3D GEO INFORMATION SYSTEM – WHERE ARE WE?

Alias Abdul Rahman[1], **Sisi Zlatanova**[2], **Volker Coors**[3] and **Chris Gold**[4]

[1]Department of Geoinformatics,
Faculty of Geoinformation Science and Engineering,
Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
alias@fksg.utm.my

[2]GIS Section, Delft University of Technology
Delft, The Netherlands
s.zlatanova@tudelft.nl

[3]Faculty of Mathematics, Computer Graphics and Geoinformatics
University of Applied Sciences,
Stuttgart, Germany
volker.coors@hft-stuttgart.de

[4]School of Computing
University of Glamorgan
Wales, United Kingdom
cmgold@glam.ac.uk

ABSTRACT

GIS evolves from a simple to advanced information system and yet we have seen many situations where these information systems were unable to provide satisfactory solutions and answers to users. More and more users are now looking for advanced and reliable systems. The problem becomes more apparent when it comes to manipulating and handling 3D spatial objects either in standalone, wired or mobile computing environment system. There are a number of challenges and issues still need to be addressed for a more decent 3D spatial information system as we used to have in 2D domain. This paper reviews recent research works toward realizing 3D geoinformation system. Discussions on spatial data modeling including spatial operators, databasing for 3D spatial objects, and utility objects form major part of the paper. The paper also describes research works on 3D Web services and the related applications such as indoor and outdoor navigation as part of solution for city modelling and way finding within building environment. Furthermore, current status of the research and development works on 3D geoinformation will be highlighted and finally, a discussion on future trend on the development of 3D geoinformation system and services.

Keywords: 3D data modeling, 3D geo DBMS, 3D spatial operations, and 3D Web service.

1 INTRODUCTION

Geo information system has evolved from simple applications to applications that demand more complex spatial datasets and situations. Many research works attempt to solve the problems by introducing several new ideas and methodologies for such applications and situations. It has been recognised that in three-dimensional GIS, i.e. 3D GIS, the software or the system should be able to describe the real world situations in a more understandable manner as expected by users. Current research development on this 3D GIS domain is still lacking in term of practical data models, mechanisms and even theories to solve the problem. Lately, we have seen a lot of efforts have been invested by many researchers and software vendors to develop and offer such 3D GIS software or system. However, literature shows that several aspects of 3D GIS remain to be solved and addressed by the community. This paper elaborates our research works for the development of 3D GIS, and it

also thoroughly discusses some aspects of 3D GIS frameworks. The paper also highlights some challenges and related issues on 3D GIS development.

Section 2 discusses object modelling and reconstruction by using recent and accurate datasets (i.e. from LIDAR). We thoroughly discuss 3D geo DBMS in Section 3 and the 3D spatial operations in Section 4. Section 5 describes the 3D Web services, a piece of research work carried out at Fraunhofer Institute of Computer Graphics. Finally, the conclusion of the paper.

2 3D SPATIAL OBJECTS MODELLING AND RECONSTRUCTION

We have successfully addressed this problem by developing a several-stage process. Our starting point is a set of raw LIDAR data, as this is becoming readily available for many areas. This is then triangulated in the x-y plane using standard Delaunay techniques to produce a TIN. The LIDAR values will then show buildings as regions of high elevation compared with the ground. Our initial objective is to extrude these buildings from the landscape in such a manner that they have well defined wall and roof planes. We want to know how to extract building outlines from the triangulation when it is not available from the national mapping department. We do this by superimposing a coarse Voronoi cell structure on the data, and identifying wall segments within each. There are two ways to examine the triangulated interior (roof) data. The first method is to find out the folding axis of the roof, but it may not be suitable for a complex roof. The second is to identify planar segments and connect them to form the final surface model of the building embedded in the terrain. This is done using Euler Operators and Quad-Edges with preserved topological connectivity. This was successfully developed by [Tse and Gold(2001), Tse and Gold(2002)]. In this paper, we will focus on discussing how to extract building blocks from raw LIDAR data and how to find out the roof shape of those building blocks.

2.1 LIDAR as Data Source

ALS (or so called LIDAR) is a new independent technology which is highly automated to produce digital terrain models (DTM) and digital surface models (DSM) Ackermann(1999). It is a laser-based technology to emit and capture the returned signal from the topographical surface. A laser scanning system, a global positioning system (GPS) and an inertial measuring unit (IMU) are the three main units in an ALS system.The laser scanning system is mounted on an aircraft, a helicopter or satellites and emits pulses toward the earth's surface and measures the distance reflected from the earth's surface and other objects on the surface back to the aircraft. IMU and GPS are important for determining the absolute position and orientation of the LIDAR sensors. The Inertial navigation system is used to correct the errors from the pitch, roll and yaw of the plane. GPS monitors the altitude and the flight path of the aircraft which observes the three dimensions data. A high accuracy GPS is installed in the plane and a ground control based station is established. Figure 1 shows an aircraft scanning over a piece of land.

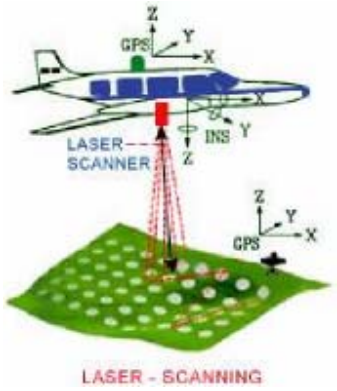


Fig. 1. Airborne laser scanning

LIDAR provides an efficient way to capture 3D data; however it is not easy to extract building information from the data. Much research focuses on extracting building outlines, and they may combine different data sources, for

example photogrammetric data or existing landline data Sohn and Dowman (2003), Sohn and Dowman (2004), Suveg and Vosselman(2004), Vosselman and Dijkman(2001). It does not work if there is no other data available. Our approach is to use LIDAR only to reconstruct the 3D buildings and remodel the roof structure without using any pre-defined models.

Building Blocks Identification

Our method is to identify building blocks from the terrain rather than searching for the building footprints directly. It separates the high-elevation data (building block) from low-elevation data (terrain surface). We make use of the duality and connectivity properties of Delaunay triangulation, and its dual Voronoi diagram. A Delaunay triangulation is created using the original high density LIDAR data (Figure 2). Then we sample it to a lower resolution triangulation (Figure 3. In Figure 3 each big Voronoi cell contains many data points (about 50 - 100), some with only the ground points (low elevation) and some with only the building points (high elevation). Voronoi cells with low and high points are extracted for further modification because building segments can be found in those cells. We are using some made-up data with an L-shaped building to illustrate the method.

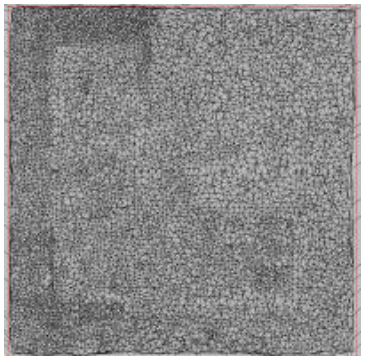


Fig. 2. Raw LIDAR data points

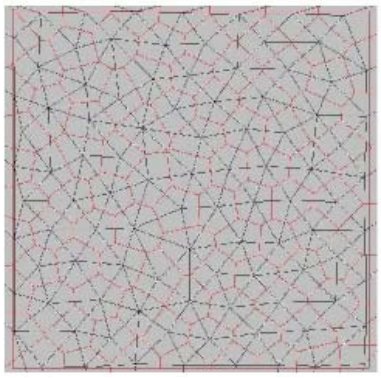


Fig. 3. Lower resolution LIDAR data points

The extracted cells contain low and high points which will be split into two. The direction of the splitting line is found by calculating the eigenvalues and eigenvectors of the 3 x 3 variance-covariance matrix of the coordinates of the points within each cell. The result of three eigenvectors “explain” the overall variance, the left-over and the residue. For example, a wrinkled piece of paper might have the first eigenvector (the highest eigenvalue) oriented along the length of the paper, the second (middle eigenvalue) along its width, and the third (the smallest eigenvalue) “looking” along the wrinkles. Thus the eigenvector of the smallest eigenvalue indicates the orientation of a wall segment, if present, and looks along it. Figure 4 shows the eigenvector with the smallest eigenvalue which shows the orientation of the splitting line between the low and high points.

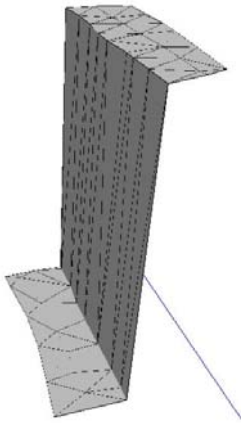


Fig. 4. The eigenvector with the smallest eigenvalue

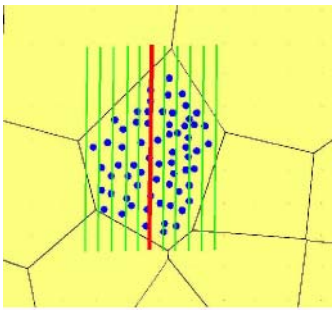


Fig. 5. The thick red line separates the low and high data points

With the orientation of the splitting line, the next step is to find the best location to put the line and split the cell. This is achieved iteratively, by testing various positions of the line parallel to the smallest eigenvector in order to find the greatest difference between the low and the high points.Figure 5 shows a thick line which separates the high points from the low points. In order to minimize the effect of sloping roofs or terrain, only those elevations close to the line are used. If this maximum difference is not sufficiently large then no wall segment was detected. Therefore walls have a specified minimum height and this height difference is achieved within a very few "pixels".

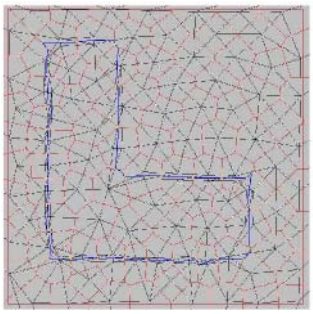


Fig. 6. Building segment in each Voronoi cell

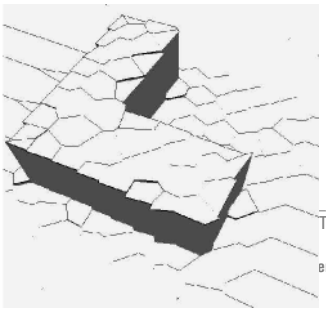


Fig. 7. Vertical Building Walls formed by split Voronoi Cells

We locate the splitting line and add a generator on each side of this line, at the mid-point to split the cell. Figures 6 and 7 show the 2D and 3D view of the split Voronoi cells. A set of "high" Voronoi cells are split and surrounded by "low" ones. In Figure 6 building boundaries are then determined by walking around the cells and connecting the Voronoi boundary segments (the splitting line) or the immediate Voronoi edges to form a closed region. If a closed high region is found, it is considered to be a building. The Voronoi boundary segments are used to estimate the building outline. We use the Voronoi boundary segments which are created with the eigenvector The split Voronoi edges in six groups

2.2 Roof and building modelling

Many systems use pre-defined building models for reconstruction Brenner (1999), Rottensteiner and Briese (2003), but we would like to reconstruct the buildings without any pre-defined models.Two methods are used to remodel the roof structure. The first is simple but works only with simple gabled roofs. The second is more complicated but can solve the problem of the complex roof structure.

