3D MODELLING FOR AUGMENTED REALITY

Siyka Zlatanova
Department of Geodesy, Delft University of Technology
Thijsseweg 11, 2629 JA, Delft, The Netherlands
s.zlatanova@geo.tudelft.nl

KEYWORDS 3D topology, 3D model, spatial objects, 3D visualisation, 3D GIS, real-time

ABSTRACT
This paper addresses an approach for storing 3D spatial objects with respect to the requirements of an outdoor augmented reality application. The presented 3D data structure focuses spatial objects of urban areas, as the aim is support of 3D topology, high level of details and appropriate performance. To satisfy these contradictory requirements, only the outlines of spatial objects are organised in a 3D topological model. Details on façades (e.g. doors, windows) are modelled as line features. An explicit relationship "line on face" indicates the link between the lines and the corresponding façades. The paper concentrates on the 3D topological data structure, which a typical example of so called boundary representations. The 3D data structure is implemented in Oracle 8*i* DBMS using three different approaches, i.e. relational, relational with object views and object-relational. The tests have showed that the proposed data structure can be tuned to meet the needs of real-time rendering and positioning for augmented reality. We consider our results a promising step toward extending 3D GISs to serve real-time applications.

1 INTRODUCTION

Augmented reality is a mixture of reality and virtuality that allows real world to be augmented with additional information. Virtual objects (text or graphics) are visualised in the field of view via special transparent glasses. The user can observe the surrounding world and the virtual objects simultaneously (see Figure 1). Until recently, the augmented reality application were restricted to indoor applications e.g. surgery, inspection of hazardous spaces. With the advances of the computer and vision technology, augmented reality systems attempt to leave the world of indoor applications, which rises new challenging topics for research. Among them, structuring and database organisation of the 3D model required for positioning and visualisation of virtual objects is most pressing. Augmented reality aiming at outdoor urban applications (e.g. rescue operations, utility management, urban development, guided navigation) needs a 3D model of size comparable to a town, i.e. thousands of houses, streets, parking lots, etc. For example, the national 2D topographic map of The Netherlands contains about 31 million line objects (see [12]). In residential areas this number might increase three or four times for the corresponding 3D model. Such an application faces all the problems in processing and maintaining large 3D data sets.

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 four 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, then the 3D re-construction procedures are briefly explained and finally some initial experiments within Oracle database are reported. The research is a part of the interdisciplinary UbiCom project carried out at Delft University of Technology, The Netherlands (see [11]).

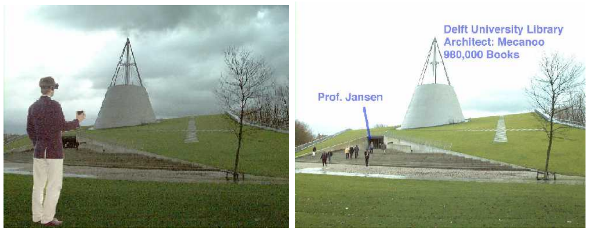

Figure 1: A person with mobile equipment and the observed view

2 REQUIREMENTS

Discussions related to the content and the structuring of data in 3D GIS can be found in many publications on 3D GIS (see [2], [3], [8], [13], [15]). Therefore, in this paper, we focus the specific requirements to the 3D model with respect to the vision system intended in the UbiCom project.

The 3D model is to be used for two critical subsystems of the system architecture, i.e. positioning and visualisation (rendering) of virtual objects. The expected types of data retrieved from the database are different for both subsystems.

