A 3D topological model for augmented reality

Siyka Zlatanova and Edward Verbree
Department of Geodesy, Delft University of technology
Thijsseweg 11, 2629 JA, Delft, The Netherlands
s.zlatanova@geo.tudelft.nl, e.verbree@geo.tudelft.nl

1 Introduction

With the advances of the computer and vision technology mobile augmented reality systems attempt to go beyond the world of indoor applications. Among the variety of challenging issues we concentrate on structuring and database organisation of the 3D model required for pose determination and rendering of virtual objects. An outdoor application will need a 3D model of size comparable to one town, i.e. thousands of houses, streets, parking lots, etc. Such application faces all the problems in processing and maintaining large data sets that are organised in a Geographic Information System (GIS).

To our experience, the augmented reality system making use of large data maintained in a GIS is still lacking. In this respect, the UbiCom project TU Delft, The Netherlands has to be considered a pioneer. The goal of the project is development of a wireless augmented reality system for outdoor applications that employs a 3D GIS for positioning and rendering (see [13]). Many vision systems have been currently developed but most commonly they operate only in office like environments that do not require large 3D models. Some examples are *Finale* (see [4]), *AVRID* (see [1]) and *RobiVision*. The project Robivision (see [14]) is one of the few projects aiming at utilisation of rather large 3D models, i.e. a model of the indoor space of a ship. The 3D model is still a typical CAD model, i.e. 3D topology is not of primary interest.

This paper presents a 3D model aiming at both maintenance of 3D topology (one of the most important features of a 3D GIS) and efficient organisation of 3D data for augmented reality applications. The paper is organised in three sections: first the requirements to the data structure are specified with respect to the tasks of the vision system, second the proposed data structure is discussed and finally some initial experiments within Oracle database are reported.

2 Requirements to the 3D model

Discussions related to the content and the structuring of data in 3D GIS can be found in many publication on 3D GIS (e.g. see [3], [5], [6], [7], [10], [14], [15], [17]). Therefore, in this paper, we will focus on the specific requirements to the 3D model with respect to the system architecture designed within the UbiCom project. Two subsystems of the augmented system architecture rely on the 3D model.

First, the 3D GIS is to be used for the accurate positioning of the mobile unit. The pose determination in the UbiCom system is based on a vision system (see [9]). The mobile equipment (a video camera, GPS, accelerators and an inertial system) provides an initial approximate positioning with insufficient accuracy (2-10 meters). The accurate positioning has to be achieved by matching lines extracted from the video images and lines retrieved from the 3D GIS. Furthermore, the more lines are organised in the database, the better is

expected to be the result of matching procedure. It is apparent that the lines (in large amounts) are the “objects of interest” for the pose determination.

.