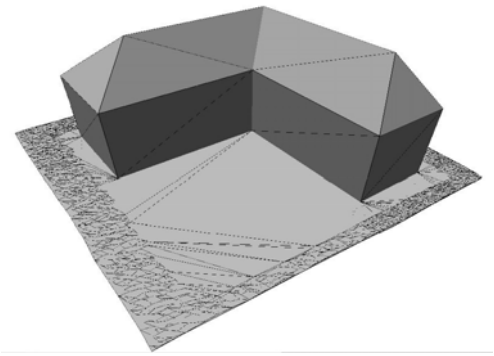


Fig. 8 An extruded L-shape building

3 3D GEOSPATIAL DBMS

DBMS have been traditionally used to handle large volumes of data and to ensure the logical consistency and integrity of data, which also have become major requirements in handling spatial data. For years, spatial data used to be organized in dual architectures consisting of separated data management for administrative data in a Relational DBMS (RDBMS) and spatial data in a GIS. This is to say spatial data has been managed in single files using proprietary formats. This approach could easily result in inconsistency of data, e.g. when deleting a record in the database no mechanism exists to check the corresponding spatial counterpart. A solution to problems of dual architecture was a layered architecture, in which all data is maintained in a single RDBMS. Since spatial data types were at that time not supported at DBMS level, knowledge about spatial data types was maintained in middle ware (Vijlbrief and van Oosterom, 1992). Presently, most mainstream DBMS offer spatial data types and spatial functions usually in an object-relational spatial extend to RDBMS (Zlatanova and Stoter, 2006). Storing spatial data and performing spatial analysis can be completed with SQL queries. Additionally, integrated queries on both spatial *and* non-spatial parts of features can be executed at database level. The spatial data types and spatial operations reflect only simple two-dimensional features, though embedded in 3D space. This support of 3D/4D coordinates allows for alternatives in management of 3D features.

3.1 3D spatial data in DBMS

Providing the spatial extend, DBMS have immediately been challenged by the third dimension. A number of experiments were performed by several researchers to investigate possibilities to store, query and visualize features with their 3D coordinates (Arens et al 2006, Stoter and Zlatanova 2003, Pu, 2005, Zlatanova et al 2002, Zlatanova et al 2004). The good news is: 3D data can be organized in DBMS, retrieved and rendered by front-end applications. However, there is also a bad news: since no 3D data type is currently supported by any DBMS, the user remains self responsible for the validation of the objects as well as for implementing true 3D functionality.
These conclusions, with small variations, are consistent for all mainstream DBMS: Oracle, IBM DB2, Informix, Ingres, PostGIS and MySQL. All of them offer 2D data types (basically point, line, polygon) but support 3D/4D coordinates (except Ingres, which is 2D) and offer a large number of functions more or less compliant with the Open Geospatial Consortium (OGC) standards (see bellow). Most of the functions are only 2D (except PostGIS, which supports few 3D operations), i.e. although not reporting error, they omit the z-coordinate. A bit frustrating is the implementations of spatial functions and operations: it varies with the DBMS. The statement: *select c from b where a < 100*, where *c,a* are numerical data type, can be executed in every DBMS. However if *c* is a spatial data type and *a* is a given distance, the SQL statement becomes depended on the specific implementation. In some cases (e.g. Oracle Spatial), even the names of the spatial data types are not that apparent. Oracle Spatial has one complex data type *sdo_geometry* composed of several parameters indicating type geometry, dimension, and an array with the *x,y,z* coordinates. At present the geometry model of Oracle Spatial is the most supported by GIS and Architecture, Engineering and Construction (AEC) Systems, but there is a strong tendency for changes (e.g. PostGIS is supported by GRASS and ESRI). The text below discusses possible organisation of the 3D real-world features (volumetric, line, point) in current DBMS. Note, the presented approaches are not dependent on the DBMS.

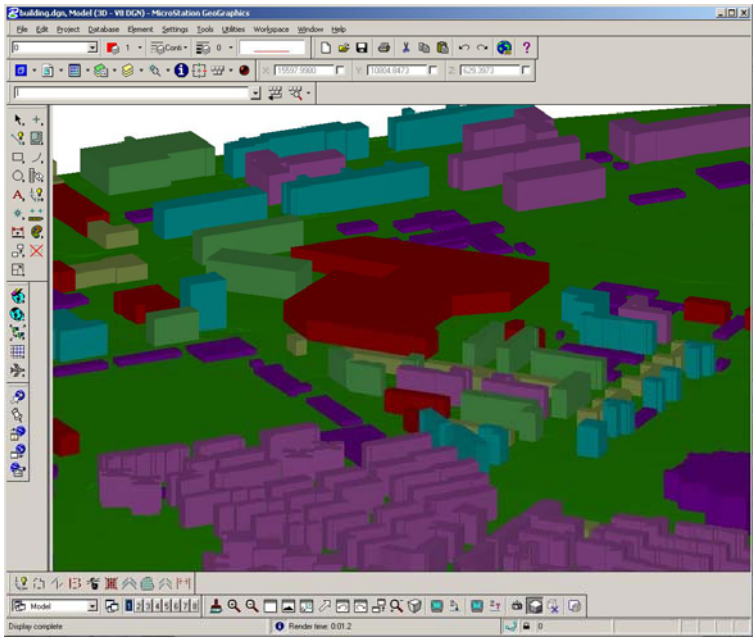


Fig. 9. Visualisation of buildings and surface, represented by simple polygons in Oracle Spatial

3D volumetric objects
Most discussed 3D features are volumetric objects, which can be used for modelling of man-made objects, such as buildings, and natural objects, such as geological formations. To have those managed in the database, the user can choose between: 1) using DBMS data types *polygon* and *multipolygon* or 2) creating a used-defined data type. The three candidates for a simple volumetric object are *polyhedron*, *triangulated polyhedron* and *tetrahedron* (see for definitions next section) and all three can be easily realized with provided data types. Moreover, there is no practical difference in the implementation of the polyhedron and triangulated polyhedron, since a separate *triangle* data type does not exist. Tetrahedron would allow for a bit simpler representation since it has only four triangles.

The first option, i.e. defining a 3D feature as a list of polygons can be realized by two columns in one relational table, i.e. feature_ID and a column for the spatial data type (i.e. *polygon*). If the reality is quite complex, leading to many 3D features with shaped polygons, the DBMS table should be normalized. This means that the

polygons have to be organized in a separate table (containing polygon_ID and a column for the spatial data type) and the 3D feature table should contain the indices to the composing polygons. Clearly, the separate relational table for volumetric objects would be simpler if the volumetric object is *tetrahedron*. It can be organized in a table with finite number of columns: one for the ID of the *tetrahedron* and four for the composing polygons (triangles). In general, such an organization can be seen as a partial topological model; since the 3D object is defined by references to the composing polygons. Fig. 9 is an example of a two-table, polyhedron data storage.

In the second approach, a 3D object is stored using the data type *multipolygon*, i.e. all the polygons are listed inside the data type, which is practically one record in the relational database. This case requires only one table, which may contain only two columns: feature_ID and column for the spatial data type. Various examples of these representations can be found in Stoter and Zlatanova, 2003.

An apparent advantage of the 3D *multipolygon* approach is the one-to-one correspondence between a record and an object. Furthermore the 3D *multipolygon* (compare to list of polygons) is recognized as one object by front-end applications (GIS/CAD). For example, a 3D *multipolygon* is visualised as grouped polygons in Bentley Microstation. However, in case of editing, the group still has to be ungrouped into composing polygons (Zlatanova et al 2002), i.e. the group is not recognised as 3D shape.

Indeed, both representations are not recognized by DBMS as a volumetric object, i.e. they are still polygons and thus the 3D objects cannot be validated. The objects can be indexed as 3D polygons but not as 3D volumetric objects. Spatial operations can be performed, as well, but on the 'flat' polygons, i.e. the z-coordinate will not be considered. Moreover, both representations are highly inefficient in terms of storage space. The coordinates of the points in a volumetric object are repeated at least three times (being part of minimum three neighbouring polygons) either in 3D *multipolygon* record or in the list of *polygons*.

User defined spatial data types can be implemented using different approaches from the simple SQL *create data type* statement, to more complex implementations, using a Procedural Language (PL), Java, C++, etc. The common drawback of such an implementation is impossibility to apply the native spatial functionality (operations and indexing) of DBMS. Moreover the user-defined spatial data types cannot be stored in the same column of the natively supported spatial data types. Visualisation in front-end applications would be possible only by developing individual connections. User-defined spatial data types, however, are very useful for prototyping for approval of new concepts. Two examples of user-defined spatial data types are discussed later in the text.

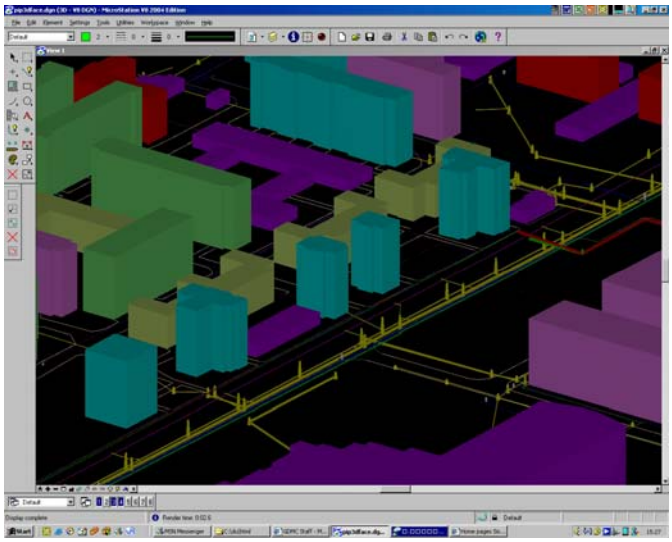


Fig. 10. 3D visualisation of pipelines, organised as lines in Oracle Spatial

3D line objects

Typical examples of 3D line objects are utility networks: pipeline and cable networks. Utility data and systems have been always predominantly two-dimensional. Only recently, investigations have been initiated towards maintaining utility networks in three dimensions. Motivation for this is extended usage of underground space and therefore the apparent need of more sophisticated mechanisms for visualizing several networks in one environment.

Utility networks (represented as lines with 3D coordinates) can be readily managed in DBMS using the supported spatial data types *line* or *multiline*. If required by an application, some point objects (valves, connectors, etc.) can be separately organized as *points.* The only trouble of 3D lines is the visualisation in 3D scenes. It is often recommended 3D lines to be substituted with tiny cylinders when rendered. Indeed, such a substitution can not be justified only for visualisation purposes. Therefore, Du and Zlatanova (2006) suggest keeping the original data as 3D lines and creating 3D cylinders on the fly only for the visualization (Fig. 10).

