Editing Features in a TEN-based DBMS approach for 3D Topographic Data Modelling

Technical Report

Friso Penninga
Peter van Oosterom

GISt Report No. 43  May 2006
Editing Features in a TEN-based DBMS approach for 3D Topographic Data Modelling

Technical Report

Friso Penninga
Peter van Oosterom

GIS Report No. 43
May 2006
Summary

In this Technical Report the most up-to-date (May 2006) ideas about using a TEN-based DBMS approach for topographic data modelling are stated. The report is in paper-style, describing the basic operators for inserting topographic features into the TEN model. These features are described as set of constrained edges (and faces). The operators focus on the insertion of these edges.
Abstract. Topographic features (i.e., physical objects such as buildings, bridges, etc.) become more complex due to increasing multiple land use. As the same time increasing awareness of the importance triggers the development of more 3D GIS tools. In this paper we extend a topological 3D data model that relies on Poincaré algebra. Internally a Tetrahedronized Irregular Network (TEN) is used to represent the topographic features. These features are modelled as set of constraints. This paper focuses on the way in which these constraints can be inserted incrementally in the constraint TEN.

1 Introduction

Extending current topographic data sets into the third dimension is the overall objective of the research project 3D Topography (http://www.gdmc.nl/3dtopo n.d.) in which the author participates. Triggered by recent developments towards increasing multiple land use and raised awareness of the importance of sustainable urban development, the 3D Topography project aims at providing full 3D capabilities in the entire geo-information process, i.e. from data collection, processing, storage and analysis to visualisation. Topographic data (i.e. data concerning the collection of physical objects, such as buildings, roads, tunnels, dikes, rivers, etc.) is used in a wide variety of applications and tasks. As a result the data model should support all these tasks, thus disabling the possibility of optimizing a data structure for specific purposes. Another important characteristic of the future 3D topographic data model is the requirement of being able to handle very large data volumes, as due to ongoing sensor technique developments the amount of available 3D geo-information rapidly increases.

Within this research a volumetric approach of data modelling is developed. Section 2 will introduce this model, which is a TEN-based (Tetrahedronized Irregular Network, the 3D counterpart of the well-known TIN (Triangulated Irregular Network)) approach using Poincaré algebra. A small survey of requirements for updating features in a constraint TEN is performed in Section 3. In Section 4 a set of basic operators will be introduced, derived from the results of the small survey. This technical report ends with conclusions and further research in Section 5.
2 Previous relevant research

2.1 3D modelling: volume partition approach

Extending topographic data models into the third dimension is most relevant when dealing with large scale topography. As this will lead to a substantial increase in data volume, maintaining and ensuring data integrity becomes of extreme importance and choosing a topological approach is an obvious solution, see also (Ellul & Haklay 2006). Initially the idea was to use an integrated 2.5D/3D approach, thus modelling the world as a 2.5D surface and gluing 3D objects, such as buildings, tunnels and viaducts on top or below this surface. As a data structure we had a combination of a TIN (Triangular Irregular Network) and several TENs (Tetrahedral Irregular Networks) in mind. However, as discussed in (Penninga 2005), despite the conceptual simplicity, serious problems at design level occurred with this approach. As a result a full 3D approach was chosen (Penninga 2005), again using a TEN. This selection is motivated by computational advantages, the flatness of the faces (well defined by three points) and the presence of well-known topological relationships (Guibas & Stolfi 1985). Earlier experiences with TEN within our research group can be found in (van der Most 2004, de Vries 2001).

The decision to use a volumetric approach instead of a boundary representation is based on the observation that in the case of topographic objects each object has by definition a volumetric shape, implying there are no such things as point, line or area features. However, at a certain level of detail volumetric features might have point, line or area representations. In a Digital Landscape Model intended for large scale topography (and thus intended for a wide variety of applications), one should model the actual (volumetric) shape of the features. Preferences for a less dimensional representation at a specific scale level and for a specific task should be stored within a Digital Cartographic Model (i.e. the set of visualisation rules) and accordingly should not interfere with the landscape model.