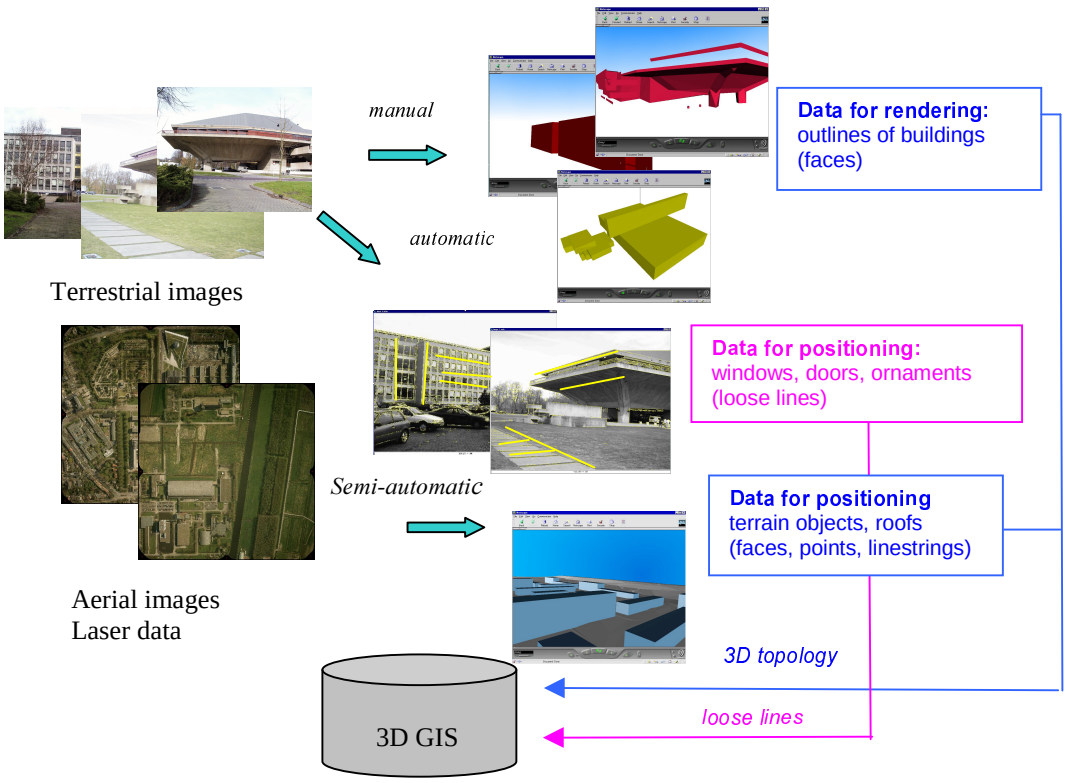

Figure 1: The process of 3D reconstruction for UbiCom

Second, the rendering subsystem (for visualisation of virtual objects) requires specific data about the position and the shape of the physical objects in the field of view, i.e. those objects that can be potential occludes of the virtual objects (see [8]). More precisely, only the outlines of 3D objects (e.g. buildings, man-made objects) are of real interest. The geometric representation of such objects has to assure connectivity and continuity, i.e. gaps between polygons or polygons with holes are not acceptable since they may disturb the rendering. Speaking formally, 3D topological consistency is highly appreciated.

The analysis of the functionality expected from 3D GIS for outdoor augmented reality system can be summarised into requirements to the database model as follows: maintenance of topologically structured 3D objects and large amounts of details represented as individual lines. Figure 1 gives a broad overview of the 3D re-construction procedure and corresponding data sets.

3 The state of the art in 3D structuring

The research in representing and structuring reality in 3D systems is extensive but rather fragmented. Concerned about the large amount of data of real-world models, computer graphics specialists explore models capable of maintaining these data (geometry and texture) and performing queries against the visualisation frustum in real time. The utilisation of different Levels of Detail (LOD) per object and their appropriate real-time control is the most popular approach to reduce data during the visualisation process.

Examples of such models are the virtual GIS presented by Kofler (see [3]) utilising 3D R-tree and Lindstrom et al (see [5]) based on progressive meshes. The disadvantage of such models is mainly underestimation of the importance of 3D topology, which results in weakness in maintaining data consistency.

In principle, only research in the GIS community is trying to work out an extended conceptual model capable of integrating geometric (position, shape and size) and thematic characteristics of objects and mutual spatial relationships. One stream of investigations emphasises on formalism (structure, ordering and operators) to construct a geometric object regardless of the dimension (see [11]). Such models aim at the complete representation of all the topological relationships among the objects from different dimensions. The models can be referred to as an *implicit* representation of objects, i.e. the relationships are stored and the description of the objects can be derived out of them. These models usually increase the size of the data for storage and require powerful techniques for restricted spatial search. Many reported 3D models give priorities on the description of the objects (i.e. an *explicit* description of objects). More details on data structures of this group can be found in [6], [10], [15], [17]. The major problem of such 3D models is that a few of them are experimented for really large data sets.

The need of standards models and operators for maintenance, query and retrieval of real-world data is recognised by many vendors designing GIS and CAD software as well. The intensive work on clarifying guidelines for developers resulted in OpenGIS specifications (see [7]). The approaches proposed there, however, are based on separate maintenance of geometry and topology objects, which in practice leads to large duplications. Furthermore, the most of the models proposed for implementation consider mostly the 2D world.