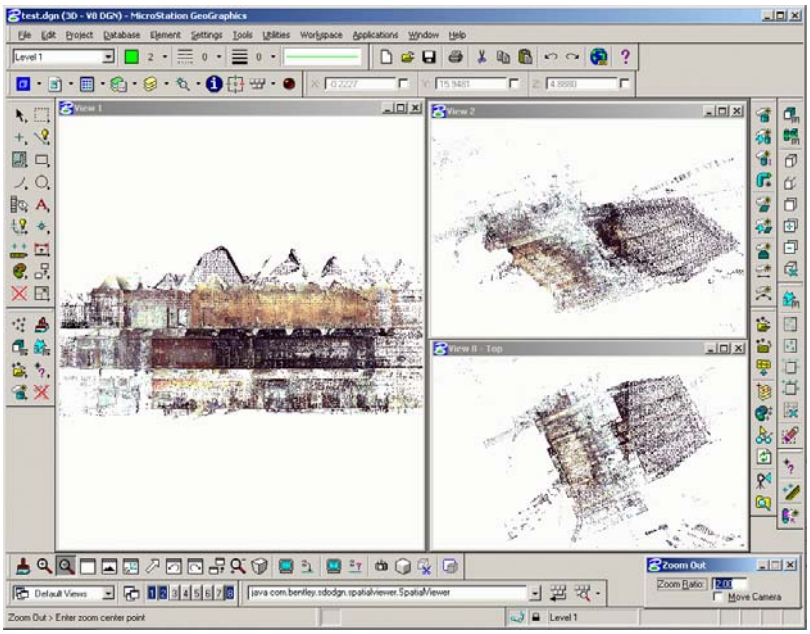


Fig. 11. 3D visualisation of point clouds, managed as points in DBMS

3D point clouds

Until recently, 3D point objects were relatively rare in real-world data sets and a little attention was given on them. But, with the advances of sensor technology, laser scanning techniques become available, which produce large amounts of very specific 3D point data. Theoretically, these points can also be organised in DBMS by either 1) using the supported spatial data types *point* (Fig. 11) and *multipoint* or 2) creating a user-defined type. Depending on the type of data (raw or processed data), the user might decide for either of the representations. Usually the most common format of processed laser scan data consist of seven parameters: *x,y,z*-coordinates, intensity and colour represented by *r,g,b*-values. The advantage of *point* data types is the possibility to manage all these attributes for each individual point. The major disadvantage refers data storage and indexing, which are very expensive (one record per point). The *multipoint* data type can be efficiently indexed, but the points lose their identification, which might be important for modelling purposes (Meijers et al 2005). Furthermore, the number of points in one *multipoint* has to be carefully considered for an acceptable performance (Hoefsloot, 2006). Depending on the point distribution and size of the point cloud, the operations can become time consuming and thus difficult to handle.

3.2 New 3D spatial data types

As shown above, 3D real-world feature can be stored and indexed in DBMS and retrieved for visualisation and editing in front-end application but they can be analysed only as 2D features. A true 3D geometry data type (polyhedron and/or tetrahedron) and corresponding 3D spatial operations (validation, volume, length, intersection, etc.) are missing in all DBMS. Furthermore, the simple 3D volumetric data type would be still insufficient for handling many shapes (cone, sphere) available in AEC/CAD applications. The sections below will briefly present two implementations of new 3D data types, i.e. polyhedron and NURBS surface.

3D polyhedron

A 3D spatial data type is the first most important development to be made by DBMS. A simple 3D object can be represented basically in three different ways as a polyhedron (consisting of arbitrary number of planar polygons which have arbitrary number of points), triangulated polyhedron (consisting of an arbitrary number of triangles) or tetrahedron (composed of four triangles). All the three representations have advantages and disadvantages (Zlatanova et al 2004). Tetrahedron is the simplest 3D object consisting of a finite number of points and

triangles. While advantageous for computations and consistency check (Peninga, 2005), tetrahedrons are less appropriate for modelling of man-made objects such as buildings, bridges, tunnels, etc., because the interior would require a subdivision into tetrahedrons (which should be omitted for visualisation). However, tetrahedrons are widely used in modelling geological formations (Breunig and Zlatanova, 2006). The polyhedron is the most appropriate representation for man-made objects, but its validation is quite complex (Arens et al, 2005). Triangulated polyhedron is compromise between the two ensuring at least the simplicity of the polygons.

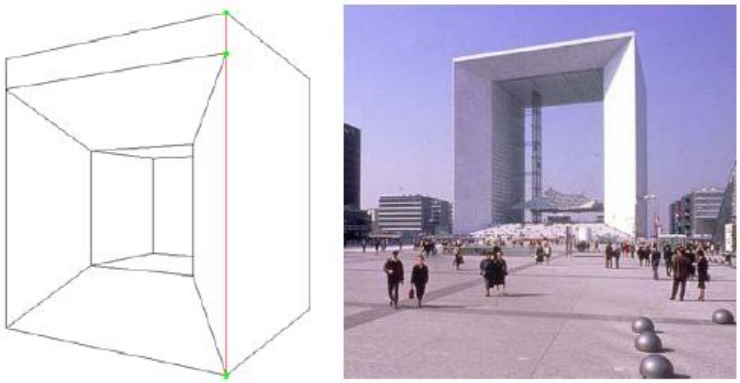


Fig. 12. 3D polyhedron (Arens 2003)

Arens 2003, and Arens et al 2005, have selected a polyhedron for implementation, since it is the most complex data type requiring strict validation rules. A polyhedron is defined as a bounded subset of space, enclosed by a finite set of planar polygons (not self-intersecting) such that every edge of a polygon is shared by exactly one other polygon. The polyhedron bounds a single volume, i.e. from every point on the boundary, every other point on the boundary can be reached via the interior. The polyhedron has clearly defined inside/outside space, i.e. it is orientable. The polyhedron defined in this way can have also cavities (Fig. 12).The polyhedron data type is implemented in Oracle Spatial object-oriented data model, but the formalism is generic. To avoid duplications of point coordinates (As mentioned above), the description has two sections: 1) a list of all the point coordinates and 2) a sequence of polygons, each composed of a list with indices to the point coordinates of the first section. The validity of the new data type is controlled by a specially designed function, which checks the definition rules. Several tests were performed on the new data type and the results were positive.
In very short terms, a tetrahedron data type will be prototyped following the formalism and implementation considerations as described in Peninga et al 2006.

3D freeform curves and surfaces
Freeform curves and surfaces such as Bezier, B-spline and NURBS, are becoming progressively important for modelling of buildings, towers, tunnels, etc. Very often these models need to be integrated with 3D GIS for investigations and adjustment of the construction (e.g. for wind resistance). One option for such an integrated environment can be DBMS. Therefore we have initiated a research aiming at developing data types that can be used by AEC applications. Among the large amount of mathematical representations of 3D space (used in CAD and AEC worlds), NURBS is the first candidate to be considered. Some of the most important NURBS characteristics are (Piegl and Tiller, 1997):

- NURBS offer a common mathematical form for both, standard analytical shapes (e.g. cone, sphere) and free form shapes,
- The shapes described by NURBS can be evaluated reasonably fast by numerically stable and accurate algorithms,
- Important characteristic for modelling real-world objects is that they are invariant under affine as well as perspective transformations.

The only drawback of NURBS is the extra storage needed to define traditional shapes (e.g. circles). Using NURBS data types, a circle can be represented in different ways but the complexity is much higher compared to its mathematical definition (i.e. radius and centre point).

The definition a NURBS curve consists of several quite complex equations, which can be found elsewhere in the literature. Based on these equations, the parameters to be included in the data type are specified as: control points, their weights, knots sequence and a degree value. These parameters double in case of surface (Pu, 2005). A NURBS curve requires fulfilment of a number of conditions such as the degree > 1, the number of control points >3, Degree = Number of knots - Number of control points – 1, he number of weight values is equal to the number of control points, ach weight value > 0, knot vector is non-decreasing and has more than 1 knot.

The new data type has been prototyped for Oracle Spatial, but outside the Oracle Spatial SDO_GEOMETRY model, which means it can be readily used for any spatial DBMS (PostGIS, MySQL, Informix, etc.). Besides the validation function, few simple spatial functions (rotation, translation, scale and distance) were developed. Since the data type is much more complex compared to the 3D polyhedron data type, a special care was taken for the visualisation in AEC software, i.e. an engine was developed to map the NURBS data type to the internal representation of Microstation and AutoCAD (Fig. 13). Two NURBS models were tested with the developed data type for retrieval, editing and posting.

Fig. 13. NURBS building retrieved from DBMS

Tests have convincingly demonstrated that appropriate data types for efficient management of freeform curves and surfaces can be created at DBMS level. The design is compliant with OGC Abstract Specifications (see Section 3.3). But the spatial functionality provided for NURBS data type has to be very carefully considered. Mathematics behind many NURBS operations might appear to complex for implementation at a DBMS level.

3.3 Standardization initiatives: Open Geospatial Consortium

With respect to spatial data management and interoperability, several standardization initiatives have been set off, e.g. ISO, IAI, Web3D, etc. The Open Geospatial Consortium will be mentioned here, because its activities has largely motivated the spatial developments in RDBMS.

The mission of the OGC, founded in 1994, is to enable interoperability of geo-services. OGC produces Abstract Specifications and Implementation Specifications (OGC, 2001). The aim of Abstract Specifications is to create and document a conceptual model sufficient to create the Implementation Specifications. The Abstract Specifications are subdivided into Topics, each of them related to different aspects of geo-spatial services. Topic 1 (identical to ISO 19107) is the most important one discussing the spatial schema for representing features. It should be noticed that the Abstract Specifications discuss a wide range primitives (also those used in CAD domain) such as cone, sphere, triangulated surfaces and freeform shapes such as B-splines and NURBS. Applying this framework it should be no real problem representing real world in 3D, using even different representations.

However, the Abstract Specifications provide only the general conceptual semantics. The way these can be implemented at different platforms (based on CORBA, OLE/COM and SQL) is described in three different Implementation Specifications. The Implementation specifications of interest for DBMS are the Simple Feature Specifications for SQL. Those provide guidance for implementing data types and spatial functions in DBMS. These specifications have greatly contributed to standardizing the implementing spatial functionality in DBMS. However, they are only 2D, i.e. limited to simple primitives such as points, lines and polygons (surfaces) and cannot meet the three-dimensional demands.

OGC has already started numerous new initiatives to meet the challenges. Currently 36 projects (working groups) are discussing various aspects of data integration and exchange only in the Specification Program and almost all of them attempt to handle the third dimension. The most relevant is the work of CAD-GIS working group (Case, 2005). The mission of this group is spanning a bridge between CAD, AEC systems and GIS by

finding opportunities to improve interoperability of geospatial data and services across these domains. Incompatibilities at various levels (semantic, geometry, topology) contribute to the complexity of the problem (Zlatanova and Prosperi, 2006).

