land-use changes [13], and are currently applied in the geoscientific 3D modelling software GOCAD[2] [14, 10], and in the topological modelling and visualisation tool Moka[15]. [16] give a concise overview of 2D and 3D topological models and propose the translation into geometric primitives for the integration of 2D and 3D topological models with 3D GIS based on relational databases. Recently, [17] describe the integration of 2D and 3D cadastral objects in a representation by regular polytopes based on pseudo-rational numbers. [18] presents the combination of 3D simplicial networks with Poincarié Algebra in a TEN-based spatial DBMS. Relations of our work with the work of [19], which has not been available at short notice, will be examined in our future work.

14.3 A general approach to modelling topology in 3D GIS

In the following, we use oriented d-CPM as a topological model for 3D GIS. d-CPM can be considered as a generalisation of simplicial complexes, but lack the algebraic properties of the latter. However, if a d-CPM is represented by a d-G-Map, the involution operations of the latter provide the cellular complex with the combinatorial structure of an abstract simplicial complex, where the cells and cell-tuples play the role of abstract nodes and abstract simplexes, while the involution operators define the neighbourhood relationships between the abstract simplexes [9]. Note that the abstract nodes n, e, f, s of a 3-G-Map belong to 4 classes distinguished by different dimensions, whereas

[2] GOCAD is a registered trademark of Earth Decision Co.

all nodes of a simplicial complex belong to the same finite set of vertices in space.

According to [9], a *d-dimensional Generalized Map (d-G-Map)* is a d+2-tuple $G = (D, \alpha_0, \ldots, \alpha_d)$, consisting of a finite set D of objects called "*darts*", and $d+1$ permutations $\alpha_i, i = 0, \ldots, d$ that verify the following two conditions:

the α_i are *involutions*, i.e. they verify for all x,

$$\alpha_i(\alpha_i(x)) = x, \tag{14.1}$$

and for all i, j with $0 \leq i < i+2 \leq j \leq d$, $\alpha_i \alpha_j$ is an involution, i.e.

$$\alpha_i(\alpha_j(\alpha_i(\alpha_j(x)))) = x, \tag{14.2}$$

which implies $\alpha_i \alpha_j = \alpha_j \alpha_i$.

The G-Maps are *embedded* in space by a mapping that to each dart associates a unique combination $(n, e, f[, s])$ of a node n, an edge e, a face f, and in 3D a solid s.

The condition that a d-G-Map always represents a d-dimensional manifold ensures that to any cell-tuple, there exists at most one partner that differs from it only by one exchange operation α_i .

A d-G-Map can be represented as a graph with cell-tuples as nodes (darts), and edges defined by the involution operations (Figure 14.1).

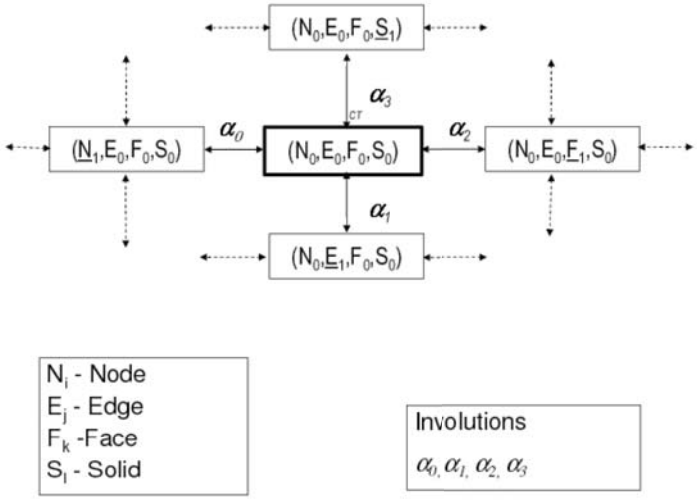

Fig. 14.1 Representation of an oriented 3-G-Map as graph with symmetries determined by the combinatorial character of the involutions.

By assumption, the cellular complexes are orientable, and the corresponding G-Maps are oriented. This implies that there are two classes of darts or cell-tuples of the same cardinality, but carrying different polarity. Different from [9], we exclude the possibility that an involution attaches a cell-tuple to itself ($f(x) = x$), e.g. at the boundary of the cellular complex. Thus we ensure that involution operations always link pairs of cell-tuples of opposite sign. Instead, following [8], we introduce a special non-standard cell, the outside, or *universe*, which in general needs not be simply connected and may comprise holes and islands, and provide the universe also with cell-tuples and involutions on the boundary. This approach increases the number of objects to be handled. However, it simplifies some operations and algorithms.