The volumetric approach as introduced in (Penninga 2005) has another basic assumption, as in the case of topography we consider the real world as a volume partition (analogously to a planar partition): a set of non-overlapping volumes that form a closed modeled space. The most important consequence is that objects like earth and air are explicitly part of the real world and thus have to be modelled. It might seem that modelling objects like earth and air, in addition to more common topographic features like buildings, is more serving the abstract goal of clean modelling than an actual useful goal. However this is not the case. Firstly the space between the common features is often our subject of analyses, for instance in case of noise or odour modelling (de Kluijver & Stoter 2003). Secondly the presence of these features enables future extensions of the data model, for instance by subdividing the class air into more specific classes, such as air traffic corridor or telecommunication corridor. The same holds for the class earth, as this classification might be replaced by a more accurate one, for instance based on geotechnical or geological layers or polluted regions.
Although the modelling approach is (amongst others) defined by the assumption of the volume partition, area features can exist within the model, but only in specific cases. Some area features actually mark a border or transition between two volumetric features that are of such importance or usefulness that one would like to include these in the model. However, these area features cannot exist without the presence of the volume features. One can consider such an area feature as first derivative of a volume feature (or two volume features to be more precise). In the UML class diagram in Figure 1 (Penninga, van Oosterom & Kazar 2006) area features are modelled as association classes between volume features. These modelling concepts will be implemented using of a Tetrahedronized Irregular Network (TEN), a connected network of tetrahedrons (3D simplexes; a simplex is the simplest geometry in a dimension). Using 3D simplexes fits nicely with the volumetric approach: each tetrahedron represents (part of) a volumetric feature. As each tetrahedron is bounded by four triangles (2D simplexes), these triangles might represent (parts of) area features, as described above. The TEN datastructure also incorporates knowledge on topological relations between the tetrahedrons, as each triangle bounds two tetrahedrons. Left or right are meaningless in three dimensions, but if oriented triangles are used, one can relate to these two tetrahedrons in either positive or negative direction. As discussed in (Penninga et al. 2006) several different implementations of a TEN structure can be considered. In this paper only the one based on Poincaré algebra will be used.

2.2 Poincaré algebra

The previously introduced volumetric approach uses tetrahedrons to model the real world. These tetrahedrons in the TEN structure consist of nodes, edges and triangles. All four data types are simplexes: the simplest geometry in each dimension. A more formal definition (Hatcher 2002) of a $n$-simplex $S_n$ is given below:

A $n$-simplex $S_n$ is the smallest convex set in Euclidian space $\mathbb{R}^m$ containing $n + 1$ points $v_0, \ldots, v_n$ that do not lie in a hyperplane of dimension less than $n$. As the $n$-dimensional simplex is defined by $n + 1$ nodes, it has the following notation: $S_n = \langle v_0, \ldots, v_n \rangle$.

Equivalent conditions to the hyperplane condition would be that the difference vectors $v_1 - v_0, \ldots, v_n - v_0$ are linearly independent or, if one considers $v_0, \ldots, v_n$ as set of vectors, that these vectors are affinely independent.

It is assumed that all simplexes are ordered. With any $n$-simplex $(n + 1)!$ distinct ordered simplexes are associated. All even permutations of an ordered simplex $S_n = \langle v_0, \ldots, v_n \rangle$ have a positive orientation, all odd permutations a negative one. So for instance the following is true: $S_1 = \langle v_0, v_1 \rangle = - \langle v_1, v_0 \rangle$ and $S_2 = \langle v_0, v_1, v_2 \rangle = - \langle v_0, v_2, v_1 \rangle = \langle v_1, v_2, v_0 \rangle = - \langle v_1, v_0, v_2 \rangle = \langle v_2, v_0, v_1 \rangle = - \langle v_2, v_1, v_0 \rangle$. A face of $S_n$ is a simplex whose vertices form a non-empty subset of $\{v_0, \ldots, v_n\}$. If the subset is proper (i.e. not the whole of $\{v_0, \ldots, v_n\}$) than the face is a proper face (Giblin 1977). A $n$-simplex has in total...
Fig. 1. UML class diagram of the proposed data structure
2(n+1)-2 proper faces. For the number of faces of a specific dimension the following holds: a \( n \)-simplex has \( \binom{n+1}{p+1} \) faces of dimension \( p \) with \( 0 \leq p < n \). The 0- and 1-dimensional faces of a \( n \)-simplex form a complete graph on \( n+1 \) vertices.