4 3D topological model

Bearing in mind the requirements to the model delineated in the previous section and utilising recent achievements in 3D GIS research, we propose a topological model that aims at facilitation of both tasks – line matching for accurate positioning and correct rendering of virtual objects. The proposed 3D model is a typical implicit boundary model. Each *n*-dimensional object is associated with four abstractions namely *point*, *linestring*, *surface* and *polyhedron* (see Figure 2). The notations of the four abstract objects correspond to the ones accepted in the OpenGIS specifications (see [7]). A *point* is an object that does not have shape or size but position. A *linestring* is a type of an object that has length and position. A *surface* is an abstraction of object that has position and area. A *polyhedron* has a position and a volume. These objects called geometric objects (**GO)** are built of smaller, simpler elements, i.e. constructive objects (**CnsO**). The model consists of two **CnsO**, i.e. *node* and *face*. Nodes describe spatial objects that can be represented as linestrings (e.g. pipe lines) and points (e.g. trees, lampposts). Nodes are constructive elements of faces as well. The order of the nodes in the face is known. Faces are to be used for the reconstruction of objects that are associated with surfaces (e.g. streets, parking lots) and polyhedrons (e.g. buildings).

Besides the geometric characteristic each object has *thematic* characteristics. For example the spatial object building may be characterised by year of building, owner and usage that is referred to as thematic characteristics. This aspect of the objects, however, is not discussed here. More details regarding this issue can be found in [6], [10], [15]. Figure 3

shows a schema of the model. Each **GO** or **CnsO** is represented by a rectangular block. The relationships “part of”, “has” and “belong to” are denoted by arrows as the direction correspond to many-to-one type of relationships. For example, face is a “part of” polyhedron and a polyhedron “consist of” many faces.

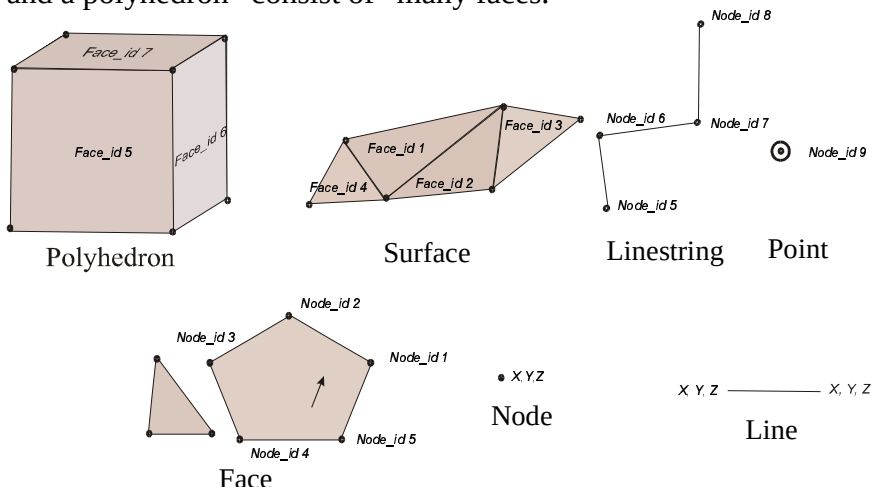

Figure 2: Examples of spatial objects to be supported

Since data for pose determination are simple loose lines, we propose a non-topological organisation, i.e. the lines are encapsulation with their co-ordinates and stored as a separate data set, i.e. *lines*. Each line is considered as a strait line represented by two sets of co-ordinates. The number of lines is expected to be rather large (for one façade, it may rise to 300-400) and therefore the 3D reconstruction process (see Figure 1) will ensure the relationships “a line belong to a face” (see Figure 3) to be created and explicitly stored in the database.