- **Line features**. The positioning system in UbiCom relies on line features (straight lines) supplied by the database. The term *positioning* refers to determining the accurate location of the user (i.e. the person using the system) in the real world. The movement of the user is followed (tracked) by mobile equipment (GPS, accelerator and inertial system) that provides an approximate positioning at range of 2-10 meters. The accurate positioning is to be achieved by establishing the correspondence between lines extracted from video images (provided by the video camera that is also a part of the mobile unit) in real-time and lines retrieved from the 3D model (see [8]). The success of the line matching (and thus the accurate positioning) is closely related to the amount of details maintained in the 3D model.
- **3D topology**. In contract to positioning, the visualisation of virtual objects requires no details but correct shape and orientation of the real objects. Since the virtual object is "mixed" with the real world, it is very likely a real object to appear between the virtual one and the user. In this case the real object is said to be an *occluder* of the virtual object (see [7]). The user should not see the occluded parts of the virtual object. In other words, the rendering engine has to be able to compute which parts of the virtual object are not visible in that particular moment and remove them from the scene. The computations require the accurate location of the user (provided by the positioning system) and the 3D model of the real objects (retrieved from the 3D GIS). The 3D model has to ensure connectivity and continuity.
- **Performance**. The mobile equipment is capable of tracking the movement in certain period of time without reference to the database. This period depends on the speed of walking and the movements of the head. The estimation is that every 6-7 sec the system has to receive new data sets from the database. Within this time (called *lag* of the system), a query has to be sent to the database, processed and the retrieved data has to be transmitted back to the mobile unit. The performance of the database is one of the critical issues influencing the lag of the system.
- **Vector representation.** Since both rendering and positioning system use vector representation, we concentrate on vector data structures.
- **VRML**. The Virtual Reality Modelling Language was chosen as a standard to exchange data between the individual subsystems. The language provides a full and flexible

description of the scene (objects, geometry, colours), which is very appropriate for the rendering system.

- **Accuracy.** The accuracy of the re-constructed 3D model plays an important role in the entire process. All the objects that are approachable by the user within few meters distance have to be reconstructed with an accuracy that corresponds to the accuracy of real-time extracted lines from the video images. The user might get as close to an object as one centimetre may be of significance. Indeed, aiming at centimetre accuracy of the objects is a very expensive and practically an unrealistic requirement. However, certain parts, elements or section of real objects have to ensure accuracy ranging from few centimetres to a decimetre. Doors and windows at street level, balconies at lower floors, statues, paths, tiles (see Figure 9), etc. are examples of such elements. The re-construction methods applied to achieve such precision are given in section 4.

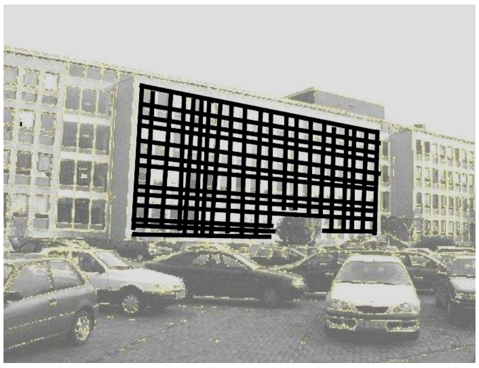
Figure 2: Geometry details represented by line features (the façade in the middle)

Analysis of the requirements reveals the complexity of the issue leading even to a contradiction, i.e. high level of details and maintenance of 3D topology. Apparently, the construction and maintenance of complete 3D topological model (including windows, doors, etc.) is time and effort expensive and therefore inappropriate. One approach (utilised in our project) is a clear discrimination between the types of data needed for the two subsystems. The line features (expected from the positioning subsystem) are kept as straight lines (see Figure 2). The outlines of the 3D objects are considered a separate data set and organised in a 3D topological data structure (see Figure 3). Thus the amount of data to be topologically maintained can be significantly reduced that contributes to the simplification of many spatial operations and thus to the speeding up the data retrieval. Furthermore, the two data sets are linked by explicit reference between line features and façades. 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 takes care relationships “a line belong to a face” to be created and explicitly stored in the database.

3 3D DATA STRUCTURING

The research in 3D data structuring attracts a lot of attention in the last several years. The focus is basically on 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 [10]). 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. The disadvantage is the size of the database that grows tremendously with the complexity of the model. Many reported 3D models give priorities to topological models that maintain objects (i.e. an *explicit* description of objects). More details on data structures of this group can be found in [3], [5], [9], [15]. The major problem of such 3D models is that a few of them are tested on large data sets under real-time requirements.