To suggest an appropriate schema for exchange of 3D spatial data, this group will consider several on-going developments, i.e. LandXML (for land survey and construction initiated in 1999 by Autodesk and EAS-E members), LandGML, CityGML (GML for city models), aecXML (for AEC including information about projects, documents, materials, parts, organizations, professionals, etc.), TransXML (a project aiming at XML schemas for exchange of transportation data), IFC (the Industry Foundation Classes used to define architectural and construction-related CAD graphic data as 3D real-world objects), OpenFlight (an industry standard real-time 3D scene description format), 3D ShapeFile (ESRI), X3D, etc. There is a firm believe the work of this group will contribute to extension of the Implementation Specifications toward real 3D data management.

4 3D SPATIAL OPERATIONS

GIS has become a sophisticated system for handling spatial and thematic information of real world spatial objects. DBMS are evolving to a core component of GIS architecture used to maintain geographic data (Breuning and Zlatanova, 2006, Zlatanova and Stoter, 2006). Therefore we believe much GIS functionality can and have to be implemented within DBMS environment. The DBMS then becomes an important medium for spatial data maintenance, spatial operations and integration purposes (e.g. data access from different front-ends). The two major aspects of DBMS functionality that define a Geo-DBMS are then spatial data and analysis. Many researches have been working for providing support to 2D spatial data types (with 3D coordinates) and operations in mainstream DBMS. In the last year almost all DBMS vendors implemented the geometry models as defined by Open Geospatial Consortium (OGC). For example, Informix (2006) supports three basic spatial data types: point, line and polygon; Ingres (2006) supports one more data type: circle, beside the three basic types; Oracle Spatial (2006) not only able to handle points, lines, polygons and circles, but also gives further support to arc strings and compound polygons. These DBMSs manage spatial objects, together with their 3D coordinates, i.e. 2D objects (2D polygon + z-coordinate) are embedded in 3D space (Zlatanova 2006). Beside spatial data types, the spatial DBMS also implement operators and functions on the spatial data types, and thus geometric queries and operations are possible at DBMS level. However, the currently supported spatial data types are rather limited, mostly to 2D space. Although the points in most spatial DBMS can be 3D, the functions on spatial data types are actually still based on 2D, the z-values are hardly considered. There are also a number of functions on these spatial data types, which can do operations like return geometric information, do basic geometric transformations, maintain geometry validity, etc. The lack of "true" 3D data type (i.e. 3D tetrahedron, 3D polyhedron) results in the unavailability of 3D operations in DBMS.
The first attempt of 3D spatial data type and operations in a spatial DBMS has been investigated quite successfully by Arens (2003). Here, the 3D polyhedrons could be stored and manipulated in Oracle Spatial. The basic idea of Arens was that a 3D polyhedron could be defined as a bounded subset of 3D space enclosed by a finite set of flat polygons, such that every edge of a polygon is shared by exactly one other polygon. Here, the polygons are in 3D space because they are represented by vertices, which can be 3D points in a spatial DBMS. This research has been studied and some concepts are taken over in the new 3D data type implemented in Oracle Spatial 11.

Yet, another attempt to define 3D object is reported by Penninga, 2005. The 3D object, i.e. tetrahedron is used for representing volumetric shapes. The tetrahedron is the simplest possible geometry in 3D domain. The conceptual design is intended for implementation for both geometry and topology model (Penninga et al, 2006). Some other related research is reported by Pu (2005), who has created complex geometry types, i.e. freeform curves and surfaces. Although freeform shapes can be simulated by tiny line segments/triangles/polygons, it is quite unrealistic and inefficient to store all these line segments/triangles/polygons into a DBMS especially when shapes are rather huge or complex. Therefore, Pu and Zlatanova (2005) stored freeform shapes directly in DBMS. Corresponding spatial operators and functions that manage these shapes are also tested.

Another research that involves 3D city model had been discussed by Kolbe *et al.* (2006), namely CityGML (2006). CityGML was developed by developed since 2002 by the members of the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure North-Rhine Westphalia (GDI NRW) in Germany. It is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language 3 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211. CityGML supports 5 kinds of Levels of Detail (LOD), i.e. LOD0 denotes 2.5D Digital Terrain Model, LOD1 denotes the blocks model

comprising prismatic buildings with flat roofs, LOD2 has differentiated roof structures and thematically differentiated surfaces, LOD3 denotes architectural models with detailed wall and roof structures, balconies, bays and projections. High-resolution textures can be mapped onto these structures. In addition, detailed vegetation and transportation objects are components of a LOD3 model. LOD4 completes a LOD3 model by adding interior structures for 3D objects. Spatial properties of CityGML features are represented by objects of GML3's geometry model. This model is based on the standard ISO 19107 'Spatial Schema' (Herring 2001), representing 3D geometry according to the well-known *Boundary Representation* (Foley *et al*. 1995). The spatial data type is not modeled by a volumetric geometry, but it must be virtually closed in order to compute their volume. They can be sealed using *ClosureSurfaces*. *Closure-Surfaces* are special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualizations. In CityGML, 3D data type is often derived from or have relations to objects in different DBMS. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set, or may be derived from an architectural model. The reference of a 3D data type to its corresponding object in an external data set is essential for such model.
With these efforts, the possibility of developing 3D data types in DBMS is obvious. Currently missing are the topological operations for 3D data type. Current DBMS only provides 2D operations, which the z-value is not considered. 3D topological operations are also rather limited in typical GIS software. In this paper, we focus on simple but complete strategy in creating new datatype, i.e. polyhedron. Furthermore, the 3D datatype is used to test the developed topological operations that based on Spatial 9i model for 3D GIS. The algorithm fully covers the third dimension and able to be applied in 3D situation. The 3D topological operations are tested for PostgreSQL and can theoretically become a part of the PostGIS.

4.1 3D topological operations for PostgreSQL

The geometrical modeling for topological operations is based on the 9-intersection model and extends to 3D. Typically, the results given by this operation are in Boolean type, i.e. either TRUE or FALSE. The related operations include Overlap, Meet, Disjoint, Inside, Covers, CoveredBy, Contain, and Equal (see Fig. 14).

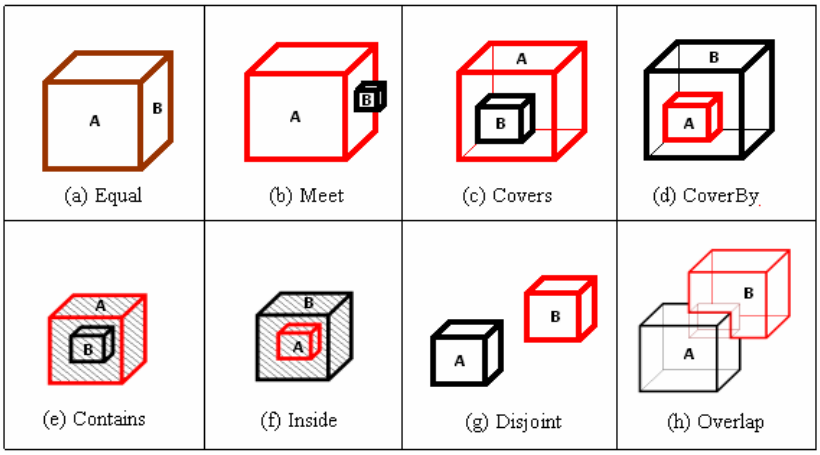


Fig. 14: Body and body relation (after Zlatanova, 2000)

For topological operation in geometrical model, coordinate triplet of vertex will be discussed. Similar to computational-geometry operation from previous section, the binary operation is divided into base and target object. However, the vertices from base object and polygons from target object will be discussed (see Fig. 15).

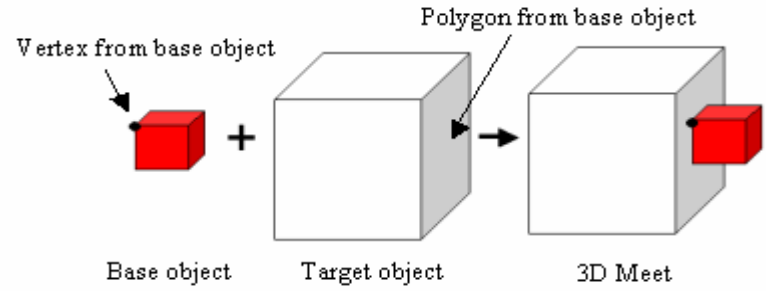
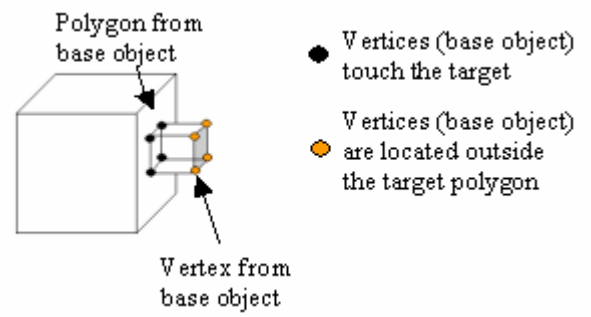


Fig. 15: Base and target object for 3D operation

This topological operation involves vertices (from base object) and polygon (from target object). Therefore, the relation between these 2 objects will be examined. The location of base vertices relative to target polygon will be either outside, touch, or inside. The implementation was discussed in Chen and Abdul-Rahman (2006). These relations will be used to determine how these 2 polyhedrons intersect each other as shown in Fig. 14. For example (see Fig. 16), vertices from base object are either touch the target polyhedron or located outside from target object.



3D topological operations	Inside	Outside	Touch
Meet	NO	YES	YES

Fig. 16: Vertices (base) are located and touch the target polygon

The following Table 1 denotes the complete relation between base and target object.

3D topological operations	Inside	Outside	Touch
Equal	X	X	✓
Meet	X	✓	✓
Covers	✓	X	✓
CoveredBy	✓	X	✓
Contains	✓	X	X
Inside	✓	X	X
Disjoint	X	✓	X
Overlap	✓	✓	✓

Table 1: Conditions for topological operations

4.2 The experiment and discussion

The 3D topological operations were tested within PostgreSQL environment. The existing spatial objects available in PostgreSQL are rather limited to 2D, but appear in three-dimensional space. The 3D primitive object is not available. Thus, 3D polyhedron will be discussed. The methodology for the complete implementation is given in Fig. 17.