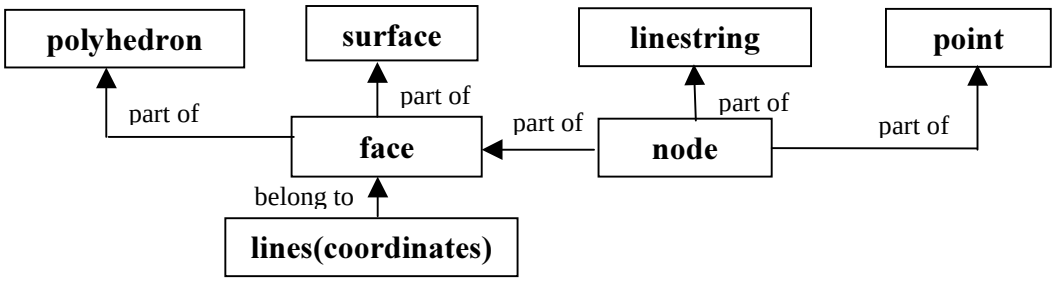

Figure 3: The proposed topological model

The topological model is similar in the part related to the **GO** to the ones presented in [6] and [10] but differs in the part of the used **CnsO**. Proposed model uses only two **CnsO**. The 1D-cell, often called *arc* or *edge* (see [6], [10], [11]), is omitted. The arc in 2D space have the unique feature of defining 1:2 relationships with faces and nodes, i.e. an arc has two neighbouring faces and nodes. This feature is only partially true in 3D and therefore the explicit storage of arcs does not bring significant facilitation. Such representation, i.e. without arcs, allows speed acceleration in the traverse of the model (see [17]). Moreover the creation of the VRML scene graph is facilitated, since the representations of polyhedron and surface are similar to description of irregular shapes in the VRML node *IndexeFaceSet*.

5 Implementation in Oracle database.

For flexible implementation of the model described above the object/relational database Oracle 8*i* was selected. The database offers a number of possibilities for representing the spatial objects described above. Three different implementations of the model were created and experimented.

5.1 Relational implementation.

The first straightforward approach is the relational implementation. For each geometric object a separate relational table is created. This is to say that the entire model consists of seven relation tables. For simplicity, the names of the tables are chosen to correspond to the names of the objects, i.e. NODE, FACE, LINE, POINT, LINESTRING, SURFACE and POLYHEDRON. The implementation of the NODE table is trivial: one column for the identifier of the node and the three columns for the (geodetic) co-ordinates of the points. The table POINT accommodates the identifier of the point and the identifier of the node that describes the spatial object. Since the remaining tables have very similar structure, only the FACE table will be explained. The relationship between a face and constituting nodes is one-to-many (1:m) which can be represented in relational form only by creating multiple rows in the table. Therefore the FACE table has to consist of three columns, i.e. a column for the identifier of the face, a column giving indication about the order and the number of the nodes in a face, and a column for the identifiers of the nodes. Thus each FACE is linked to the identifiers of the nodes and not to the co-ordinates. The SQL statements creating NODE and FACE tables are given below:

```
create table NODE (Node_ID number(5), XC number (12), YC number (12), ZC number (12));
create table FACE (Face_ID number(5), SEQF number (3), Node_ID number (5));
```

The LINE table contains columns for the six co-ordinates of the lines and identifier of the face that the line feature belongs to. Each line feature is thus represented by one row in the relational table. The SQL statement to create this table is:

```
create table LINE (Face_ID number(5), X1 number (12), Y1 number (12), Z1 number (12), X2 number (12),
Y2 number (12), Z2 number (12));
```

Pure relational representation is not appropriate for object-oriented models. The needed information for an object has to be assembled from the different records in tables, which usually is slow due to required extra operations. Oracle DBMS offers a way to overcome this disadvantage by creating *object-oriented* views. The employment of object-oriented views give advantages in several directions: 1) the view is processed entirely on the database level that results in significantly fewer SQL statements and thus round trips (query-respond); 2) the data can be extracted from a single view table instead of writing complex joins to get data from multiple tables; 3) the objects in views does not place any restrictions on the characteristics of underlying mechanism. All this is expected to speed up the traverse of the relational tables. Therefore we have created an object type *vrml_export* (that encompasses the data for the VRML scene graph, i.e. the format in which the data will be delivered to the subsystems of UbiCom) and object views using this type:

```
create type VRML_EXPORT as object (Face_ID number (5), SEQF number (3), Node_ID number (5),
        XC number(12), YC number (12), ZC number(12));
```

```
create view VRML of VRML_EXPORT with object identifier (Face_ID) as
        select FACE.Face_ID, SEQF, NODE.Node_ID, XC, YC, ZC
        from FACE, NODE, POLYHEDRON
        where POLYHEDRON.Fase_ID=FACE.Fase_ID and FACE.Node_ID=NODE.Node_ID
```

5.2 Object oriented implementation

The relational implementation of one-to-many relationships has the disadvantage of storing some extra data for representing the relationship (more columns and more records). This often leads to significant increase of the database size. In our case, the column SEQF in the FACE table is a way out of storing the one-to-many relationship. One object is represented by a number of rows, i.e. the column Face_ID contains the same value for one object. The object-oriented approach offers more flexible representations of such relationships. The basic difference is that an object can be stored in a row or a column and can be retrieved by referencing to only one row or column. In Oracle, we can have row objects and column objects. The row objects are stored in an object table that practically is very similar to the relational table but allows an additional object identifier column and index. The object identifier is automatically generated and indexed for efficient lookups. The row representation of objects is not explored yet.