According to Poincaré algebra the boundary of a \( n \)-simplex is defined by the following sum of \( n-1 \) dimensional simplexes (the hat indicates omitting the specific node):

\[
\partial S_n = \sum_{i=0}^{n} (-1)^i < v_0, \ldots, \hat{v}_i, \ldots, v_n >
\]

This results in (see Figure 2 (Hatcher 2002)):

\[ S_1 = < v_0, v_1 > \quad \partial S_1 = < v_1 > - < v_0 > \]
\[ S_2 = < v_0, v_1, v_2 > \quad \partial S_2 = < v_1, v_2 > - < v_0, v_2 > + < v_0, v_1 > \]
\[ S_3 = < v_0, v_1, v_2, v_3 > \quad \partial S_3 = < v_1, v_2, v_3 > - < v_0, v_2, v_3 > + < v_0, v_1, v_3 > - < v_0, v_1, v_2 > \]

As with a simplex \( S_n(n+1)! \) distinct ordered simplexes were associated, the boundaries of all other combinations can be given. In case of \( S_1 = - < v_0, v_1 >= < v_1, v_0 > \) the boundary \( \partial S_1 = < v_0 > - < v_1 > \). In a similar way the boundaries of the other five combinations of \( S_2 \) and the other 23 combinations of \( S_3 \) can be given. Note that for instance in the case of \( S_3 \) for every combination the orientation of all four bounding triangles is identical; either all triangles point towards the exterior or all triangles point towards the interior of the tetrahedron.

An interesting application of the boundary formula is joining or merging two simplexes of equal dimension. For instance, if we join the two neighbouring triangles \( S_{21} = < v_0, v_1, v_2 > \) and \( S_{22} = < v_0, v_2, v_3 > \) into a 2D complex \( C_2 \), adding the boundaries result in:
\[ \langle v_1, v_2 \rangle - \langle v_0, v_2 \rangle + \langle v_0, v_1 \rangle = \langle v_1, v_2 \rangle + \langle v_0, v_1 \rangle + \langle v_2, v_3 \rangle + \langle v_3, v_0 \rangle. \]

Note that the shared boundary \( \langle v_0, v_2 \rangle \) is removed as it appeared once with positive and once with negative direction.

Adding boundaries can be very useful in our modelling approach. If for instance a building is modelled as a set of eight tetrahedrons (see Figure 3), the buildings boundary representation can be obtained by merging the boundaries of all eight tetrahedrons. This combination of several simplexes is called a simplicial complex or \( n \)-cell (Giblin 1977). The triangles of \( C_3 \) are the boundary triangulation of this building. This boundary triangulation might be used in the visualization process. If one is interested in the polyhedron of the building, adding up the boundaries of the boundary triangles with identical (within a tolerance) normal vector direction into flat polygons will result in the seven boundary faces of this building. Up to this point only simplexes and simplicial complexes are discussed. It is also possible to create a larger structure consisting of connected \( n \)-simplexes, partitioning the whole \( n \)-dimensional domain. In
3D this is the Tetrahedronized Irregular Network. Such a structure contains several topological relationships, thus enabling both topological querying and, more important, validation tools in order to maintain data integrity. Another important concept with respect to topological relationships in a TEN structure is the coboundary. A coboundary is more or less the opposite of a boundary. We define the coboundary of a $n$-dimensional simplex $S_n$ as the set (of varying size) of all $(n + 1)$-dimensional simplexes $S_{n+1}$ of which the simplex $S_n$ is part of their boundaries $\partial S_{n+1}$. For instance, a triangle has three boundary segments (i.e. edges) and two coboundary segments (i.e. tetrahedrons). An edge has two boundary segments (i.e. nodes) and (in 3D!) an unknown number of coboundary segments (i.e. triangles).