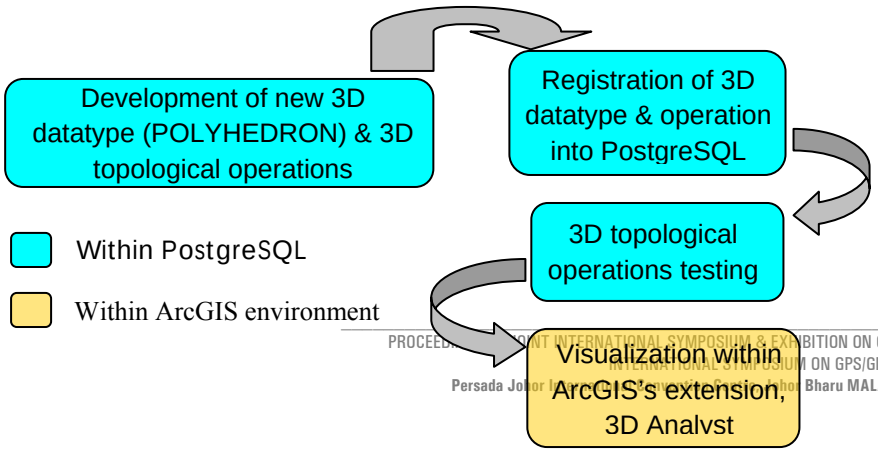


Fig. 17: The implementation of new 3D datatype & operations for DBMS

The following data structure denotes a sample of a polyhedron will be defined in PostgreSQL:

POLYHEDRON(PolygonInfo(6,24),SumVertexList(8),SumPolygonList(4,4,4,4,4,4),VertexList(100.0,100.0,100.0,400.0,100.0,100.0,400.0,400.0,100.0,100.0,400.0,100.0,100.0,100.0,400.0,400.0,100.0,400.0,400.0,400.0,400.0,100.0,400.0,400.0),PolygonList(1,2,6,5,2,3,7,6,3,4,8,7,4,1,5,8,5,6,7,8,1,4,3,2))

1). PolygonInfo(6,24) denotes 6 polygons and 24 IDs in PolygonList,
2). SumVertexList(8) denotes the total vertices,
3). SumPolygonList(4,4,4,4,4,4) denotes total vertices for each of polygon (total polygon is 6, referred to (1)),
4). VertexList() denotes the list of coordinate-values for all vertices (with no redundant), and
5). PolygonList() denotes the information about each polygon from sets of ID.

The graphical representation of the sample polyhedron stated above is given as follows (see Fig. 18):

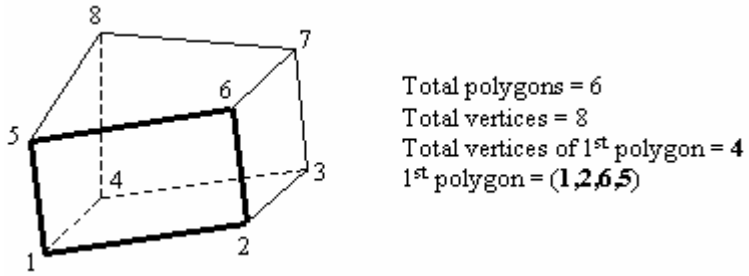


Fig. 18: Sample structure of a polyhedron

The following examples are implemented within PostgreSQL environment. 2 polyhedrons are inserted into a table, test, as follows:

INSERT INTO test(PID,POLYHEDRON) VALUES
(1,'POLYHEDRON(PolygonInfo(6,24),SumVertexList(8),SumPolygonList(4,4,4,4,4,4),VertexList(100.0,100.0,100.0,400.0,100.0,100.0,400.0,400.0,100.0,100.0,400.0,100.0,100.0,100.0,400.0,400.0,100.0,400.0,400.0,400.0,400.0,100.0,400.0,400.0),PolygonList(1,2,6,5,2,3,7,6,3,4,8,7,4,1,5,8,5,6,7,8,1,4,3,2))');

INSERT INTO test(PID,POLYHEDRON) VALUES
(2,'POLYHEDRON(PolygonInfo(6,24),SumVertexList(8),SumPolygonList(4,4,4,4,4,4),VertexList(300.0,300.0,300.0,600.0,300.0,300.0,600.0,600.0,300.0,300.0,600.0,300.0,300.0,300.0,600.0,600.0,300.0,600.0,600.0,600.0,600.0,300.0,600.0,600.0),PolygonList(1,2,6,5,2,3,7,6,3,4,8,7,4,1,5,8,5,6,7,8,1,4,3,2))');

The following SQL statement runs the 3D Overlap (see Figure 9):

SELECT GMOVERLAP3D(a.POLYHEDRON,b.POLYHEDRON) AS GM_OVERLAP3D FROM test a, test b where a.PID=1 and b.PID=2;

The result:

```
GM_OVERLAP3D
----------
(TRUE)
```

For visualization purposes, ArcGIS's extension, 3D Analyst will be used to verify the result. Although the integration between PostgreSQL and ArcGIS will not be discussed in this paper, the testing within the DBMS produces positive results as given in Fig. 19 and Fig. 20.

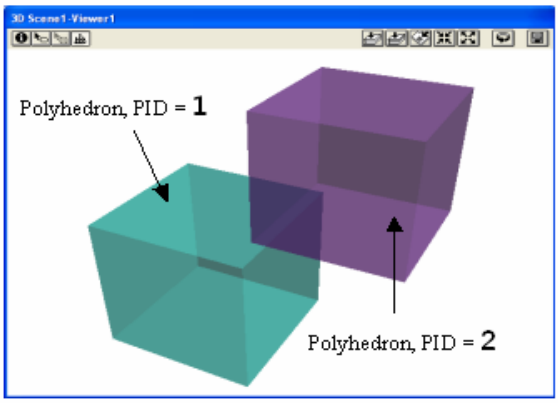


Fig. 19: 3D Overlap

The same implementation was done for the rest of 3D topological operations, i.e. 3D Meet, 3D Disjoint, 3D Inside/Contains, 3D Covers/CoveredBy, and 3D Equal (see Fig. 20a to 20e).

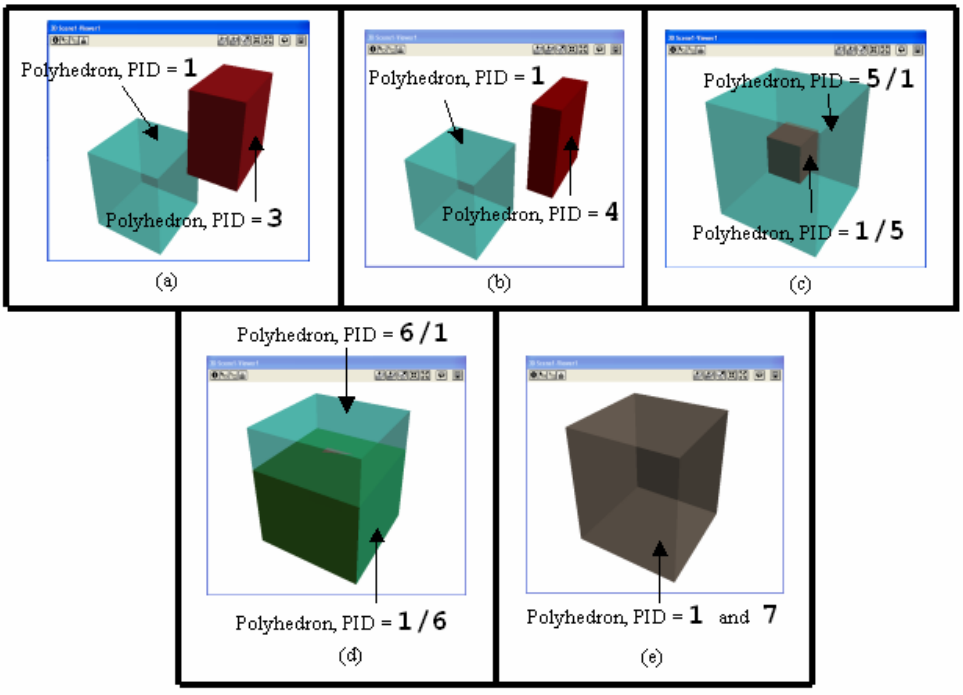


Fig. 20: (a) 3D Meet, (b) 3D Disjoint, (c) 3D Inside/Contains, (d) 3D Covers/CoveredBy, and (e) 3D Equal.

5 3D WEB SERVICES

With the growing usage of three-dimensional geographic systems (3D GIS) and related systems like Google Earth, the need of usable standards is obvious. While standards for data formats are well-discussed and also

developed, specific standards for services are in early development. Common data formats for 3D spatial data are Geographic Markup Language (GML), its profiles like CityGML (Kolbe et al. 2005) and proprietary types like Shape-files or Autodesk's DXF. Also, GeoTIFF for 2.5D data and Virtual Reality Modeling Language (VRML) and its successor X3D are often used. Regarding services, Web Feature Service (WFS) and its extension, the transactional WFS, are important because of their capability to distribute GML and their already high acceptance. Despite their disadvantage of providing only a projection of 3D Data, Web Terrain Service (WTS) and Web Map Service (WMS) are also used to derive products from 3D data.
In contrast to the WMS and WTS, the Web 3D Service (W3DS) (Quad and Kolbe 2005) as a portrayal service for three-dimensional geodata, delivers 3D scene graphs. These scene graphs will be rendered by the client and can interactively be explored by the user. The W3DS merges different types (layers) of 3D data in one scene graph.
One of the first implementations of the W3DS was integrated to the CityServer3D. The department Graphic Information Systems of Fraunhofer Institute for Computer Graphics has developed the CityServer3D system to implement new concepts and mechanisms for 3D GIS in current and future research projects. Due to its architecture the CityServer3D provides a platform to be easily extended by new modules. Based on a three-tier architecture, the system provides dynamically combinable data sources and can be accessed via several service interfaces. The geometric models are managed for example in the GeoBase21 database, processed by server components and delivered to clients. Due to the wide spectrum of supported interfaces, ranging from a custom Request/Response mechanism to standardized Web Services, these can be 3rd-party clients or the clients that were specifically developed for CityServer3D. With a facade technology that enables the integration of new service interfaces, the existing WMS facade was extended by the W3DS functionality. Now, CityServer3D provides the possibility to apply 3D city models in several application areas like tourism, navigation, risk management, decision support systems, or urban planning.

5.1 Visualization for Navigation Projects

New mobile and ubiquitous computing applications are rapidly becoming feasible, providing people an access to online resources at any time and everywhere. Location-based Computing (LBC) (Zadorozhny and Chrysanthis 2004) infrastructure comprises a distributed mobile computing environment where each mobile device is location-aware and wirelessly linked. As location awareness is the core of LBC, most Location-Based Services and applications rely on a global tracking technology and a large portion of geospatial information as provided by GIS. Typical Location-Based Services are city guide applications and mostly deal with three major problems:

Where am I now?
How do I get to my point of interest?
Where can I find something?

Nowadays, LBS can provide feasible answers to these questions - the problems are solved. New aspects come up in the context of mobile services like personalising, provisioning of attractive content and actual information but also the usability and graphic rendition in presenting of these services gain in importance. The user has - beside his navigation and orientation support - the demand for background information and more detailed visualisations to objects located on his route.

In a 3D-map the navigational value of a map increases due to the high visual correspondence between map objects and real world objects. This correspondence allows the user to recognize buildings easily (Coors et al. 2004), which leads to a more intuitive navigation inside the map in comparison to a 2D-map. Combined with an augmented reality user interface a 3D map with a dynamically created content fitting to his information needs will be the ultimate navigational aid for Location-Based Services.
Some milestone research projects and prototypes on the way to geospatial augmented reality and location aware systems are the ARCHEOGUIDE project (ARCHEOGUIDE 2002), the MARS project (Höllerer and Feiner 2004), the GEIST project (GEIST 2004), and the ULTRA project. To our knowledge the first 3D maps for cell phones, running on a Nokia communicator, were developed in the Tellmaris project (TELLMARIS 2003). The research on 3D-maps for mobile devices is continued in the m-Loma project (Nurminen 2006). For further details see (Blechschmied et al. 2005).