For our spatial model, the column representation is quite appropriate. The data for low-dimensional object (used to describe the higher dimensional object) can be represented in one column and thus the number of rows will be reduced to the actual number of the higher dimensional object. This will allow more compact representation and hence reduction of the database size and the number of rows to be traversed. The one-to-many relationship is represented in two ways by *varrays* or by *nested tables.* The two different representations are given bellow.

Varrays:

```
create type NodeArray AS varray (30) OF number (5);
create table FACE_A  (FID number (5), NUM number(5), NLIST NodeArray);
```

Nested tables:

```
create type NodeTable AS table OF number(5);
create table FACE_T (FID number(5), NUM number(5), NLIST NodeTable) nested table NLIST stored as
NLIST_TAB;
```

6 Tests in PL/SQL

The tests with the four representations are performed with the help of the PL/SQL, a high-level language build-up on the top of Oracle. It is a block-structured language similar to C/C++ and capable of manipulating Oracle data using SQL queries. The greater advantage of PL/SQL utilisation is the ability to incorporate many SQL queries in one block or store them as separate compiled procedures. The procedural calls are then quick and efficient. That reduces network traffic and improves the round trip performance. Furthermore the executable code is automatically cashed and the memory requirements are reduced.

A 3D model of Vienna City is used in the initial tests. The model consists of 1600 buildings (no terrain object) with approximately 20 000 faces and 30 000 nodes. The tests are performed under the several assumptions and simplifications:

- The tests are related only to the extraction of information needed for rendering of virtual objects, i.e. outlines of buildings.

- The relational tables are tested without implementing indexing schemas. Since the experimental data set is relatively small an eventual indexing schema will decrease the differences in performance and will complicate the comparison between the four representations.
- The output of the results is to be represented in VRML format, i.e. three compulsory steps can be distinguished. First, the objects (buildings) belonging to the specified area (with respect to the position of the mobile unit) are clarified. Second, the data to create the VRML document is extracted. Third the retrieved data are structured according to the VRML syntax. Clearly, the VRML geometric representation cannot be obtained straightforward from the 3D topological model. However, the set of data needed for the transformation is standard, i.e. co-ordinates, faces (grouped into objects) and orientation of faces. The performance tests focus mainly the second step assuming that the objects of interest are known. Tests including the third step are still to be performed.

When the objects are specified, the second part of the query can be verbally expressed as "extract all the data needed for the VRML output". Depending on the data structure utilised, the SQL syntax to retrieve these data has different representation. For example, the SQL query for the relational mapping have the syntax:

```
select FACE.Face_ID, SEQF, NODE.Node_ID, XC, YC, ZC
from FACE, NODE, POLYHEDRON
where Poly_ID<User_Defined and POLYHEDRON.Fase_ID=FACE.Fase_ID and
FACE.Node_ID=NODE.Node_ID
order by Face_ID, SEQF
```

The same SQL query performed on the object-view tables has the form:

```
select * from VRML where Poly_ID <User_Defined
```

The object-oriented representations have longer and more complex syntax that will not be given here. The results of the queries are given in Table 1:

Table 1: Experimental results for SQL query extracting 1600 buildings

Database representation	Query
Relational	2.60 sec
Relational with object views	0.05 sec
Object – oriented (varrays)	4.96 sec
Object – oriented (nested tables)	23.28sec

The initial experiments show very good performance for the relational implementation compare to the object-oriented. The better performance of relational schemas can be explained with the smaller number of operations needed to extract the co-ordinates (by a direct join operation) from the NODE table.