An intensive work on clarifying guidelines for developing GIS and database software (maintaining 2D, 3D topology) is carried out as well. OpenGIS specifications are one of the commonly accepted standards (see [6]). 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 topological models proposed for implementation consider mostly the 2D world.

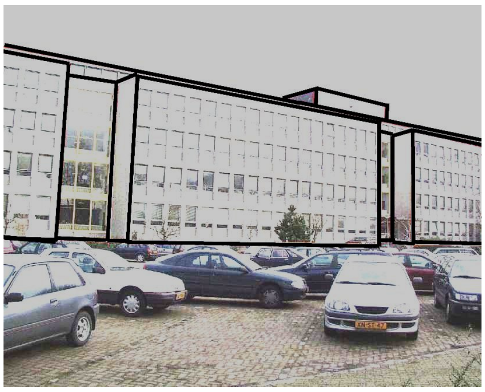
Figure 3: Outlines of 3D objects

Bearing in mind the requirements to the model delineated in the previous section and utilising recent achievements in 3D GIS research, we propose a 3D topological model extended to provide data to augmented reality application. The proposed 3D model is a typical boundary model with explicit description of objects. To represent its *geometric properties* related shape, size and position, a spatial object can be associated with four abstractions namely *point*, *linestring*, *surface* and *polyhedron* (see Figure 4). The notations of the four abstract objects correspond to the ones accepted in the OpenGIS specifications. 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 are built of smaller, simpler elements, called *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 maintained as wheel. The orientation of the faces is anticlockwise looking at the objects (e.g. buildings) from outside. 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).

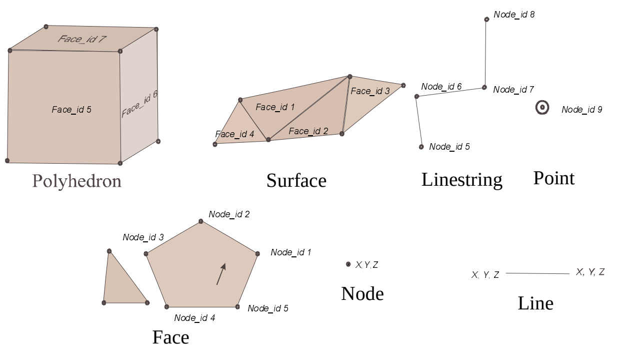

Figure 4: Examples of spatial objects to be supported

Every 3D object can be represented by its *thematic* and *physical* characteristics as well. For example the spatial object building has a year of building, owner and usage that is referred to as thematic characteristics. Physical properties are related to surface reflectance that determines the colour and texture of the objects. Although the model is potentially capable of maintaining thematic and physical characteristics, these aspects are not discussed here. Since the rendering system visualises virtual objects (real objects are considered in case of occluding), colour and texture of real objects are not of interest and are not to be maintained. More details related to physical properties of a spatial object can be found in [15].

The new element in the 3D topological model is the organisation of the data for positioning. The line features are encapsulated 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 IFO semantic data model is used to represent the spatial relationships between all the objects. The motivation is based on the fact that the IFO semantic model uses object-oriented notations to represent objects and their relationships and provides a mechanism to map them into a relational model. Three kinds of *object-types* (abstract, printable and derived) are distinguished by the model (see [1]). An abstract type is an object that cannot be shown printed, mapped, etc., e.g. a building, a street. A *printable* type of object is an object that can be printed, shown, etc. e.g. co-ordinates of a building or a street. The objects that are composed of some printable objects or other atomic objects are *derivable*. From atomic (printable or derivable) objects complex objects can be derived by *aggregation* and *grouping*. The principle difference is that aggregated objects contain elements from different types, while grouped objects contain objects only from one type. *IS-A* relationships are utilised to define sub-types (specialisation) and super-types (generalisation) of objects. Figure 5 shows the graphical notations for the objects, relationships and principles of construction.