14.4 Topological operations on oriented d-G-Maps and cell-tuple structures

In the following, we briefly present some topological operations on oriented G-Maps, a more extensive discussion can be found e.g. in [11, 20].

14.4.1 Orbits

Orbits are defined as subsets that can be reached by any combination of involution operations of a given subset of $\alpha_0, \ldots, \alpha_d$, starting from a given dart or cell-tuple cT_0. They are noted $orbit^d(cT_0, \alpha_i, \ldots, \alpha_j)$, or shorter $orbit^d_{i\ldots j}(cT_0)$. Orbits that comprise all indices $0, ..., d$ with the exception of k leave the k-th cell of cell-tuple cT_0 fixed and can be interpreted as the subset of all cell-tuples containing the same cell of dimension k as cT_0. Such orbits can also be noted as $orbit^d(cT_0, \not{k})$. Orbits of this type provide another way to describe cells of dimension k. While most types of orbits are implemented by single or double programming loops of fixed or variable size, [11] implements orbits of type $orbit^2_{012}()$, $orbit^3_{012}()$, $orbit^3_{123}()$, and $orbit^3_{0123}()$ recursively using a stack. Whereas loop implementations of orbits yield continuous closed paths in the G-Map graph, recursively implemented orbits in certain situations may produce discontinuities ("jumps"). Orbits of type $orbit^2_{012}()$, $orbit^3_{0123}()$ produce the complete connected component containing cT_0. Besides orbits, other *loops*, i.e. closed paths in the G-Map graph may be defined by an application or by a user. Orbits and loops are the main methods for the navigation on the G-Map graph. They are also indispensable for the implementation of some of the topological operations discussed below.

14.4.2 Topological operations on cells

Two classes of topological operations can be distinguished: Euler operations that conserve the Euler-Poincaré characteristic and thus the global connectivity properties of a G-Map ([1, 11]), and non-Euler operations that alter the connectivity of the structure. Examples of Euler operations are the subdivision of a cell of any dimension $k > 0$ by a newly created separating cell of dimension $k-1$, e.g. the division of a face f by a new edge e, and the corresponding inverse operations, under certain conditions ensuring the consistency of the resulting G-Map. An example of a Non-Euler operation is the attachment and subsequent *sewing* [11] of two previously disconnected cells, and the inverse operation. These operations constitute the most important methods for the building, transformation and in particular generalisation of d-G-Maps.

14.5 Data management for topological cell-tuple structures

In the following, we discuss some aspects of a different realisation of an oriented d-G-Map, namely as a cell-tuple structure in an object-relational database. This representation aims at providing a general topological access structure in 2D and 3D to existing GIS based on object-relational databases (ORDBMS). Whereas in the graph representation the main attention is given to the involutions α_i, the relational representation uses Brisson's [8] cell-tuples as a realisation of the darts, while involutions are implemented using foreign keys and exchange operations. It is an interesting question, to what extent the functionality of orbits can be replaced by subset queries, join operations, and sorting, i.e. by standard operations of a relational DBMS.

14.5.1 Implementation of the topological data structures as database representation

The topological data structure presented here shall manage the topology of complex spatial objects in 2 and 3 dimensions. It is based on Lienhardt's [9] *d-Generalized Maps* and on Brisson's [8] closely related *cell-tuple structures.*

A d-G-Map can be represented in the relational model as follows (Figure 14.2): the set of cell-tuples is stored in tabular form, e.g. by two relations
$cTpos(node\_id, edge\_id, face\_id[, solid\_id], '+', n\_inv, e\_inv, f\_inv[, s\_inv])$
$cTneg(node\_id, edge\_id, face\_id[, solid\_id], '-', n\_inv, e\_inv, f\_inv[, s\_inv])$,
and the involution operations are modelled as symmetric 1:1 relationships

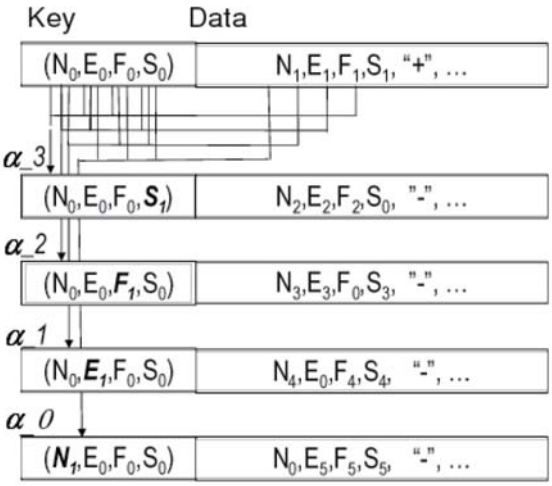

Fig. 14.2 Representation of a 3-G-Map as relation with nodes N_i, edges E_i, faces F_i, solids S_i, and involutions α_i

defined by the *switch operations* [8], linking e.g.
$cTpos(\underline{node\_id}, \ldots, '+', \underline{n\_inv}, \ldots,)$ to $cTneg(\underline{n\_inv}, \ldots, '-', \underline{node\_id}, \ldots,)$.