7 Conclusions

We have presented a 3D topological structure that aims at providing data for augmented reality, i.e. pose determination and rendering of virtual objects. The proposed structure maintains four abstractions of geometric representation (the ones mostly employed in 3D

modelling) based on two constructive elements (faces and nodes). To be able to provide cheap detailed line features needed for pose determination, the model incorporates a non-topological data type *line* is linked to a face by relationship *belong-to*. Although, the data structure is at very initial stage, the first results are very encouraging: 20 000 polygons can be retrieved in less than a second, which is compatible to the rendering requirements of 2-3 sec (see [8]). In principal, the data to be extracted for the rendering process is expected to be much less (between 50 and 5000 polygons). However, the total size of database can range from few millions to few hundred millions of polygons. For example, the national 2D topographic map of Netherlands contains about 30 million line objects (see [14]). This number might increase three to four times in residential areas. Search in such large databases requires efficient indexing. In near future, the research on the model will concentrate on an appropriate indexing schema, as well.

Still more experiments are needed to clarify the organisation of the persistent data in the database. In a month time the same tests will be carried out with the experimental 3D model of UbiCom area that is expected to be much larger than the 3D model of Vienna City. Currently, the SQL queries are executed from the Oracle high-level language that cannot be integrated in the UbiCom architecture. One of the first steps is developing of C/C++ based modules for incorporating the PL/SQL commands.

References

1. Christensen, H.I., N.O. Kirkeby, S. Kristensen, L. Knudsen, E. Granum, 1994, Model-driven vision for in-door navigation, Robotics and autonomous systems, 12, pp. 199-207
2. Harris, C., 1992, Geometry from visual motion, Active vision, A. Blake and A. Yuille (eds.), Chapter 16, The MIT press, Cambrige, Massachusetts, pp. 263-284
3. Kofler, M. and M. Gruber, 1997, Toward a 3D GIS Database, GIM, Vol. 11, No. 5, pp. 55-57
4. Kozaka, A. and A. Kak, 1992, Fast vision-guided mobile robot navigation using model-based reasoning and reduction of uncertainties, CVGIP: Image understanding, Vol.56, No. 3, November, pp. 271-32
5. Lindstrom, P. D. Koller, W. Ribarsky and L. Hodges, 1996, Real-time, continuos level of detail rendering of height fields, SIGGRAPH'96, New Orleans, Louisiana, USA, pp. 109-118
6. Molenaar, M., 1990, A Formal Data Structure for 3D Vector Maps, Proceedings of EGIS'90, Vol. 2, Amsterdam, The Netherlands, pp. 770-781.
7. OpenGIS specifications, 2000, available on http://www.opengis.org/techno/specs.htm
8. Pasman, W., A. van der Schaaf, R.L Lagendijk and F.W. Jansen, 1999, Low latency rendering for mobile augmented reality, Proceedings of the ASCI'99, 15-17 June, Heijen, the Netherlands, pp. 372-376
9. Persa, S and P. Jonker, 1999, On Positioning Systems for Augmented Reality Applications, Handheld and Ubiquitous Computing 1999, Springer Lecture Notes in Computer Science 1707
10. Pilouk, M., 1996, Integrated Modelling for 3D GIS, PhD thesis, ITC, The Netherlands, 200 p.
11. Pigot, S., 1995, A Topological Model for a 3-Dimensional Spatial Information System, PhD Thesis, University of Tasmania, Australia, 228 p.
12. Rolland, J.P., Y. Baillot and A.A. Goon, 1999, A survey of tracking technology for virtual environments, Augmented Reality Wearable Computer, Bardfield and Caude (eds.), Chapter, Mahwah NJ
13. Ubicom project, 2000, available on http://bscw.ubicom.tudelft.nl/
14. van Oosterom, P., 1997, Maintaining consistent topology including historical data in a very large spatial database, Auto Carto 13, April, Seatle WA, pp.
15. Verbree, E., G. van Maren, R. Germs, F. Jansen and M.J. Kraak, 1999, Interaction in virtual world views—linking 3D GIS with VR, Int. J. Geographical Information Science, Vol13, no 4, pp. 385-396
16. Zillich, M., D. Legenstein, M. Ayromlou and M. Vincze, 2000, Robust object tracking for robot manipulation and navigation, Proceedings of the XIX congress of ISPRS, Com. V/12, pp. 951-958
17. Zlatanova, S., 2000, 3D GIS for urban development, PhD thesis, ITC publication 69, Enschede, The Netherlands, 222 p.