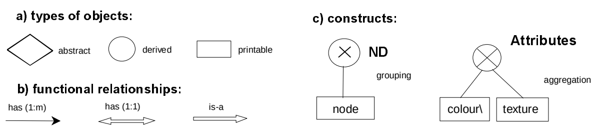

Figure 5: Notations of IFO model

Applying these graphical notations we obtain the schema of the model shown on Figure 6. The four abstract objects named *geometric objects* (GO), one explicit spatial relationship (GR) and two *constructive objects* (CnsO) attempt at representation of all the objects.

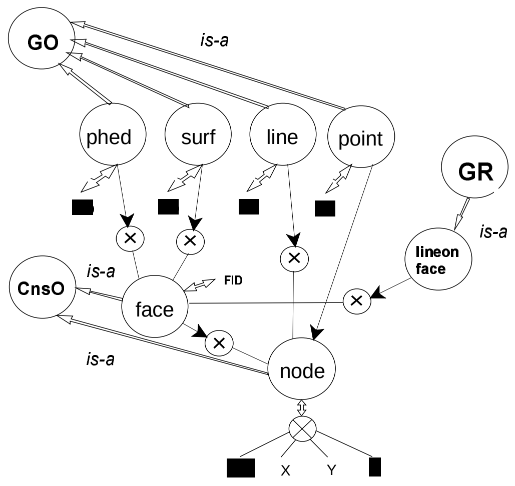

Figure 6: The proposed 3D topological model

The topological model is similar to the ones presented in [3], [5] and [9] but differs in number of construction elements, i.e. only two. The 1D-cell, often called *arc* or *edge* (see [5], [9], [10]), 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 a significant facilitation. Such representation, i.e. without arcs, has shown speed acceleration in many data retrieval operations (see [15]). Moreover the computational complexity of writing the VRML file is largely reduced, since the representations of polyhedron and surface are similar to description of irregular shapes in the VRML node *IndexedFaceSet* (see [14]).

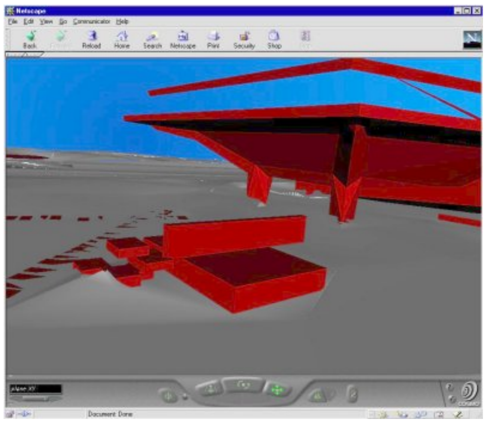

Figure 7: Modelling of complex spatial objects

4 DATA COLLECTION

The 3D reconstruction of urban areas is still a rather complex process involving quite a lot of time and manual interactions. Several different methods are applied to provide data with the appropriate accuracy and resolution and to obtain the 3D topology. The utilised methods can be subdivided into three major groups: manual, semi-automatic and automatic. Whilst manual and semi-automatic methods are used in the case of relatively limited amounts of object elements with high accuracy and 3D topology are required (i.e. to build the topological model), automatic methods are applied for collection of many elements with non-topological organisation (i.e. the line features). Furthermore objects (e.g. buildings) with complex shape and variety of details (balconies, overhanging roofs) are modelled manually from images taken at street level. The images are processed in a 3D modelling software (i.e. *PhotoModeller*) that ensures accuracy and resolution within required values. A snapshot from *CosmoPlayer* shows some of the reconstructed objects of this group (see Figure 7).