2.3 Data model

To summarize and conceptualize the ideas of this section, the UML class diagram of our topographic data model is given in Figure 1. This model is the newest version of earlier modelling attempts made in (Penninga 2005) and (Penninga et al. 2006). In the upper left corner the abstract class RealWorldPhenomena can be seen, which is a generalization of the class VolumeFeature. This generalization is included to emphasize the concept that all topographic (i.e. physical) objects have a volumetric shape and therefore should be modelled as volume features. Less dimensional features are present in the model (AreaFeature, LineFeature, PointFeature), but these features are modelled as association classes. As a result instances of these derived features are lifetime dependent from the higher dimensional instances. Each feature is represented by one or more simplex(es) of corresponding dimension. Between the Tetrahedron, Triangle, Edge and Node classes derived associations exist that are equivalent with the boundary and coboundary relationships. Note that -in contrast with the models in (Penninga 2005) and most models in (Penninga et al. 2006) - the Tetrahedron, Triangle and Edge class are all specified by an ordered list of Nodes. This is a direct result of the implementation of Poincaré algebra within our modelling approach.

3 Editing features: conceptual requirements for the DBMS approach

As TEN models are expected to be very large, efficient DBMS data structure design is very important. Issues regarding the implementation of an incremental tetrahedronization algorithm within the DBMS, locking, update functions both on simplex level and feature level, indexing and clustering and storage requirements are addressed in (Penninga et al. 2006). This paper focuses on the design and implementation of editing features in the DBMS-incorporated TEN structure. In this section the requirements from a users perspective will be identified. In the next section the requirements will be translated into operations on a constrained TEN.
As less dimensional features can only exist as derivatives from volume features, it is sufficient if the model is capable of adding volume features: the less dimensional features will automatically be present in the structure in the form of a (set of) triangle(s), edge(s) or node(s). If one wants to add a volume feature, for instance a building, one needs to ensure correct geometrical representation in the TEN by enforcing the boundary faces to be present in the structure. This is not trivial, as the basic input of a triangulation/tetrahedronization algorithm is a set of nodes. Usually the algorithm determines, based on certain rules (for instance the well-known Delaunay criterion) which nodes will be connected by edges and therefore one cannot be sure that two specific nodes will be connected by an edge or that three specific nodes will form a triangle. As a result one cannot represent a specific shape into a triangulation/tetrahedronization by inserting only its nodes. In order to overcome this drawback the constraint edges were introduced. The constraints on these edges means that the edges cannot be removed (by flipping) by the algorithm in order to fulfill its rules. As a result it is possible to represent features by inserting a set of constraint edges. Unfortunately, constraint faces cannot be inserted directly, but have to be translated into a set of constraint edges.

As a result the insertion process of the volume features needs the following aspects:

- Its outer boundary needs to be triangulated and all resulting edges should be treated as constraints (actually one needs an additional check whether all faces are present, see for more details the end of this section.
- The interior needs to be tetrahedronized. This tetrahedronization can be performed either directly in the TEN or separately, after which all resulting edges can be inserted into the TEN. Input in both cases is the set of constraint edges of the outer boundary.
- Regardless which of the two previous options is used, local tetrahedronization might be necessary in order to optimize the structure by creating better-shaped tetrahedrons.
- Updating the relevant feature table(s)

These four steps are the basic steps in the process. During implementation this process might be extended with steps that check which features are currently located at the position of the new feature, automatically fit buildings to the earth surface, etc., etc.

However, a far more interesting question is the exact procedure by which the feature representation is inserted into the model. One might consider enforcing the presence of the complete outer boundary triangulation and detect and classify the resulting internal tetrahedrons (as these tetrahedrons represent the volume feature) or one can delete the existing tetrahedral structure within the boundary triangulation and replace it with the separately calculated tetrahedronization. The first option has the advantage that it influences the structure very local, the second option has the advantage that optimizing the internal tetrahedral structure will be easier in a separate meshing process.
4 Editing features: translating requirements into basic operations on topological elements

As most operations introduced in this paper are based on the operations introduced in (Penninga et al. 2006), these operations are given again. Note that these operations on topological elements do not take notice of constraints.