In a cell-tuple, the combination of cell identifiers, augmented by the positive or negative sign, $(node\_id, edge\_id, face\_is[, solid\_id], sign)$ is used as a unique *cell-tuple key*, while the identifiers of the cells to be exchanged by the involutions are stored as *data*. The data access by cell-tuple keys is enhanced by sorted indexes or hash indexes. The involutions are implemented in two steps: first, from a given cell-tuple entry, create a new cell-tuple key by exchanging exactly one cell_id. Second, use this key to retrieve the corresponding complete entry from the database.

The implementation of a d-G-Map is thus realised as a network of cell-tuples that is made persistent by relations of an Object-Relational Database Management System (ORDBMS). With the goal of a topological component for multi-representation databases [20], we implemented 2-G-Maps and 3-G-Maps with the ORDBMS PostgreSQL[3] [21] in combination with the open source GIS *PostGIS*[4] [22]. In our future work, we intend to implement the graph representation of a G-Map as a topological access structure for our object-oriented 3D/4D Geo-DBMS GeoDB3D [25, 26]. GeoDB3d uses sim-

[3] PostgreSQL ©1996-2005 by the PostgreSQL Global Development Group ©1994 by the Regents of the University of California is released under the BSD license

[4] PostGIS has been developed by Refractions Research and is released under the GNU General Public License

plicial complexes to represent geometry, and is based on the DBMS ObjectStore[5].

14.5.2 Implementation of topological database operations

As the general topological data model is to be integrated into existent spatial ORDBMS, we focus on a clear translation of the G-Map into the relational model, and on the integration of the topological operations with the SQL-commands of a database server. In our view, optimization efforts should rather make use of RDBMS functionality, like sorting, indexing, clustering and caching, than perform the topological operations in client memory.

Orbits of the form $orbit^{d}_{0\ldots\not k\ldots d}(cT_0)$ comprise all celltuples that share with cT_0 a cell of dimension k. The corresponding cell-tuple subset can be retrieved by an appropriate relational query, though not in the same arrangement. For $orbit^{2}_{012}()$ and $orbit^{3}_{0123}()$, a corresponding relational query would yield all celltuples, regardless whether from the same connected component or not. For many purposes, this may be sufficient, but for the implementation of the two last-mentioned orbits, and for applications that require an identical arrangement, we can either explicitly model the orbit using the involution operations, or rearrange the subset on the client side after retrieval. Implementing orbit re-arrangement as an additional functionality of the server would be the best option, if this is supported by the ORDBMS. Loops can be implemented associating each cell-tuple with a selector variable that defines the involution to be performed next.

14.5.3 Example implementation of a database operation

A Basic Euler operation.

As an example of a basic topological operation, in a 2-G-Map comprising nodes $n\ldots$, edges $e\ldots$ and faces $f\ldots$, consider the insertion of a new node n that splits an edge $e(n_0,n_1,f_0,f_1)$ between nodes n_0, n_1 and faces f_0,f_1 into two edges $e_0(n_0,n)$ and $e_1(n,n_1)$ (Figure 14.3). At node n, four cell-tuples are inserted:

- $(n,e_0,f_0,-,n_0,e_1,f_1)$,
- $(n,e_0,f_1,+,n_0,e_1,f_0)$,
- $(n,e_1,f_0,-,n_1,e_0,f_1)$,

[5] ObjectStore is a registered trademark of Progress Software Co.

- $(n, e_1, f_1, +, n_1, e_0, f_0)$.

Eight cell-tuples at nodes n_0 and n_1 are transformed:

- $(n_0, e, f_0, +, n_1, e_a, f_1) \rightarrow (n_0, e_0, f_0, +, n, e_a, f_1)$,
- $(n_0, e, f_1, -, n_1, e_b, f_0) \rightarrow (n_0, e_0, f_1, -, n, e_b, f_0)$,
- $(n_1, e, f_0, -, n_0, e_c, f_1) \rightarrow (n_1, e_1, f_0, -, n, e_c, f_1)$,
- $(n_1, e, f_1, +, n_0, e_d, f_0) \rightarrow (n_1, e_1, f_1, +, n, e_d, f_0)$.
- $(n_0, e_a, f_0, -, \ldots, e, f_1) \rightarrow (n_0, e_a, f_0, -, \ldots, e_0, f_1)$,
- $(n_0, e_b, f_1, +, \ldots, e, f_0) \rightarrow (n_0, e_b, f_1, +, \ldots, e_0, f_0)$,
- $(n_1, e_c, f_0, +, \ldots, e, f_1) \rightarrow (n_1, e_c, f_0, +, \ldots, e_1, f_1)$,
- $(n_1, e_d, f_1, -, \ldots, e, f_0)\,(n_1, e_d, f_1, -, \ldots, e_1, f_0)$.