5.2 Architecture

The Web 3D Service

The Web 3D Service as a portrayal service for three-dimensional spatial data delivers graphic objects from a given spatial extent. Several formats can be chosen to deliver the scene. The data is delivered as 3D scene graph and therefore the client has to render the scene.
According to the query format which is related to the WMS and the WTS constraints can be given in a query. The two main requests are GetCapabilities to receive information about the service's capabilities and GetScene to receive a three-dimensional scene. Some important parameters to a GetScene query are:
SRS: Spatial reference system
POI: Point coordinates according to SRS
BBOX: A two-dimensional bounding box, defined by x_{min},y_{min},x_{max},y_{max}.
FORMAT: MIME type of the delivered scene
LAYERS: A list of layers
POC: The Point of Camera, or the viewing point describes the position of the camera.
STYLES: A list of so-called styled layer descriptors for each layer
There are also parameters like PITCH (angle of inclination), YAW (azimuth), ROLL (rotation around viewing vector) and some more which are optional or conditional. An example query to a W3DS would be:

http://clustera.igd.fraunhofer.de:7070/CityServer3D_R1.0.0/WMS?
request=GetScene&
exceptions=text/plain&
version=0.3.0&
srs=EPSG:16263&
bbox=62574.91,36549.58,63023.33,37102.43&
format=model/vrml&
poi=62800,0,-36800&
poc=62300,200,-37505

Additionally to the shown query, a layer list and style information can be provided. Here, layers are understood as thematically different types of 3D data. But the layer structure is depending on the data structure of the server which offers the W3DS services. Since there is no standard way of grouping – layers of the features in a data source can be regarded as grouping elements – the features in a data repository, the first step is to call the W3DS's GetCapabilities function. This will usually be triggered and also interpreted by a human user, to get the information which layers are available and which one should actually be retrieved in the GetScene requests that will follow. Common layers in a 3D city model often are buildings, street furniture or digital terrain models.

As mandatory output format of a GetScene request VRML was defined. For the result of the GetCapabilities operation, XML is used. The choice to define VRML as least common denominator was mainly driven by the aforementioned broad use of this format. Looking at classic internet-based scenarios with a desktop 3D-accelerated computer connected with good bandwidth to the Internet VRML has the advantages of being supported by a lot of applications, e.g. browser plug-ins.

Leaving these scenarios and looking towards mobile scenarios, e.g. pedestrian navigation with the use of cell phones, VRML produced files are too big to be transferred to the device in acceptable time. Also the VRML support on mobile devices is very heterogeneous.

The W3DS description also advises to integrate GeoVRML and/or X3D as formats to be supported. The advantage of both formats is that they overcome some gaps of VRML but both lack a broad usage inside the market. Due to the requirements of W3DS some other formats are being discussed:

- CityGML, with its inclusion of X3D elements, could be applicable to some scenarios where GML is used as an exchange format for visualization purposes. Here it has to be kept in mind that GML neither is meant as visualization format nor does it usually map a scene graph – the client will have to contain a powerful parsing engine for this task. Also, the W3DS does not aim at being a geodata exchange service, so for spatial data exchange services the WFS is the better choice.
- M3G (JSR-184), as a scenegraph format for mobile devices using J2ME. Having the growing support for Java ME in mind, this nowadays seems to be a good solution to achieve a high acceptance and also to reduce implementation effort due to Java's concept of “write one and run anywhere”.

5.3 W3DS Usage and Extensions

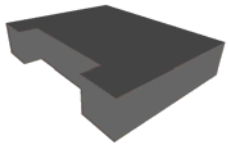
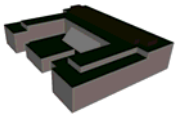
From the implementation and use of in the described mobile navigation scenarios and applications, it becomes clear that while the idea of complementing the WFS and WMS standards with a W3DS to handle visualization-centric requests of 3D geo-data is very promising, it still needs to be extended in several points to fulfill its role. The problems that occurred during work are:

VRML as Exchange Format
As mentioned, VRML on mobile clients, especially on cell phones, is not well supported and this is regarding consumer services an unacceptable restriction. Also, other 3D-supporting formats are not available on cell phones.
J2ME as an API offers the possibility to implement client software which is capable of querying, receiving and rendering 3D scenes. An approach could be to create a VRML viewer for cell phones or by integrating M3G support into mobile applications. J2ME can be combined with an API for M3G which makes an implementation easy.

Low Bandwidth in Combination with VRML
Another problem occurring with the use of VRML is generated by the low bandwidths which are available in today's mobile telecommunication networks. Often UMTS is claimed to be the solution for any performance problems of mobile applications. However, the bandwidth within an UMTS cell cannot be predicted. All devices which are logged in to a cell have to share the available bandwidth. This leads more or less to unpredictable bandwidth situation and limits – depending on the actual infrastructure development – the use especially where it might be required the most. For example, in cities where LBS and routing would be required, a high density of UMTS users will be logged in. This is a problem with which navigation systems using 3D city models will have to come along.

There is a clear need to reduce the network traffic as much as possible. Here, 3D scenes formatted as VRML are growing too big. By using M3G, the file size can be reduced in two steps. The following table is comparing the two formats:

Table 2: Comparison of VRML and M3G file sizes. For a detailed description of the definition of Level of Details for urban models see (CityServer3D 2005) and (Kolbe et. al. 2005)

	File content	Complexity	VRML	M3G	M3G compressed
	Fraunhofer IGD LOD 1	20 triangles	1,24KB	267B	403B
	Fraunhofer IGD LOD2	120 triangles	4,68KB	1,34KB	1,34KB
	Part of the city of Darmstadt, Germany	15000 triangles	1,06MB	59,9KB	3,93KB

Reducing the Data Transfer Volume by Compression
In a first step towards achieving a high compression, we currently use the ZIP encoding, as it is already supported by J2ME. However, by using a compression that uses the specific properties of 3D models, especially the topological information of meshes, much higher rates could be achieved. An example for such a

compression is the Delphi algorithm (Coors and Rossignac 2004), that uses a prediction scheme to reduce connectivity information down to less that 2 bits per triangle. Adding support for such a compression scheme to a standard like M3G as an optional feature would allow a very network-efficient transfer of even complex geometries, at the cost of slightly higher computing time on the side of the server and the client. The client could then tell the server which compression schemes it can decode and which one it would prefer, and the server will answer accordingly.

Extension of Query Constraints

The W3DS allows to define constraints via two parameters. The minimum bounding box is used to define the spatial selection whereas layers can be used to define a thematic selection. The second possibility strongly depends on the layer structure which a service owns. If it does not offer an application-fitting layer structuring, it cannot be used. This could lead to the situation that W3DSs would only be set up for special application scenarios and therefore would be too restricted. Within the market of geographic information systems, approximately 2/3 of the effort is spent in data acquisition and conversion. Due to this it should be possible that data and service providers could set up services which can be independently used from application scenarios.

Here, more mighty query manipulations can be thought of. In one of our scenarios, an extension to request objects via an identifier instead of the BBOX will be used. The identifier can be a single identifier, e.g. if just one building is requested, or a list of identifiers, e.g. a set of buildings is requested. These identifiers can be of different types. The identifiers used by the system, e.g. the land register identifier key, are appropriate for technical users while for end consumers a more user-friendly access has to be found. Here, the integration of names, e.g. “Darmstadt”, as identifiers has to be favoured whereas it has to be taken into account that the service has to process these identifying names with the help of thesauri or/and gazetteers.

Extensions of Spatial Query Constraints

Navigation always includes a movement and by using a cell phone on-line navigation system in combination with an W3DS supporting the BBOX as spatial constraint the whole scene has to be downloaded at the beginning of the trip. Since this leads to big scenes, the data transfer will take a long time, the client device will have to handle big scenes and a lot of unused data will be transferred. Finally, often it is not known at the beginning where the trip is going to end so that it is not possible to get the entire scene at the start.
In geographic information processing, spatial objects are also used to define spatial constraints in queries, e.g. a two-dimensional polygon can be used to query a result set out of a three-dimensional city model. Since the CityServer3D internally works with spatial constraint objects for queries, the W3DS was easy to adapt to these mechanisms.
For navigation systems on mobile devices, this approach enables the client to intelligently manage the data transfer. So, if the user moves in reality and wants to have a location-based visualization only missing features are going to be requested and transferred to the server. Therefore, the client has to be capable of extending dynamically the displayed scene.

Extension of LOD Support within Query Formulation

Modern systems which manage three-dimensional city models also support the management of LODs of features. In this usage LOD is used as term for persistent geometries with different quality levels. In [CityServer 2005] and [Gröger et. al., 2004] definitions can be found. To access different LODs the W3DS does not support a dedicated query parameter. Without an extension of the service, two solution approaches can be regarded. A first possibility is to use the layer or style query parameter to transfer information about the LOD which is wanted by the client. This approach is justified by thinking of the LODs as different styles to display a layer. However, the mentioned LODs usually have more than a pure visual purpose, and thus, using the layer descriptors this way is a way of mixing up various application domains and has to be considered a dirty solution.
A second way might be that the service decides by itself which LOD is the right one to choose and to deliver. By using this approach some interesting questions concerning the assessment in spatial and semantic aspects of feature in a city model come up. But with the heterogeneous hardware capabilities of mobile devices a W3DS service cannot determine the correct LOD since the service does not have any information about the bandwidth and rendering power of the requesting device.
By extending the W3DS by the parameter LOD, we give the power to the device to decide which model is requested. Here we also cannot get a exact prediction of what will be delivered by the service and what can be processed, but on the client we at least approximately know what can be rendered. The LOD parameter has to be quite flexible:
Assignment to a layer: All features of a layer are delivered in the defined LOD.
Assignment to a query: All result features of a query are delivered in the requested LOD.

Assignment according to the available resources: In case of bandwidth of processing restrictions, a definition of a limit for LODs is useful. So, if no other LODs are given to the query, it can be assured that the service does not deliver any LODs above the set limit.

5.4 Applications

For applications we use the technology CityServer3D which is extended according to the requirements of the specific application. The CityServer3D consists of several components which form together a system to visualize, manipulate, and handle 3D spatial data like city and digital elevation models. The system is serving as a technical platform to realize solutions for different application scenarios like tourism, navigation, risk management, decision support systems or urban planning. The following figure shows a non-photo realistic visualization of the area around the Fraunhofer Institute for Computer Graphics.

Fig. 21 Sample Visualization of a city model using sketch rendering

The CityServer3D uses facade mechanisms within its interface layer which also provides the OGC services. W3DS uses this facade of the interface layer and is able to use not only the VRML converter from the converter layer of the server. Also other appropriate formats can be integrated into the W3DS interface as long as they are implemented as components in the converter layer. The W3DS is already used in projects of the Fraunhofer Institute for Computer Graphics. Regarding 3D Maps on mobile devices two supplementary solutions which use the W3DS were developed.