Insert node and incident edges/triangles/tetrahedrons (or the reverse operation remove node), where 3 cases can be distinguished depending on where the node is inserted:

- Middle of tetrahedron (one tetrahedron involved) and added are +1 node, +4 edges, +6 triangles, and +3 tetrahedrons (respectively the 0/1/2/3-simplexes).
- Middle of triangle (2 tetrahedrons involved) and added are +1 node, +5 edges, +7 triangles, and +4 tetrahedrons.
- Middle of edge (n tetrahedrons involved) and added are +1 node, +(n+1) edges, +2n triangles, +n tetrahedrons.

Note that these three cases can be described nicely by the use of Poincaré algebra from the previous section. No calculations or whatsoever are necessary. For instance in the case of inserting a node $v_4$ in the interior of tetrahedron $S_3 = \langle v_0, v_1, v_2, v_3 \rangle$, the boundary operator can be used to derive the four bounding triangles. From these triangle $S_{21}, \ldots, S_{24}$ the new tetrahedrons can be constructed by adding $v_4$ as fourth node. Using the boundary operator again on the triangles will result in the six edges and by adding the fourth node to these edges, the six new triangles are defined. By applying the boundary operator again the four nodes are found and in combination of the fourth node this will result in the four new edges.

4.1 Insertion of volume features into a TEN structure

Basic assumption is that the creation of the TEN model starts with an initial TEN, large enough to contain the model, consisting of a few initial tetrahedrons representing air and earth (see (Penninga 2005) for more details). As a result every update step will take place in an incremental algorithm, regardless whether the model is almost complete or almost empty.

As seen in the previous section the complete boundary of the volume feature needs to be inserted into the TEN structure. Before inserting this boundary needs to be triangulated. With this triangulated surface as a basis two different approaches are possible.

The proposed new set of basic operators is based on expanding the previous operators from insert node to insert edge operators. Insert edge operators are more complex than just repeating an insert node operator twice, as one also needs to ensure that the start and end node are connected by an edge. In inserting a start- or endnode in a tetrahedron in an existing TEN structure four
different cases can be distinguished, as a the node can lie within a tetrahedron, on a triangle, an edge or a node. In case of inserting the start- and endnode of the edge 16 theoretical cases can be identified, in which 9 unique cases will be distinguished (see Table 1). Case 0 is not a real case, as by defintion this edge would already be present.

<table>
<thead>
<tr>
<th>Node lies on</th>
<th>Node</th>
<th>Edge</th>
<th>Triangle</th>
<th>Tetrahedron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Edge</td>
<td>(1)</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Triangle</td>
<td>(2)</td>
<td>(5)</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Tetrahedron</td>
<td>(3)</td>
<td>(6)</td>
<td>(8)</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1. $4 \times 4 = 16$ theoretical cases, of which 9 are unique (and 1 does not exist)

First, we describe the nine basic operators (based on where the edges start and end node are inserted):

1. One node on an existing node, one node on an edge ($n$ tetrahedrons involved): As one node is already present, only a single node needs to be inserted, by which each tetrahedron is split into two new tetrahedrons (see Figure 4 for case $n=4$). Added are: +1 node, +$n+1$ edges, +2$n$ triangles, +$n$ tetrahedrons.

2. One node on an existing node, one node on a triangle (2 tetrahedrons involved): Again only a single node needs to be inserted, by which the two tetrahedrons are split into three new tetrahedrons (as the endnode lies in
plane $< v_0, v_2, v_3 >$, see Figure 5. Added are: +1 node, +5 edges, +10 triangles, +4 tetrahedrons.

**Fig. 5.** The tetrahedron is split into three tetrahedrons

3. One node on existing node, one node inside the tetrahedron: The tetrahedron is split into four tetrahedrons (see Figure 6). Added are: +1 node, +4 edges, +4 triangles, +3 tetrahedrons.

**Fig. 6.** The tetrahedron is split into four tetrahedrons

4. Both nodes lies on edges (2n or n+m tetrahedrons involved):
   Depending on the question whether the two nodes lie on the same edge, the number of added elements is +2 nodes, +(2n+2) edges, +4n triangles, +2n tetrahedrons or +2 nodes, +((n+1)+(m+1)) edges, +(2n+2m) triangles, +(n+m) tetrahedrons.