The translation into SQL is straightforward:

```
BEGIN TRANSACTION
INSERT INTO celltuples VALUE (n,e0,f0,-,n0,e1,f1);
INSERT INTO celltuples VALUE (n,e0,f1,+,n0,e1,f0);
INSERT INTO celltuples VALUE (n,e1,f1,-,n1,e0,f0);
INSERT INTO celltuples VALUE (n,e0,f0,+,n0,e1,f1);
UPDATE celltuples
 CASE
  WHEN edge=e AND node=n0 THEN SET edge=e0 SET node_inv=n
  WHEN edge=e AND node=n1 THEN SET edge=e1 SET node_inv=n
  WHEN edge_inv=e AND node=n0 THEN SET edge_inv=e0
  WHEN edge_inv=e AND node=n1 THEN SET edge_inv=e1
 END
 WHERE edge= e OR edge_inv= e;
COMMIT;
```

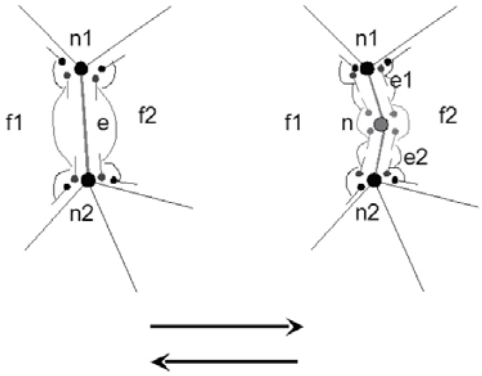

Fig. 14.3 A simple Euler operation: splitting an edge by the insertion of a node

While this example is particularly simple, corresponding operations on cells of higher dimension, e.g. in a 3-G-Map the splitting of a face by the introduction of a separating edge initially follow a similar pattern. However, after the "sewing" i.e. adaptation of the α_i transitions at the new separating edge, additional operations are necessary to modify the links of all cell-tuples that refer the divided face. These operations are supported by two $orbit^2_{01}$ about the two newly created faces. For the splitting of a 3D solid by a new 2D face, a "loop" is required that defines the location where the new face is incident with the boundary of the existing solid to be split. Typically, a database client would provide a set of basic operations for the management, navigation and retrieval of topological information. These operations should be combined into short programs or scripts that fulfil more complex tasks.

14.5.4 Integrity checks for the relational representation of d-G-Maps

A spatial database has no a-priori knowledge about the way a newly introduced dataset has been constructed, nor on the order of update operations executed by a user or by a client application. It is therefore necessary to provide it with a set of tools to check the integrity of a stored G-Map at any time. In a relational representation of a G-Map, join operations can serve to implement some basic integrity checks. A possible test for α_0 verifying condition (1) is the following operation which for a consistent G-Map returns zero:

```
SELECT COUNT(*)
   FROM cT_pos, cT_neg
   WHERE (cT_neg.node_id = cT_pos.node_inv)
     AND NOT (cT_pos.node_id = cT_neg.node_inv);
```

Condition (2) on α_0, α_2 is checked e.g. by the following SQL query involving a triple join, that must return zero, if the G-Map is consistent with condition (2):

```
SELECT COUNT(*)
   FROM cT_pos as cT_p1, cT_neg as cT_n1,
        cT_pos as cT_P2, cT_neg as cT_n2
   WHERE (cT_n1.node_id = cT_p1.node_inv)
     AND (cT_p2.face_id = cT_n1.face_inv)
     AND (cT_n2.node_id= cT_p2.node_inv)
     AND NOT (cT_p1.face_id = cT_n2.face_inv);
```

14.6 Mesh representation of geometry

By merging adjacent d-cells, hierarchies of G-Maps can be built, that consist of a sequence of nested subsets of cell-tuples and their involutions. The α_i transitions at a higher level correspond to a sequence of transitions at the lower, more detailed level. Such a nested hierarchy of G-Maps [11] can be used to integrate a more detailed geometric representation into the topological model. Suppose that the geometry of flat or curved surface patches is represented by triangle nets, which may have been generated by any modelling method, and are described by a set of vertices (v_j, x_j, y_j, z_j) and a set of triangle elements $(tr_i, v_{i_0}, v_{i_1}, v_{i_2}, n_{i_1}, n_{i_2}, n_{i_3})$, where tr_i is an identifier of the triangle element, $v_{i_0}, ..., v_{i_2}$ reference its vertices, and $n_{i_0}, ..., n_{i_2}$ reference the neighbour triangles (Figure 14.4).