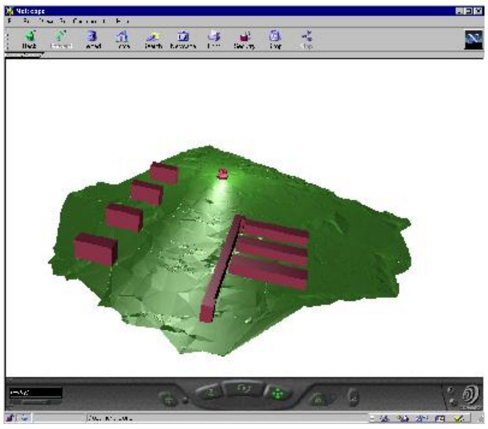

Figure 8: Modelling of buildings without overhanging roofs

Simpler objects such as buildings with flat or gable roofs are reconstructed applying semi-automatic methods. For example the reconstruction of buildings with flat roofs and roof outlines that can be projected onto the footprints (i.e. lack of overhanging roofs) are reconstructed by a procedure utilising existing DTM and manually digitised roof outlines (see Figure 8). The walls and the corresponding 3D topology are obtained fully automatically (see [16]). Terrain objects as streets, bicycle or pedestrian paths, parking lots, etc. are also integrated in the model as 3D objects. Quite often terrain objects are the only clearly visible objects that can be used for positioning (see Figure 9). In our case, we obtained the terrain objects by overlapping dense laser data (0.60m x 2.00m) with data from the topographic base map of the Netherlands (scale 1:1000). Every 2D point of the topographic map obtains Z-coordinate by interpolation on TIN created from the filtered laser data.

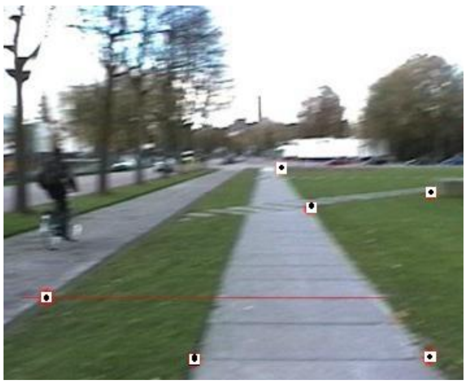

Figure 9: An image from the video camera with marked points needed for positioning

The procedure for 3D line feature extraction is an integration of algorithms for edge detection (on two or more images) and 3D line computation using knowledge based methods. The work on the algorithms for 3D line extraction as well as the final integration of partially reconstructed 3D models from the different approaches is in progress.

5 IMPLEMENTATION

The argument about advantages and disadvantages of developing a database system from "scratch" may give preferences to either of the approaches. A newly design database system gives the freedom to create operators that suit the tasks of the application in a better way. The utilisation of commercial database systems brings advantages in few directions: standard data definition and data manipulation (e.g. SQL) languages, a variety of indexing schemas, optimisations in maintenance of large data sets. Computer graphics specialists often consider commercial databases rather slow for real-time applications and develop their own databases (see [4]). However, the technology and software achievements in the last several years in the field of database research challenge with improved functionality and performance that may be sufficient for any application. Therefore we have decided on implementation of the model described above in a commercial database, i.e. the relational database Oracle 8*i*. 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. The IFO model can be converted directly into a relational data model according to the rules:

- abstract object types are represented as relational tables (or relations)
- derived object types are represented as relational tables or domains (if they are atomic objects to a complex object)
- printed object types are represented as domains
- *IS-A* relation is propagated until a printed object type.

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, LINEONF, SURF and PHED. 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 this spatial object. Since the remaining tables have very similar structure, only the FACE table is explained. The relationship between a face and constituting nodes is one-to-many (1:m) which can be represented in a relational form by creating multiple rows or columns in the table. Since the multiple-column representation leads to a table with large amounts of zero fields, we give preferences to multiple-row representation. Therefore the FACE table has to consist of three columns, i.e. a column for the identifier of the face, a column indicating the order and the number of the nodes in a face, and a column containing the identifiers of the nodes. Thus each FACE is linked to the identifiers of the nodes and not to the co-ordinates. The LINEONF 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.