5. One node lies on a boundary triangle, the other node lies on an edge ((n+1) tetrahedrons involved):
The node on the triangle is inserted, like the first step in case 2. Added are: +1 node, +5 edges, +7 triangles, +4 tetrahedrons. The second node lies on one of the outer edges, in the case of Figure 7 on the edge $<v_0, v_1>$. As a result tetrahedron $<v_0, v_1, v_2, startnode>$ will be split into two tetrahedrons and the same goes for the (n-1) other tetrahedrons that use this edge, changing the totals to: +2 nodes, +(n+6) edges, +(2n+7) triangles, +(n+4) tetrahedrons.

**Fig. 7.** Start node on boundary triangle, end node on edge

6. One node lies on an edge, the other node lies in the middle of one of the n involved tetrahedrons (n tetrahedrons involved):
   First the node on the edge is inserted. Added are +1 node, +(n+1) edges, +2n triangles, +n tetrahedrons. In Figure 8 the case n=4 is illustrated. As one can see, each original tetrahedron in split into two new tetrahedrons. Since one knows in which original tetrahedron the second node would be inserted, two sub-classes can be distinguished, related to these two new tetrahedrons:

**Fig. 8.** Insertion of a node on an edge in case n=4
(a) The second node lies in the middle of one of the two tetrahedrons, changing the totals to: +2 nodes, +\(n+5\) edges, +\(2n+6\) triangles, +\(n+3\) tetrahedrons. Similar to case 2(a) the constraint edge is one of the newly formed edges.

(b) The second node lies in the triangle between the two tetrahedrons, changing the totals to: +2 nodes, +\(n+6\) edges, +\(2n+7\) triangles, +\(n+4\) tetrahedrons. Similar to case 2(b) the constraint edge is one of the newly formed edges.

As the second node lies in the interior of the split tetrahedrons and not on its boundary, it is impossible for this second node to lie on an edge, as this edge lies in the triangle (boundary) of the original tetrahedron.

7. Both nodes lie on boundary triangles (2 or 3 tetrahedrons involved):
Regardless whether the nodes lie in the same boundary triangle, after insertion of both nodes +2 nodes, +10 edges, +14 triangles, +8 tetrahedrons are added to the TEN structure. The constraint edge is one of the newly formed edges.

8. One node lies on a boundary triangle, the other node lies in the middle of one of the two tetrahedrons (2 tetrahedrons involved):
First the node on the boundary triangle is inserted. Added are: +1 node, +5 edges, +7 triangles, +4 tetrahedrons. Now the node in the middle of the tetrahedron is inserted. As this tetrahedron is split into three tetrahedrons, three different sub-cases can be distinguished:

(a) The second node lies in the middle of one of the three (as one already knows in which of the two split tetrahedrons the node will be inserted) newly formed tetrahedrons, see Figure 9. This tetrahedron is split into four tetrahedrons, changing the totals to: +2 nodes, +9 edges, +13 triangles, +7 tetrahedrons. The constraint edge is already present as it is one of the four newly formed edges.

(b) The second node lies on one of the three (as one already knows in which of the two split tetrahedrons the node will be inserted) newly formed triangles, in the case in Figure 10 in triangle \(< v_0, v_3, \text{startnode} >\). The totals change to: +2 nodes, +10 edges, +14 triangles, +8 tetrahedrons. Again the constraint edge is created by the insertion of the second node.

(c) The second node lies on the edge from the start node to the node opposite of the triangle in which the first node is inserted (as one already knows in which of the two split tetrahedrons the node will be inserted), see Figure 11 (edge \(< v_0, \text{startnode} >\). In this specific case one knows that there are three tetrahedrons involved, so the totals change to: +2 nodes, +9 edges, +13 triangles, +7 tetrahedrons. Again the constraint edge is already present in the structure after insertion of the second node.