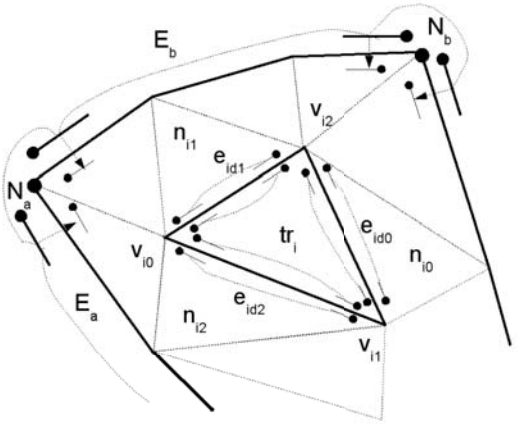

Fig. 14.4 Representation of a face by a triangle mesh. Within each triangle element tr_i, celltuples and switch transitions are generated automatically (small darts). The cell-tuples of the complete face are situated at selected "corner" vertices N_j of the mesh, while involutions follow the mesh boundaries E_k

The corresponding G-Map representation of topology of a mesh element comprises six cell-tuples $(v_{i_k}, edge_l, tr_i, sign)$, where the values $edge_l$ still have to be determined. It is possible to generate a new numbering of edges and a new cell-tuple representation, and to store it as a separate object, but this would greatly increase the size of the model, without adding any new information. In order to save space, we therefore suggest to generate the six cell-tuples of a triangle element on the fly when required, using an edge numbering scheme that can be reproduced as long as the triangle mesh is not altered: If an edge is situated at the triangle mesh boundary, we identify the edge by the triangle identifier tr_i and by its relative position within the

triangle, the latter being a number $p < 3$. For an interior edge separating two triangles tr_i and tr_j, we choose the smaller of the two numbers, e.g. tr_j, and the corresponding relative position p_j. Clearly two bits are sufficient to store the local position, and the edge identifiers may be represented e.g. by $tr_j * 3 + p_j$. By restricting the admissible number of triangles, we can store triangle identifiers, vertex identifiers and edge numbers in fixed length fields, e.g. as long integers, and reproduce the corresponding cell-tuples at any time: for a given triangle tr and its three neighbours tr_0, tr_1 and tr_2, determine first which neighbours have lower id, and second, the relative position p_e of the corresponding edge e within, then compose tr or tr_k and p_e into the edge number e_id; finally, return the six cell-tuples, where the signs '+' and '-' are only given as an example:

$$
\begin{aligned}
&(v_{i_0}, e_{id_2}, tr_i, +)(v_{i_1}, e_{id_1}, tr_2)\\
&(v_{i_1}, e_{id_2}, tr_i, -)(v_{i_0}, e_{id_0}, tr_2)\\
&(v_{i_1}, e_{id_0}, tr_i, +)(v_{i_2}, e_{id_2}, tr_0)\\
&(v_{i_2}, e_{id_0}, tr_i, -)(v_{i_1}, e_{id_1}, tr_0)\\
&(v_{i_2}, e_{id_1}, tr_i, +)(v_{i_0}, e_{id_0}, tr_1)\\
&(v_{i_0}, e_{id_1}, tr_i, -)(v_{i_2}, e_{id_2}, tr_1)
\end{aligned}
$$

As long as the triangle mesh topology is not altered, these cell-tuples can be reproduced at any time.

Thus, at the lowest, most detailed level of the hierarchy, the cell-tuple structure is represented implicitly by the triangle net. By a similar argument, any mesh consisting of elements with a bounded number of vertices can be integrated at the cost of a small number of bits for each element, e.g. a quadrangular mesh for boundary representation, or a tetrahedral mesh for solid modelling. Combined with a corresponding interpolation method, e.g. linear or bilinear interpolation for each mesh cell, at the lowest level topology and geometry representation can be integrated with the G-Map hierarchy.

14.7 Time-dependent topology

The objects of a city model, or of any other 3D GIS possess a "life span", i.e. a temporal interval of existence, that in turn can be decomposed into several intervals during which their structure is constant. As the G-Map representation of topology is a discrete structure, we consider intervals of constant topology as the smallest temporal units, separated by time instants at which the topology changes. Geometry and thematic properties, however may vary within these intervals, and using appropriate interpolation methods continu-

ous variation of geometry and thematic properties at interval boundaries can be modelled.