Route Extension

Planning, analysing and visualization of route objects for navigation of cyclists and hikers are processed by the route extension. To support mobility for targeted user groups the CityServer3D was extended by three components which enable users to store routes and to extend them to travel guides. These extensions correspond to three different user groups (roles). The data administrator has to maintain the stored models which are mainly routes, two dimensional vector-based mapping data and digital elevation models. So-called authors use the system to generate, import and store routes which are part of travel guides. End users are browsing in travel guides and are provided with the functionality to buy interesting guides.

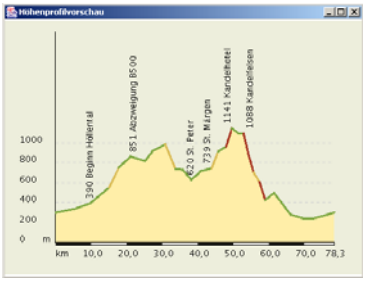


Fig. 22 Dynamically generated height profile

To generate new routes, authors can define a route manually or import a route by using GPS devices. The new or manipulated route is stored in the CityServer3D system and is exported as XML, as overview map, and as an altitude profile image. Fig. 22 shows a sample profile and the usage of colours to show the gradients of route segments. Due to the user's requirements a highly configurable but simple and well-known visualization -the height profile- was used.

Within the definition of a route an author also provides information about important waypoints and POIs. While waypoints are part of a route object and belong to the three-dimensional line object, POIs are not part of the route geometry but are assigned to a route since they are of important character. POIs can be of several types which are predefined by an application-specific code list. They mark an interesting point according to the navigation, e.g. a high tower, or to the route theme, e.g. an archaeological site.

3D Visualization
To visualize POIs and landmarks in 3D maps on mobile devices, M3G in combination with a self-developed Java-based mobile viewer is used. POIs and landmarks can be used to answer the question: How do I get from my present position to the desired destination? Up to now this typical question made by visitors of an unknown place has been answered by using two-dimensional maps. These maps are indeed able to give a rough overview of a large area since cartographic functions are well researched and developed for two-dimensional maps. For orientation in a smaller area, however, two-dimensional maps are often insufficient as they do not take landmarks into account. Thus, informational needs like the current position and the direction to go are not directly deducible from a map leading to problems in finding the way to a desired destination.

By supporting the visualization of different LODs a situation-dependent and query-adaptive visualisation of city models can be generated. To visualize a route from a specific point – like the current position of the user – to a destination an optical accentuation and detailed presentation of objects being of high importance to the route is sufficient, surrounding buildings can be shown in a lower level of detail. This enables the transmission of large-area maps in short time, with low costs for the user and the presentation of maps on a cell phone in spite of its limited resources. The following figures are showing two approaches to visualize 3D features on cell phones.

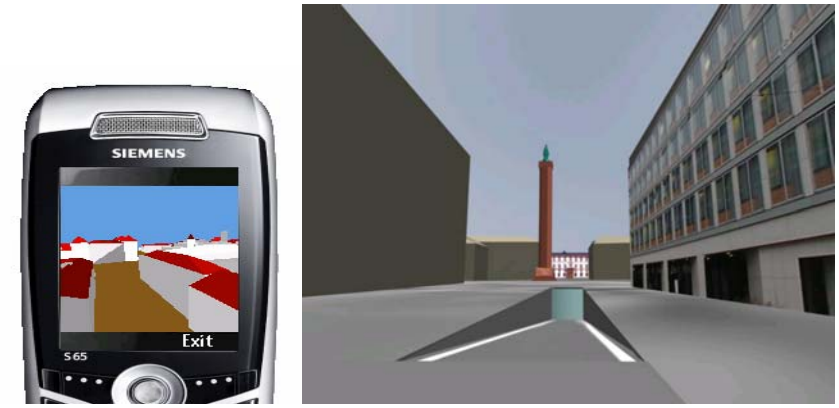


Fig. 23 (left) M3G-based visualization of Darmstadt on a Siemens S65 cell phone (right) Streamed movie visualizing landmarks and building of less interest

Fig. 23 (left) shows the mobile viewer on a Java-enabled cell phone. The navigation with the loaded model is done by shortcuts, e.g. to access a bird's eye view, buttons, and via the joystick if it is available. Within this version of the mobile viewer W3DS is used to query and receive the required data. Fig. 23 (right) shows a non-Java version which presumes the capability of receiving and rendering videos on the client device. Here, the Sony P900 was used to visualize the route which was prepared and rendered on the server. With this approach Java is not needed but an interaction with the model is not possible for the user anymore.

6 CONCLUDING REMARKS

We have presented and discussed some recent research works done within our group, that is covers 3D modelling for 3D objects, 3D spatial database (geo-DBMS), 3D spatial operations, and 3D Web services.

First, we have outlined a procedure for the direct extraction of building exteriors from LIDAR data without any additional data sources and pre-defined building models. More complicated buildings have been built. With the help of the research of Charlesworth et al. (1975) may be able to model the roof like the Wales Millennium Centre which is an arch shape roof.

Second, in the last five years DBMS made a large step toward maintenance of geometries as GIS used to manage them. The support of 2D objects with 3D coordinates is adopted by all mainstream DBMS. The offered

functions and operations are predominantly in the 2D domain. The DBMS spatial schemas have to be extended to fully represent the third dimension (first with simple volumetric object and later with more complex 3D data types). Concepts for 3D objects and prototype implementations are already reported, DBMS have to make the next step and natively support them. 3D operations and functions have to be developed not only for the volumetric object but also for all other objects embedded in 3D space. 3D functionality is next to be considered. It should be remembered that Spatial DBMS is a place for storage and management, and less intended for extensive analyses. The 3D functionality should not be completely taken away from front-end applications such as GIS and CAD/AEC. 3D Spatial DBMS should provide the basic (generic) 3D functions, such as computing volumes and finding neighbours. Complex analyses have to be attributed to the applications.
Some existing data types are clearly not sufficient for the purpose of some applications. A very typical example is the *multipoint.* It was definitely not designed for large amounts of points as from laser scanning. DBMS fail to handle efficiently such amounts of data until now. Such points need a special treatment. On the one hand, with the progress of data collection techniques, the amounts of points will only increase. Many laser-scanning companies are increasingly getting concerned about the management of such data. On the other hand, the advances in 3D modelling would require more intelligent management of both row and processed data. Clearly a new spatial data type with internal structure and index has to be developed. *Triangle* (or TIN) data type is also quite demanded. It is likely that TIN will continue to be widely used for all kinds of complex surface representations in GIS. Most of the terrain representations presently maintained in GIS as well as many CAD designs (meshes) are TIN representations. TINs can be stored in DBMS using the polygon data type. This data type is generally assumed for multiple vertices and thus over-attributed. Thus a simpler adapted data type is required. Maintenance of multiple representations to be used as Level-Of-Details is another critical aspect of large three-dimensional models. Still the management of multiple representations is far from formalized. A very promising initiative is CityGML, which concepts can further be incorporated in the Spatial Schema and later incorporated in Implementation Specifications. Management of texture and mechanism for texture mapping and texture draping is critical for management of realistic 3D City models. 3D objects usually need more attributes for visualisation compared to 2D objects. Moreover, very often 3D objects are textured with images from real world. As AEC and GIS applications come together the question of linking textures to geometries will appeal. Textures can be understood as 'presentation' attributes of 3D objects an also decoded in the data types. As mentioned above, the update of the OGC Implementations Specifications is very critical. Standardized vision on 3D objects and functions on them will stimulate DBMS toward 3D implementations. The first extension should be the volumetric object. To be able to provide a stronger management of objects from AEC and GIS, the Implementation Specifications have to be extended with other more complex 3D objects such as parametric shapes, freeform curves and surfaces. The 3D spatial functionality regarding those complex 3D spatial data types has to be further defined. Note, the Abstract Specifications provide only the Spatial Schema and do not discuss intended functionality. 3D Spatial DBMS has to offer an appropriate 3D user interface. To date, 3D user interfaces has not yet been exhaustively examined. Future 3D query interfaces should support the formulation of complex SQL-like mixed spatial and non-spatial database queries as well as 3D graphical input supporting the intuitive graphical formulation of 3D queries. Only few CAD/AEC interfaces (e.g. Bentley, Autodesk with respect to Oracle Spatial) are flexible enough to support the manipulation and database update of 3D objects. Further extended 3D solutions are expected primarily from the coupling of AEC software with Spatial DBMS. AEC applications offer a rich set of 3D modelling and visualisations tools that can be further extended toward specifying 3D spatial (SQL) queries.

Third, we also have implemented an approach for 3D topological operations of geometrical model in Geo-DBMS. The results show that a more complete modeling for 3D GIS analysis is possible. Our experiment was tested within PostgreSQL computing environment and has provided a promising outcome with respect to the developed algorithm. The test data set consists currently of very simple objects as it can be seen from the example, but some more experiments with real data sets are planned and will be completed soon. Besides, the same implementation could also be used in other DBMS, e.g. Oracle Spatial, similar to the work by Pu (2005) that had been done using free-from objects. We believe this research effort towards realizing a fully 3D spatial analysis tools within Geo DBMS environment would be beneficial to 3D GIS research community. This is because major GIS task involves DBMS (except 3D visualization), i.e. dataset handling, spatial operations, etc. It is our aim to move further in addressing this issue of spatial data modeling and geometrical modeling for 3D GIS. The research could also be extended for the integration between 3D Geo-DBMS and 3D visualization. Realizing that graphical display is one of the important components for GIS, 3D visualization plays an important part for displaying dataset from DBMS.

Fourth, we also have provided an overview on the current status and proposed some extensions which were motivated by our work for a further development of the W3DS. Making 3D visualization for spatial data available via web services seems to us an important function to build up networked and connected applications.

As a well-known example, Google Earth springs to mind. For a wide usage of such services, e.g. in navigation systems, there is nowadays a lack of standards. Here we propose to use W3DS which is still in development but shows promising possibilities. Especially the relationship to existing standards eases the set up and usage of the W3DS. Besides the described extensions others are possible. In navigation scenarios, especially if a route is planned, an enhanced scene could be useful. By extending the scene with a route object and a related animation, the user could receive a animated preview of his trip. Especially hikers and bicyclists could improve their travel preparations. In this case, an animation all over the route would most probably not be helpful since the information depth would not be well distributed. So, there should be the possibility to integrate not one but more animation object, which can be assigned to a route. This approach could be used to provide the user with animation of hotspots, e.g. crossings, points of interest.

Finally, we hope our findings on several aspects of 3D GIS development could be improved for better understanding of the real world situations as they exist in 3D world and eventually lead to full blown 3D GIS system in the near future.