9. Both nodes in the middle of the tetrahedron (1 tetrahedron involved): First the start node is inserted, by which the tetrahedron is split in four new
Fig. 9. First node on the triangle, second node within a tetrahedron

Fig. 10. First node on the triangle, second node on triangle $< v_0, v_3, \text{startnode} >$
First node on the triangle, second node on edge $< v_0, startnode >$

tetrahedrons. Added are: +1 node, +4 edges, +6 triangles, +3 tetrahedrons.
Now the end node is inserted, leading to three sub-cases:
(a) End node lies within one of the four newly formed tetrahedrons (1 tetra-
hedron involved), changing the totals: +2 nodes, +8 edges, +12 triangles, +6 tetrahedrons
(b) End node lies on one of the six newly formed triangles (2 tetrahedrons
involved), changing the totals: +2 nodes, +9 edges, +13 triangles, +7 tetrahedrons
(c) End node lies on one of the four newly formed edges (in general this
affects n tetrahedrons, but in this specific case: 3 tetrahedrons involved),
changing the totals: +2 nodes, +8 edges, +12 triangles, +6 tetrahedrons

Although the start and end node are present, one needs to check whether
the constraint edge is present already. This has to be done for each sub-case:
(a) In this case the constraint edge is not present yet in the model. The
situation is illustrated in Figure 12 where only the two relevant tetra-
hedrons are drawn $< v_0, v_1, v_2, startnode >$ and $< v_0, v_1, v_2, endnode >$.
The constraint edge (thick gray line) has to be inserted, but intersects
triangle $< v_0, v_1, v_2 >$. A new node is inserted at this intersection point,
resulting in 5 new edges, of which two represent the constraint edge. The
totals change to: +3 nodes, +13 edges, +19 triangles, +10 tetrahedrons

(b) Since the end node lies on one of the newly formed triangles and these
triangles all have the start node as one of their nodes, the constraint
dge is already present after insertion of the end node, see Figure 13.
The totals remain the same: +2 nodes, +9 edges, +13 triangles, +7 tetrahedrons
(c) Since the end node lies on one of the newly formed edges and the start
node is also a node of these four newly formed edges, the constraint
dge is already present after insertion of the end node, see Figure 14.
Fig. 12. The constraint edge is segmented by a new node in the intersecting triangle

Fig. 13. The second node is inserted within one of the triangles
The totals remain the same: +2 nodes, +8 edges, +12 triangles, +6 tetrahedrons

**Fig. 14.** The second node is inserted on one of the edges

With this set of operations all constraint edges can be inserted. In case constraint edges are longer, i.e. the edges are crossing multiple tetrahedrons, the constraint edges can be split at the intersection points with the tetrahedron boundaries. These segments can be treated separately by one of the previous operations. However ensuring the presence of all constraint edges is not sufficient, one has no guarantee that the corresponding faces are also present. This is illustrated in Figure 15. In this illustration triangle \( \langle v_0, v_1, v_2 \rangle \) lies more or less horizontal and triangle \( \langle v_0, v_2, v_4 \rangle \) more or less vertical. Node \( v_3 \) is placed behind the face spanned by \( \langle v_0, v_2, v_4 \rangle \) and above the face spanned by \( \langle v_0, v_1, v_2 \rangle \). In this case the gray edges \( \langle v_0, v_2 \rangle, \langle v_2, v_4 \rangle, \langle v_0, v_3 \rangle \) are the constraint edges, but since the tetrahedrons are defined as \( \langle v_0, v_1, v_2, v_3 \rangle, \langle v_0, v_1, v_3, v_4 \rangle \) and \( \langle v_1, v_2, v_3, v_4 \rangle \), the required constraint face \( \langle v_0, v_2, v_4 \rangle \) is missing. Therefore an additional test is required that checks whether each constraint triangle from the boundary triangulation is also present in the TEN, a task that is quite straightforward due to the node-based notation of all simplexes. However one should not only test for 1:1 relations between triangles in the TEN and triangles in the previously calculated boundary triangulation, but also for 1:m relationships. Here an additional test is required to see whether the m triangles lie in the same plane, most likely performed by comparing the triangles normal vectors. A prerequisite of this testing method is that the TEN structure is valid, i.e. there are for instance no dangling edges or faces.