We define a time-dependent d-G-Map as an application ϕ that to any temporal instant t of a temporal interval T attaches a d-G-Map $\phi(t) = G(D(t), \alpha_0(t), \ldots, \alpha_d(t))$. At each time instant t, $\phi(t)$ must verify the conditions (1) and (2) mentioned above. Given a sequence $t_0, T_1, ..., t_i - 1, Ti, ..., t_n$ of time interval composing a "life span" $[t_0, ..., t_n]$, we require $\phi(t)$ to be constant on each interval T_i. We do not in general impose continuity at the temporal interval boundaries t_i, but in some cases, a smoothness criterion at the transition between consecutive time intervals may be required. This can be achieved e.g. by associating with the time instant t_i the common refinement of the topologies associated with time intervals T_i and T_{i+1} meeting at t_i. As a time instant or time inteval is attached to any component of a time-dependent G-Map, we may search, for a given cell-tuple or a given transition, to find a minimal subdivision of its "life-span". This can result in a bundle of a large number of different temporal interval sequences that may be more difficult to manage than their common subdivision. For the modelling of time-dependent d-G-Maps, we therefore propose a compromise between both approaches, by identifying larger groups of spatiotemporal elements, the so-called spatiotemporal components, that consist of the cartesian product of a temporal sequence with a constant subset of the time-dependent cellular complex. With each spatio-temporal component, a single sequence of time intervals is associated, thus reducing significantly the amount of storage required, while simplifying the management. This approach results in a hierarchical decomposition of the spatio-temporal G-Map into a number of component G-Maps, each of which is constant over its temporal interval of definition. The retrieval of a spatio-temporal cell then proceeds in two steps: first identify the temporal segment and the attached spatio-temporal component, and second, locate the cell within the ST-component.

14.8 Application example: combination of a 2D map with part of a 3D city model

In an ongoing application study, the first results of which are documented in [23], we examine the combination of 2D topology from a cadastral map of the city centre of Osnabrück with freely available 3D city model data of Osnabrück ([24] into a topological model of the environment of Osnabrück palace 14.5).

We used PostgreSQL ([21] as database platform, java[6] and pl/java [27] as programming languages, and the program Moka [15] as visualisation and editing tool. The cell-tuple structure, the involution operations, orbits and

[6] Java is a registered trademark of Sun Microsystems, Inc.

loops, as well basic Euler non-Euler operations on cellular complexes as described in [20] have been implemented. A certain gain in execution speed was achieved by server-side implementation as stored procedures [23], other optimisation attempts are still under way.

The application example consists of several connected buildings enclosing a courtyard, which is linked to streets and to a park by seven archways - a configuration which cannot be modelled without true 3D topology (14.5).

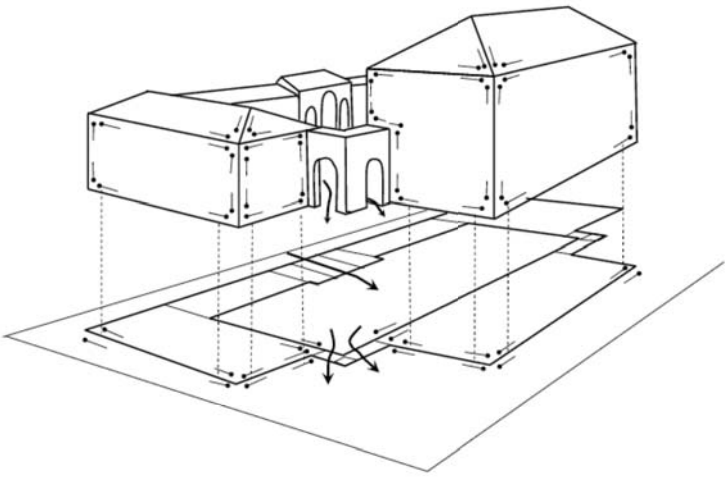

Fig. 14.5 Topological representation (sketch): 3-G-Map of Osnabrück Palace with the 2D city map. Small pins symbolize a subset of the cell-tuples

From a database point of view, the goal is to provide topological and mixed database queries supported by a topological access structure. The queries contain adjacency queries and other useful queries for way finding, e.g. *Can I pass through the courtyard on my way from the street to the park?*.

The use of G-Maps respectively of cell-tuple structures leads to a clear method:

1. Select a rectangular working area in the 2D cadastral map, and a set of buildings from the city model.
2. Correct the location of the vertices using digital elevation data.
3. Extract the topology of the 2D cadastral map as a 2-G-Map.
4. Convert the 2-G-Map of the working area into a 3-G-map:
    a. Extend the relational representation by addition of two columns.
    b. Duplicate the set of cell-tuples, inverting the orientation such that a "lower" and an "upper" side can be distinguished.

   c. Add four nodes, eight edges and five faces, and introduce a solid - the "underground", resulting in a "sandbox" that carries the oroginal 2D map as upper surface.

5. Construct a 3-G-Map from the data of the 3D city model, which in fact is composed of 2D patches in 3D space:
   a. Extract simply connected surface patches, and represent them as faces, edges and nodes in the database.
   b. Build a 3-G-Map by composing the faces into boundaries of volume cells representing the topology of individual building parts.
   c. Transform the 3-G-Map by merging the common boundaries of adjacent building parts, representing the topology of the 3D city map.
6. Combine the models by defining faces on top of the "sandbox" corresponding to the ground faces of the solids composing the city model.
   a. Edit the two topology models to define the faces, edges and nodes to be matched
   b. Correct of vertex positions if necessary.
   c. Sew the cells of the two models at the contacts.

Most of these steps are either trivial or can easily be automated. The constructions of the topologies of the 2D map (step 3.) and of the 3D city model (step 5.), however are not simple. As the 2D cadastral map data are stored as polygons in a shape file, a spatial join on vertex and arc locations has to be used to establish the contacts between faces. The construction of a topologically consistent 3D model from the city model data involves considerable user interaction. In fact, the city model, derived from satellite data, consists of 2D surface patches suitable for a "virtual reality" visual representation, but is neither consistent nor complete, and does not comprise volume cells. Therefore available floor plans, elevations and vertical sections of the buildings, which belong to Osnabrück university, have to be consulted to control the construction of the 3D model.

After the two models are merged into one, further editing of the cells can be used to e.g. cut out cellars, windows, doors and archways, or to create interior walls and intermediate ceilings, in order to yield a more realistic consistent topological 3D model integrating indoor and outdoor spaces (Figure 14.6).

Topological database queries such as determining the neighbouring buildings of the palace can be directly answered using the topological 3-G-Map structure.

14.9 Conclusion and outlook

In this paper we have described a general approach for modelling and managing the topology of objects in a 3D GIS - based on oriented d-Generalised

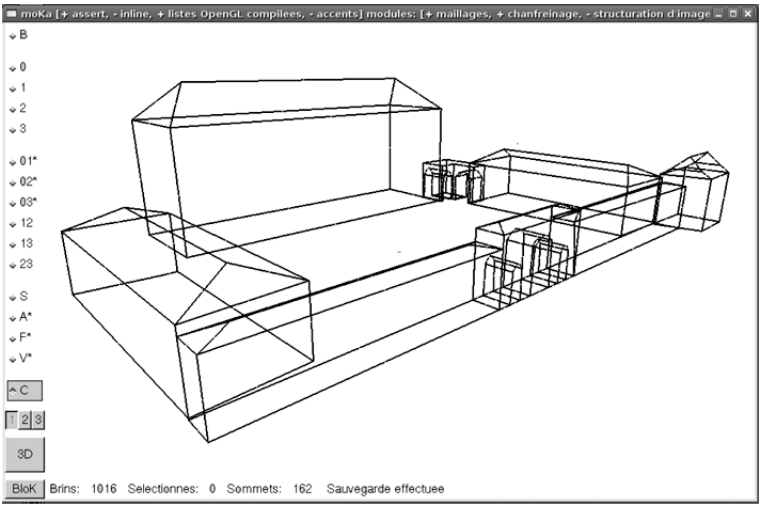

Fig. 14.6 Screen snapshot during editing session with Moka [15]: Introduction of archways, and correction of vertex position nconsistencies. Edges are represented by straight lines, though the corresponding arcs may be curves or polylines

Maps. The topological data model and its realisation in a database management system have been presented in detail. The realisation of the approach in an Object-Relational Database Management System (ORDBMS) has been presented. An application example as part of the Osnabrück city map combined with a 3D model of Osnabrück Palace showed the applicability of the approach.

In our future work we will also deal with topological operations on objects with different levels of detail (LOD) based on hierarchical d-G-Maps. This approach shall be implemented in a Multi-Representation Database and evaluated with cartographic data of our project partners at Hannover University.

References

[1] Mäntylä, M.: An Introduction to Solid Modelling. Computer Science Press (1988)

[2] Turner, A.K. (1992)(Ed.): Three-Dimensional Modelling with Geoscientific Information Systems, proc. NATO ASI 354, Kluwer, Dordrecht, 123–142.

[3] Balovnev, O., Bode, T., Breunig, M., Cremers, A.B., Müller, W., Pogodaev, G., Shumilov, S., Siebeck, J., Siehl, A., Thomsen, A.: The Story of the GeoToolKit – An Object-Oriented Geodatabase Kernel System. GeoInformatica 8(1) (2004) 5–47.

[4] Egenhofer, M.J.: A formal definition of binary topological relationships. In: Proc. 3th Int. Conf. on foundation of data organisation and algorithms (1989) 457–472
[5] Zlatanova, S.: 3D GIS for Urban Development. PhD dissertation, TU GRAZ, ITC Dissertation 69 (2000).
[6] Egenhofer, M.J., Frank, A.U. and Jackson, J.P.: A topological data model for spatial databases. In: Buchmann, A. P., Günther, O., Smith, T. R. and Wang, Y.-F.(eds.): Design and Implementation of Large Spatial, LNCS 409, Springer, Berlin (1990) 271–286 Zlatanova, S.: 3D GIS for Urban Development. PhD dissertation, TU GRAZ, ITC Dissertation 69 (2000).
[7] Pigot, S.: A topological model for a 3D spatial information system. 5th SDH, Charleston, (1992), 344–360.
[8] Brisson, E.: Representing Geometric Structures in d Dimensions: Topology and Order. Discrete & Computational Geometry 9 (1993) 387–426.
[9] Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. Int. Journal Comp. Geometry and applications 4(3) (1994) 275–324.
[10] Mallet, J.L.: Geomodelling. Oxford University Press (2002)
[11] Lévy, B.: Topologie Algorithmique – Combinatoire et Plongement. PhD Thesis, INPL Nancy (1999)
[12] Fradin, D., Meneveaux, D. and Lienhardt, P.: Partition de l'espace et hiérarchie de cartes généralisées. AFIG 2002, Lyon, décembre (2002), 12p.
[13] Raza, A., Kainz,W.: An Object-Oriented Approach for Modelling Urban Land-Use Changes. ACM-GIS (1999) 20–25.
[14] Mallet, J.L.: GOCAD: A computer aided design programme for geological applications. In: Turner, A.K. (Ed.): Three-Dimensional Modelling with Geoscientific Information Systems, proc. NATO ASI 354, Kluwer, Dordrecht, 123–142.
[15] MOKA: Modeleur de Cartes http://www.sic.sp2mi.univ-poitiers.fr/moka/ (2006).
[16] van Oosterom, P., Stoter, J.E., Quak, C.W., Zlatanova, S., The Balance Between Geometry and Topology. In: Richardson, D. and van Oosterom, P. (eds.), Advances in Spatial Data Handling, 10th SDH, Springer, Berlin (2002).
[17] Thompson, R.J.., van Oosterom, P.: Implementation issues in the storage of spatial data as regular polytopes. Information Systems for Sustainable Development-Part1 (2006) 2.33–2.46 (2006).
[18] Penninga, F., van Oosterom, P. and Kazar,B. M.: A tetrahedronized irregular network based DBMS approach for 3D topographic data modeling. In: Riedl, Andreas, Kainz, W., Elmes and Gregory A. (eds.): Progress in Spatial Data Handling, 12th SDH 2006, Springer, Berlin (2006) 581–598

[19] Saadi Mesgari, M.: Topological Cell-Tuple Structures for Three-Dimensional Spatial Data. PhD thesis University of Twente. ITC Dissertation 74 (2000).
[20] Thomsen, A., Breunig, M.: Some remarks on topological abstraction in multi representation databases. In: Popovich, V., Schrenk, M. and Korolenko, K. (eds.): 3rd workshop Inf. Fusion & GIS, Springer, Berlin (2007) 234–251.
[21] PostgreSQL.org: http://www.postgresql.org/docs (2006).
[22] PostGIS.org: http://postgis.refractions.net/documentation (2006).
[23] Butwilowski, E.: Topologische Fragestellungen bei der Kombination von 3D-Stadtmodellen mit 2D-Karten in einer Räumlichen Datenbank. Diplomarbeit, Fachgebiet Geographie, Universität Osnabrück, (2007).
[24] FRIDA: Free data from the city of Osnabrueck. http://frida.intevation.org/ueber-frida.html (2007).
[25] Breunig, M., Bär W. and Thomsen, A.: Usage of Spatial Data Stores for Geo-Services. 7th AGILE Conf. Geographic Information Science, (2004) 687–696.
[26] Bär, W.: Verwaltung geowissenschaftlicher 3D Daten in mobilen Datenbanksystemen PhD Thesis, dept. of Mathematics/Computer Science, University of Osnabrück (2007).
http://elib.ub.uni-osnabrueck.de/cgi-bin/diss/user/catalog?search=sqn&sqn=693
[27] pl/java: http://wiki.tada.se/display/pljava/Home