Let us adopt the following representation for a relational table: R = ({A_1, …, A_n}, {PK}) where R = () is the relational table, {A_1, …, A_n} are domains (columns) and {PK} is the primary key. Then we obtain the following relational mapping:

NODE ({NID,X,Y,Z},{NID})
FACE ({FID,SEQF,NIDF},{FID})
POINT ({PID, NID},{PID})
LINE ({LID, SEQL,NIDL},{LID})
SURF ({SID,SEQS,FIDS},{SID})
PHED ({BID,SEQB,FIDB},BID})
LINEONF ({LFID,FID,X1,Y1,Z1,X2,Y2,Z2)},{LFID})

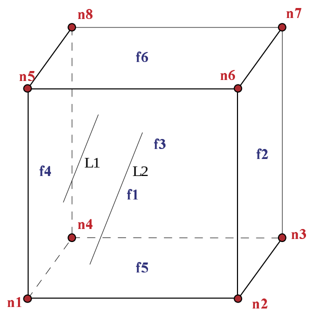

Figure 10: An example of 3D object with line features on face f1

For example, a cube that consists of six faces (f1…f6), eight nodes (n1…n8) and two lines (L1 and L2) on the face f1 will be represented by the following records in the tables FACE, LINE and PHED:

FACE			PHED		
Fid	Seqf	nid	bid	Seqb	Fid
1	1	1	1	1	6
1	2	2	1	2	5
1	3	6	1	3	1
1	4	5	1	4	2
…	…	…	1	5	3
6	1	5	1	6	4
6	2	6			
6	3	7			
6	4	8			

LINE							
Lid	Fid	X1	y1	Z1	x2	y2	z2
1	1	…	…	…	…	…	…
2	1	…	…	…	…	…	…

A pure relational representation is well known to be inappropriate for object-oriented models due to rather large amount of duplicated data to represent groups and aggregations of objects (i.e. one-to-many relationships). In our case for example, 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 as FID is repeated in each record. Oracle DBMS offers two options to overcome this disadvantage i.e. *object-oriented views* and *object-relational implementation*.

5.2 Object-oriented views

Object-oriented views do not change the relational type of the data structure but provide certain facilitation in maintenance of objects. The employment of object-oriented views gives 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 for multiple tables; 3) the extracted data can be straightforward used by object-oriented languages for further processing. Views are especially appropriate for retrieval of standard data sets, e.g. the geometry needed for composing a VRML file. To assess the performance of the object-oriented views, we have created an object type *vrml_export* (that contains the data for the VRML scene graph) and an object view using this type. The syntax of the SQL commands is given bellow:

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

create view VRML of VRML_EXPORT with object identifier (FID) as
select FACE.FID, SEQF, NODE.NID, X, Y, Z
from FACE, NODE, PHED
where PHED.FIDB=FACE.FID and FACE.NIDF=NODE.NID
ordered by FID, SEQF
```

5.3 Object-relational implementation

A step further is the object-relational implementation. While the object-oriented views provide a mechanism to encompass data from relational tables in an object, the object-relational implementation allows an object to be stored in a relational table. There are basically two approaches, i.e. an object can be stored in a row or in a column. The retrieval of the object then is based on referencing to only one row or column. 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.

Our 3D topological model make use of the column representation. The data of an object of lower dimension (used to describe the higher dimensional object) is stored in a single column. This means that the number of rows in the object table will be reduced to the actual number of the higher dimensional object. The advantage is compact representation and hence a reduction of the number of rows to be traversed.

Object-oriented implementation is a two-step procedure, i.e. creating objects and creating tables. We use two extended Oracle data types that are intended for representing the one-to-many relationship, i.e. *varrays* and *nested tables.* While *varrays* are recommended for objects which elements are always retrieved in their completeness, nested tables are said to be suitable for cases that require accessing and retrieving individual elements of an object. According to Oracle manuals, the better performance is the major advantage of *varrays* compare to nested tables. Although most of the operations (i.e. retrieval of geometry and spatial relationships) can be classified as retrieval of objects with their complete set of elements, we utilised both data types. The syntax of the commands is given bellow:

Varrays:
```
create type NodeArray AS varray (30) OF number (5);
```

Nested tables:
```
create type NodeTable AS table OF number(5);
```

Utilising the newly created data types NodeArray and NodeTable, the FACE object can be stored in the database in two ways as follows:

FACE_A ({FID, NUM, NLISTA}, {FID}), where NLISTA is of data type NodeArray

FACE_T ({FID, NUM, NLISTT}, {FID}), where NLISTT is of data type NodeTable

Note that a second column (giving the actual number of nodes per face) is introduced in both tables. Similar tables are created for LINE, SURFACE and POLYHEDRON, e.g. the tables using *varrays* are given bellow:

LINE_A ({LID, NUM, NLISTA}, {LID})
SURF_A ({SID, NUM, NLISTA}, {SID})
PHED_A({BID,NUM,NLISTA},{BID})

5.4 Using the spatial data types of Oracle

Currently Oracle Spatial has implemented only the geometry object model as it is specified in Open GIS specifications. This means that all the supported *shapes* (i.e. geometric objects in Oracle Spatial) are represented by their co-ordinates. Consequently, if two objects share co-ordinates, they are stored two times in the database. The topological operations are then explicitly computed when it is necessary. Furthermore the implemented topological operations are limited to cases in 2D space. Hence, the utilisation of the spatial objects for our model is rather limited. The only straightforward possibility is to represent NODE object by the Oracle *shape* SDO_POINT. The description of the *shape* SDO_POINT and NODE are identical. The advantages and disadvantages of such representation are to be further explored.

6 OPERATORS TO RETRIEVE THE GEOMETRY

The model presented in the previous sections does not have a direct link to the geometry of the objects, i.e. the co-ordinates of objects have to be derived. The operators to convert topology to geometry depend on the geometric model and the implemented mapping. It is possible to utilise the geometric model for spatial objects defined in OpenGIS specification. However, within the UbiCom project there is an agreement on adopting the geometric model of VRML. This model (known also as a *scene graph*) preserves the topology of each individual object or group of objects. In this respect, the operators to create a VRML file can be considered as operators for transformation from one topological data structure to another and therefore they are more sophisticated. For example, the SQL queries to retrieve the geometry (co-ordinates) of a POLYHEDRON with identifier 23 from relational and object-relational (varrays) mappings are:

Relational:
```
select FID, SEQF, NID, X, Y, Z
from FACE, NODE, PHED
where PHED=23 and PHED.FIDB=FACE.FID and
FACE.NIDF=NODE.NID
ordered by FID, SEQF
```

Varrays (part of a PL/SQL script):
```
inid arra:= arra(0);
jj number (5);
t1 number(5);
t2 number(5);
begin
…
 select NUM, NLIST into t1, inid from FACE_A
   where FID=23;
 jj:=1;
 while jj< t1 loop
   select NID into t2 from NODE
     where inid(jj) = nid;
   jj:=jj+1;
 end loop;
…
```

The result is a list of co-ordinates structured according to the order of the faces and order of the nodes in a face. To create the VRML file, the data set obtained with these queries has to be further structured according the VRML model. In our example, two options are possible to represent the faces constructing the POLYHEDRON 23 in VRML, i.e. they can be stored as individual faces or as a part of one polyhedron. The first option is simpler and can be derived directly from the SQL query. The second representation requires control of duplicated co-ordinates and the correct ordering of the corresponding co-ordinates in the description of the faces (i.e. in the VRML node *coordIndex*). Currently, we concentrated on development of operators that create a VRML file consisting of objects (i.e. not individual faces). Since standard SQL statements cannot perform this operator, the computations have to be completed with the help of a host language (PL/SQL, C++, Java). At this stage, we have used PL/SQL, i.e. the script language provided by Oracle.

A number of tests aimed at clarification of the best mapping and fastest operators to retrieve the needed data. For the purpose the first tree mappings discussed in the paper are implemented and populated with data. The corresponding VRML creators are tested for performance. Although the data structure is at very initial stage, the first results are very encouraging: 20 000 faces (part of 1600 buildings) can be retrieved in less then 23 seconds for representation with nested tables and less then a second for the relational representation and object-oriented views. The conditions of the test and the data sets are reported and discussed in more details in [17]. In principal, we expect less data to be extracted for rendering, i.e. the objects that are visible only from the current position of the user. These data are estimated in the range between 50 and 5000 faces if spatial search is applied. This is indication for a possible further improvement of the obtained results.

7 CONCLUSIONS

We have presented a 3D topological structure that provides data for a real time application i.e. serves two tasks (pose determination and rendering of virtual objects) that require real-time commuting. 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” details in terms of line features, the model incorporates non-topologically organised data, i.e. lines. Explicit relationship links a set of lines to a face. To be able to achieve the required accuracy and to build the 3D topology, a number of 3D reconstruction methods are applied. The model is implemented in relational database Oracle utilising relational and object-relational mappings. Several operators to create a VRML file are created and tested.

The experiments clearly show that a 3D topological model can be adopted for an augmented reality application. The performance of the mappings in relational database drops bellow the required 6 seconds. The methods utilised in the 3D reconstruction ensure accuracy of few decimetres that is agreed to be sufficient for the positioning system. Therefore we consider the results reported in this paper a successful step toward a 3D GIS supplying data for a real time application.

Still more experiments are needed to clarify the relational mapping that will assure the best performance. Currently, the SQL queries are executed from the Oracle high-level language that cannot be integrated in the UbiCom architecture, i.e. C++ modules have to be developed and further tested. Location and efficient spatial search in such large databases can not be performed without appropriate spatial indexing. One of the directions for further research within the project is related to developing a set of specific operations than will reduce the amount of data transmitted to the vision system. Examples of such operators are determination of the area of interest (using approximate positioning obtained by the mobile equipment), back-face culling (to eliminate invisible faces) and a variety of line filters for retrieval of line features.

References

[1] Abiteboul, S and R. Hull, 1987, IFO: A formal semantic database model, ACM Transactions on Database Systems, Vol.12, No.4, pp. 525-565

[2] Bodum, L., I. Afman and J. Smith, 1998, Spatial Planning moves out of the flatlands, AGILE, 23-25 April, Enschede, The Netherlands, CD-ROM

[3] de la Losa, A. and B. Cervelle, 1999, 3D topological modelling and visualisation for 3D GIS, Computer& Graphics, Vol.23

[4] Kofler, M. and M. Gruber, 1997, Toward a 3D GIS Database, GIM, Vol. 11, No. 5, pp. 55-57

[5] Molenaar, M., 1990 A Formal Data Structure for 3D Vector Maps, Proceedings of EGIS’90, Vol. 2, Amsterdam, The Netherlands, pp. 770-781

[6] OpenGIS specifications, 2000, available on http://www.opengis.org/techno/specs.htm

[7] 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

[8] 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

[9] Pilouk, M., 1996, Integrated Modelling for 3D GIS, PhD thesis, ITC, The Netherlands, 200 p.

[10] Pigot, S., 1995, A Topological Model for a 3-Dimensional Spatial Information System, PhD thesis, University of Tasmania, Australia, 228 p.

[11] Ubicom project, 2000, available on http://bscw.ubicom.tudelft.nl/

[12] van Oosterom, P., 1997, Maintaining consistent topology including historical data in a very large spatial database, Auto Carto 13, April, Seatle WA, pp.

[13] 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

[14] The Virtual Reality Modelling Language, 1997, available on http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm

[15] Zlatanova, S., 2000, 3D GIS for urban development, PhD thesis, ITC public. 69, Enschede, The Netherlands, 222 p.

[16] Zlatanova, S., J. Paintsil and K. Tempfli, 1998, 3D object reconstruction form aerial stereo images, Proceeding of the 6th International Conference in Central Europe on Computer Graphics and Visualization'98, 9-13 February, Plzen, Czech Republic, pp. 472-478

[17] Zlatanova, S. and E. Verbree, 2000, A 3D topological model for augmented reality, Proceedings of the Second international symposium on MMSA, 9-10 November, Delft, The Netherlands, pp. 19-26

BIOGRAPHY

Siyka Zlatanova:
MSc degree at the University of Architecture, Civil Engineering and Geodesy, Sofia, Bulgaria in 1983 (Geodesy and Photogrammetry). PhD degree at the Graz University of Technology, Graz, Austria. Currently, a post-doctoral researcher at the Delft University of Technology, Delft, The Netherlands. Research field: Spatial Data Handling, 3D topology, 3D object reconstruction and visualisation.