Acknowledgements

We would like to thank to our PhD students and co-researchers such as Chen Tet Khuan at the Department of Geoinformatics, UTM, Malaysia; Shi Pu at TU Delft, The Netherlands, Jorg Haist and Thorsten Reitz at Fraunhofer Institute, Darmstadt, Germany; and Rebecca Tse at Hong Kong Polytechnic University, Hong Kong. Their contributions are highly valuable. Thanks also go to research grant sponsor, i.e. the Malaysian Ministry of Science, Technology and Innovation (MOSTI).

REFERENCES

Ackermann, F. (1999) Airborne laser scanning - present status and future expectations. ISPRS Journal of Photogrammetry Remote Sensing, 54 (1):64–67, 1999.

ARCHEOGUIDE (2004) project website: http://archeoguide.intranet.gr/ (accessed 6 July 2006)

Arens, C, 2003, Maintaining Reality; Modelling 3D spatial objects in a Geo-DBMS using a 3D primitive, Master's Thesis TU Delft, 2003, 76 p.

Arens, C., J.E. Stoter, and P.J.M. van Oosterom, 2005, Modelling 3D spatial objects in a geo-DBMS using a 3D primitive, In: Computers & Geosciences, Volume 31, 2, pp. 165-177

Bittner, T., M. Dinnelly and S. Winter, 2006, Ontology and Semantic Interoperability, in: Zlatanova and Prosperi (Eds.) Large-scale 3D Data Integration: Challenges and Opportunities, Taylor&Francis, A CRC press book, pp. 139-160

Blechschmied H, Coors V, and Etz M (2005) Interaction and Visualization of 3D City Models for Location-Based Services. In: Zlatanova S and Properi D (2005) Large-Scale 3D Data Integration: Challenges and Opportunities, CRC Press, Taylor & Francis Group, New York.

Brenner, C. (1999) Interactive modelling tools for 3D building reconstruction. In D. Fritsch and R. Spiller, editors, Photogrammetric Week '99', pages 23–34, Wchmann Verlag, Heidelberg, 1999.

Breuning, M. and S. Zlatanova, 2005, 3D Geo-DBMS, Chapter 4 in S. Zlatanova&D. Prosperi (Eds.) Large-scale 3D Data Integration: Challenges and Opportunities, Taylor&Francis, A CRC press book, pp. 88-113

Case, T., 2005, CAD/GIS Working Group, Charter and Mission, available at http://portal.opengeospatial.org, 4 p.

Charlesworth, Charlesworth, Langenberg, and Ramsden (1975) Determining axes, axial places and sections of macroscopic folds using computer-based methods. Candian Journal Earth Science, 13:54–65, 1975.

CityServer3D: http://www.invisip.de/CityServer3D/ (accessed 6 July 2006)

Coors, V., Elting, C., Kray, C., Laakso, K. (2004) Presenting Route Instructions on Mobile Devices - From Textual Directions to 3D Visualization, In: Dykes, J., A.M. MacEachren & M. J. Kraak (eds). Exploring geovisualization. Amsterdam: Elseviers, 2005, pp. 529 - 550

Coors V, and Rossignac J (2004) Delphi: geometry-based connectivity prediction in triangle mesh compression In: The Visual Computer, International Journal of Computer Graphics, Vol. 20, Number 8-9, Springer Verlag, Berlin Heidelberg, November 2004, pp 507 – 520.

GEIST (2004) project website: http://www.igd.fhg.de/igd-a5/projects/geist.html (accessed 6 July 2006)

Haist J, and Coors V (2005): The W3DS-Interface of Cityserver3D. In: Kolbe, Gröger (Ed.); European Spatial Data Research (EuroSDR) u.a.: Next Generation 3D City Models, pp. 63-67

Höllerer T, and Feiner S (2004) Mobile Augmented Reality. In: H.A.Karimi and A. Hammad (Eds): Telegeoinformatics Location-Based Computing and Services, CRC Press, pp 221-260

Hoefsloot, M. 2006, Storing Point clouds in DBMS, MSC Thessis, available at: http://www.gdmc.nl/publications, 80p.

Hofmann, Maas, and Streilein (2003) Derivation of roof types by cluster analysis in parameter spaces of airborne laserscanner point clouds. In IAPRS International Archives of Photogrammetry and Remote Sensing and Spatial Information Sciences, volume 34, Part 3/ W13, pages 112–117, Dresden, Germany, 2003.

INSPIRE, 2004, http://www.ec-gis.org/inspire/

Kolbe T., Gröger G, and Plümer L (2005) CityGML – Interoperable Access to 3D City Models In: Oosterom P, Zlatanova S, and Fendel E (Eds.): Proceedings of the Int. Symposiumon Geo-information for Disaster Management, Springer Verlag

LoVEUS, 2004, Project website http://loveus.intranet.gr (accessed 24 January 2005)

Meijers, M., S. Zlatanova and N. Preifer, 2005, 3D geoinformation indoors: structuring for evacuation, , in: Proceedings of Next generation 3D city models, 21-22 June, Bonn, Germany, 6 p.

Nurminen A (2006) m-Loma – a mobile 3D city map. In Proceedings of Web3D 2006, New York: ACM Press, pp 7-18

OGC, 2001,OpenGIS Consortium, The OpenGIS Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), Version 5, edited by J.H. Herring, OpenGIS Project Document Number 01-101, Wayland, Mass., USA.

OGC, 2005, Open GIS Specifications, Simple Feature Specifications for SQL (SFS) http://www.opengeospatial.org/specs/?page=specs

Oosterom, van P., J. Stoter, W. Quak and S. Zlatanova, 2002, The balance between geometry and topology, in: D. Richardson and P.van Oosterom (Eds.), Advances in Spatial Data Handling, 10th International Symposium on Spatial Data Handling, , Springer-Verlag, Berlin, pp. 209-224

Oosterom, van P., J. Stoter, and E. Jansen, 2006. Bridging the worlds of CAD and GIS, in Zlatanova and Prosperi (Eds.) Large-scale 3D Data Integration: Challenges and Opportunities, CRCpress, pp. 9-36

Penninga, F., 2005, 3D Topographic Data Modelling: Why Rigidity Is Preferable to Pragmatism, in 'Spatial Information Theory', Cosit'05', Vol. 3693 of Lecture Notes on Computer Science, Springer, pp. 409–425.

Penninga, F., van Oosterom, P. & Kazar, B. M., 2006, A TEN-based DBMS approach for 3D Topographic Data Modelling, in 'Spatial Data Handling '06'.

Piegl, L. and Tiller W., 1997, The NURBS Book 2nd Edition, Springer-Verlag

Pu, S. 2005, Managing Freeform Curves and Surfaces in a Spatial DBMS, MSc Thesis, TU Delft, 2005, 77 p. available at http://www.gdmc.nl/publications

Reitz T (2005), Architecture of an interoperable 3D GIS under special considerance of visualization applications. Master Thesis Fachhochschule Furtwangen

Rottensteiner, F. and C. Briese (2003) Automatic generation of building models from LIDAR data and the integration of aerial images. In H.-G. Maas, G. Vosselman, and A. Streilein, editors, Proceedings of the ISPRS
working group III/3 workshop '3-D reconstruction from airborne laserscanner
and InSAR data', volume 34 Session IV, Dresden, Germany, 2003. Institute of Photogrammetry and Remote Sensing Dresden University of Technology.

Sohn, G., and I. Dowman (2003) Building extraction using lidar DEMS and IKONOS images. In H.-G. Maas, G. Vosselman, and A. Streilein, editors, Proceedings of the ISPRS working group III/3 workshop '3-D reconstruction from airborne laserscanner and InSAR data', volume 34 Session IV, Dresden, Germany, 2003. Institute of Photogrammetry and Remote Sensing Dresden University of Technology.

Sohn, G., and I. Dowman (2004) Extraction of buildings from high resolution satellite data and LIDAR. In ISPRS 20th Congress WGIII/4 Automated Object Extraction, Istanbul, Turkey, 2004.

Suveg, I., and G. Vosselman (2004) Reconstruction of 3D building models from aerial images and maps. ISPRS Journal of Photogrammetry & Remote Sensing, 58(3–4):202–224, 2004.

Stoter, J. and S. Zlatanova,2003, Visualisation and editing of 3D objects organised in a DBMS, Proceedings of the EuroSDR Com V. Workshop on Visualisation and Rendering, 22-24 January 2003, Enschede, The Netherlands, 16p

TELLMARIS (2003) project website: project website: http://www.igd.fhg.de/igd-a5/projects/geist.html (accessed 6 July 2006)

Tse, R. O. C. (2003) Semi-Automated Construction of fully three-dimensional terrain models. PhD thesis, The Hong Kong Polytechnic University, Hong Kong, 2003.

Tse, R., and C. Gold (2001) Terrain, dinosaurs and cadastres - options for three-dimension modelling. In C. Lemmen and P. van Oosterom, editors, Proceedings: International Workshop on "3D Cadastres", pages 243–257, Delft, The Netherlands, 2001.

Tse, R., and C. Gold (2002) TIN meets CAD - extending the TIN concept in GIS. In P.M.A. Sloot, C.J.K. Tan, J. Dongarra, and A.G. Hoekstra, editors, Computational Science - ICCS 2002, International Conference, Proceedings of Part III. Lecture Notes in Computer Science, volume 2331, pages 135–143, Amsterdam, the Netherlands, 2002. Springer-Verlag.

Tse, R. and C. Gold (2004) TIN meets CAD - extending the TIN concept in GIS. Future Generation Computer Systems (Geocomputation), 20(7):1171–1184, 2004.

Vijlbrief, T. and P.J.M. van Oosterom, 1992, GEO++: An extensible GIS, Proceedings 5th International Symposium on Spatial Data Handling, August, Charleston, South Carolina, USA.

Vosselman, G. and S. Dijkman (2001) 3D building model reconstruction from point clouds and ground plans. In International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, volume 34, part 3/W4, pages 37–43, Annapolis, MA, USA, 2001.

Vosselman, G., B.G.H. Gorte, Sithole, and T. Rabbani (2003) Recognising structure in laser scanner point clouds. In International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, volume 46, part 8/W2, pages 33–38, Freiburg, Germany, 2004.

Zadorozhny, V. and Chrysanthis, P., 2004, Location-Based Computing, In: H.A.Karimi and A. Hammad (Eds): Telegeoinformatics Location-Based Computing and Services, CRC Press, pp 145-170.

Zlatanova, S., D. Holweg and V. Coors, 2004, Geometrical and topological models for real-time GIS, in: Proceedings of UDMS 2004, 27-29 October, Chioggia, Italy, CDROM, 10 p.

Zlatanova and Prosperi (Eds.), 2006. Large-scale 3D Data Integration: Challenges and Opportunities, Taylor&Francis/CRC press book

Zlatanova, S. and J. Stoter, 2006, The role of DBMS in the new generation GIS architecture, Chapter 8 in S.Rana&J. Sharma (Eds.) Frontiers of Geographic Information Technology, Springer, pp. 155-180

Zlatanova, S., A.A. Rahman and M.Pilouk, 2002, Trends in 3D GIS development, in: Journal of Geospatial Engineering, Vol.4, No.2, pp. 1-10.