Another (and better-known) option to insert the boundary triangulation into the TEN structure is to insert only the nodes and try to perform edge recovery. The
Fig. 15. although all constraint edges are present, the constraint face isn’t

process of recovering missing edges is described by (Cavalcanti & Mello 1999) and covers stitching, a process that is based on adding nodes on midpoints of missing constraint edges. This idea uses the property of the Delaunay triangulation that each node is connected to the closest node and by adding nodes on midpoints one tries to make these additional nodes the closest nodes. However, this process does not always converge. This problem is solved partially in 3D (Cavalcanti & Mello 1999).

Another approach to edge recovery is given by (Liu & Baida 2000), whose approach is based on flipping. They use four basic types of flipping, namely T23, T32, T22 and T44, following the notation of Joe ((Joe 1995)).

Of course the important question of this section is which approach is better. The first approach requires prior knowledge of the spatial relationships between the constraint edges and the TEN structure, including possible prior segmentation of the constraint edges by intersecting the edges with the TEN structure. During the operation a number of tests are required to determine whether the node lies inside a tetrahedron, on its boundary triangle or on its edge. However, as long as we can use the locked area box of the update process as a boundary for our search area, performance can be acceptable. An advantage of this process is that it does not require a lot of additional nodes, as in most operations (except operation 1.1) the constraint edge (segment) is already present after insertion of the second node.

4.2 Modelling the interior of volume features

Although the volume feature will be present in the data structure as soon as the constraint edges and faces are present and all internal tetrahedrons are classified
as the specified object, one might consider different approaches in tetrahedronizing the interior of the features. The method with the least additional workload is to identify all tetrahedrons within the boundary triangulation and classify them as representing the specific volume object. However, this will not give any insight in the interior tetrahedronization with respect to its quality or its size (for instance if more nodes than strictly necessary are present). An alternative would be to keep record of all segmentation of the prior boundary triangulation and use this refined boundary triangulation as input for a separate tetrahedronization algorithm, which will result in an optimal internal tetrahedronization. This tetrahedronization can replace the current interior tetrahedronization.

4.3 Quality improvement of TEN structure

As the model will grow over time due to ongoing refinements and incremental updates, a set of tools for quality maintenance of the data structure is necessary. Due to series of deletes and inserts ill-shaped triangles and tetrahedrons might appear. To ensure numerical stability these triangles should be repaired. Several different approaches might be used complementary, for instance flipping and insertion of centrepoints of circumspheres of poor-shaped tetrahedrons.

3D flipping is based on the fact that there are only two ways to tetrahedronize a convex set of five points (assuming this is not prevented by constraints): one with two tetrahedrons and one with three tetrahedrons (see Figure 16). One of these two tetrahedronization must be the Delaunay tetrahedronization of five points (Cavalcanti & Mello 1999). Therefore switching from 2 to 3 tetrahedrons or vice versa might improve the quality of the data structure. Another option

for TEN improvement is the method used by (Shewchuk 1997). This method is based on the circumradius-to-shortest edge ratio of triangles or tetrahedrons. The circumsphere is a sphere through the nodes of a tetrahedron, the circumradius the radius of this circumsphere. The ratio between the circumradius and the shortest edge is (in most cases) a good indicator for how evenly the tetrahedron is shaped. In case ill-shaped tetrahedrons are detected, adding a node in the circumcentre is an adequate cure in most cases. By this procedure most sliver
tetrahedrons will be removed. In case of an ill-shaped tetrahedron that can't be improved by adding the circumcentre as node (for instance due to the presence of constraint edges) using Steiner points to partition constraint edges might be very useful.

5 Conclusions and future research

In this paper a further step is taken in the development of a 3D topographic data structure. The TEN structure is very well accessible through the use of Poincaré algebra, which also enable manipulation (joining, merging). A last gap in the theoretical foundation of the TEN-based DBMS approach is closed with the development of six basic classes of operations on the constraint TEN to integrate the features in the model. By segmenting the constraint edges these nine classes are sufficient for the insertion of features and effects of this insertion are very local. In the upcoming months further implementation of our ideas on the TEN model will hopefully lead to a prove of concept of our current theoretical basis.

References

http://www.gdmc.nl/3dtopo (n.d.).
Reports published before in this series:
