A DBMS-BASED 3D TOPOLOGY MODEL FOR LASER RADAR SIMULATION

C. Jun [a, *] G. Kim [a]

[a] Dept. of Geoinformatics, University of Seoul, Seoul, Korea - (cmjun, nani0809)@uos.ac.kr

Commission VII

KEY WORDS: Modelling, Simulation, Data Structures, Database, Laser scanning, Three-dimensional

ABSTRACT:

Developing LADAR(Laser radar) is viewed to be an important technology for next generation guided weapons in many countries. However, experiments using real guided weapons are not practical and we need computing environment that can simulate the 3D detections by LADAR. Such simulations require dealing with large sized data representing buildings and terrain over large area. They also need the information of 3D target objects, for example, material and echo rate of building walls. However, currently used 3D models are mostly focused on visualization maintained as file-based formats and do not contain such semantic information. In this study, as a solution to these problems, a method to use a spatial DBMS and a 3D model suitable for LADAR simulation is suggested. The 3D models found in previous studies are developed to serve different purposes, thus, it is not easy to choose one among them which is optimized for LADAR simulations. In this study, 4 representative 3D models are defined, each of which is tested for different performance scenarios. As a result, one model, "BODY-FACE" structure, is selected as being the most suitable model for the simulation. A process to build a spatial DBMS and to compute and visualize with the proposed model was illustrated using a test area.

1. INTRODUCTION

LADAR(Laser radar), the 3D detection technology is increasingly getting attention as being the next generation guided weapons. Experiments using real guided weapons for the development of the LADAR would require tremendous amount of money. Therefore, we need computing environment that can simulate the 3D detections by LADAR. Such simulations require dealing with large sized data representing buildings and terrain over large area. They also need the information of 3D target objects, for example, material and echo rate of building walls. However, currently used 3D models are mostly focused on visualization and do not contain such semantic information.

Modeling and visualizing terrains in 3D have been well known techniques now and most commercial GIS packages accommodate tools to represent terrain data types (i.e. TINs). However, modeling techniques for buildings in 3D have less been established and are still being studied theoretically without explicit implementations. Current techniques can be categorized into two—CAD and GIS. CAD systems, with diverse data types (e.g. cylinders, cones and freeform shapes), have been extensively used to model complex shapes in architecture or mechanics fields. On the other hand, GIS is mainly designed to represent geographical features and use less number of data primitives than CAD, which are points, lines and polygons. However, 3D representation provided by current commercial GIS packages is limited to 2.5D, which means that one location can have only one z value. Although both approaches have been used for years and suffice visualization purposes at certain application domains, still some issues remain to be resolved.

Neither CAD nor GIS for 3D objects supports topological structure. Topology is the key property implemented in 2D GIS such as adjacency and connectivity that enables diverse analyses. We need such property in 3D buildings in order to define semantic information in urban models for LADAR simulations. LADAR simulations frequently deal with large areas and, thus, we need to represent a larger number of buildings than a few. Most CAD systems use file-based formats, which are unfavorable for storing and visualizing many building objects due to the computational performances. In contrast, although GIS packages support both file and DBMS formats, they are mostly software-dependent and do not support 3D topology as of now.

A solution to these problems would be using DBMSs, which are widely accepted as reliable method for managing large amount of data. The purpose of this study is to compare DBMS-based 3D models and suggest suitable ones that satisfy spatial operations and visualization for LADAR simulations while minimizing computation time. We first categorized 3D topology data models found in literature including our own model. Then we carried out performance tests for each of the models to test differences in retrieval times for visualization and range queries. The PostgreSQL/PostGIS was used for the tests. We also visualized the queried geometries using VRML and OpenGL.

2. 3D MODELS

2.1 3D models for building details

Architecture may be the field that use 3D models most extensively. CAD-based models have been used widely for detailed 3D building modeling, and there is a growing interest in using IFC(Industry Foundation Classes) format for modeling and developing building information systems. Although these formats offer flexibility in modeling 3D objects with various data primitives, their file-based formats have limitations for

* Corresponding author.

dealing with many objects as in urban scales. On the other hand, CityGML which was adopted as a standard by OGC(Open Geospatial Consortium) is a 3D model that provides different levels of details ranging from region to interior spaces (Stadler *et. al*., 2007; Elllul *et. al*., 2008). CityGML is based on XML format for the storage of data and has capability of storing complex semantic information. However, as of writing this paper, it has not provided fully functional database implementation. One of the reasons is attributed to the fact that current commercial DBMSs do not fully support topological structure of 3D objects yet.

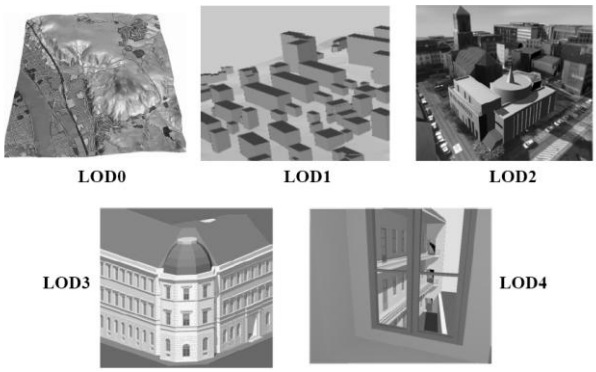

Figure 1. Five. LoDs in CityGML (Kolbe, 2008)

Theoretical 3D models have been studied actively for the last decade, and some researchers have connected their models with DBMS implementation. Stoter *et. al*. (2003) proposed a 3D data model and applied it to a cadastre problem to solve the property rights in 3D situations. She used a DBMS for storing her model and visualized in VRML. Different model types proposed by others are described in the next section. However, we choose to focus on the cases that use DBMSs rather than purely theoretical models for the purpose of this study.

2.2 3D Topology Models

The possibilities of using DBMS for 3D objects have been recently investigated by many researchers (Ellul, 2008; Stoter, 2002; Stoter *et. al*., 2003; Zlatanova, 2000). With some variations, they mostly suggested hierarchical relationships between 3D element classes, which are based on SOLID-FACE-EDGE-NODE in order to represent 3D topology. Faces are bounded portions of a solid surface, edges are used to define the face boundary, and nodes constitute the edges. However, as of now, we have not found any literature that shows experiments with large amounts of 3D objects. The major motivation of this paper is testing the applicability of DBMS approaches whether they can accommodate large amount of 3D data while supporting 3D topology. We first categorized 3D topology data models suggested in other literature (Type 1, 2 and 4) including our own model (Type 2) as follows:

- Type 1: SOLID – FACE – EDGE – NODE
- Type 2: SOLID – FACE – EDGE
- Type 3: SOLID – FACE – NODE
- Type 4: SOLID – FACE

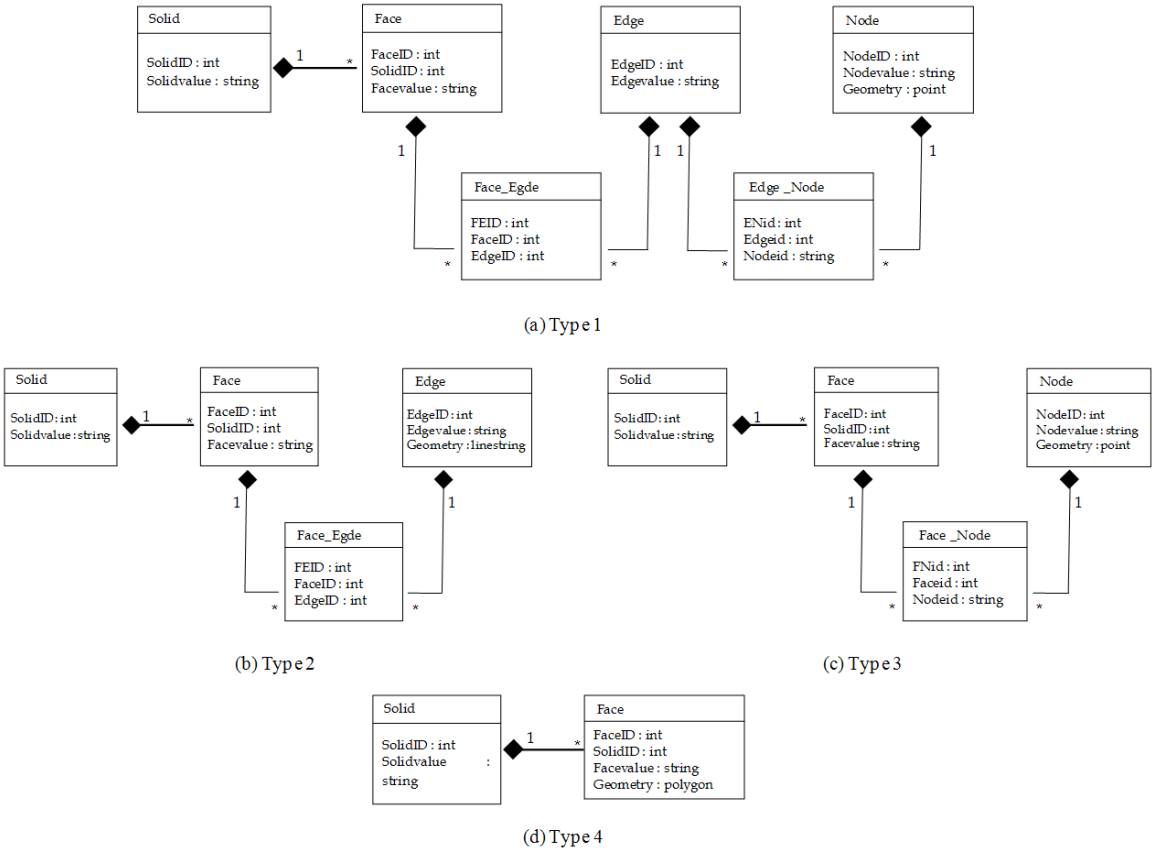

Figure 2. UML class diagrams describing storage of a polyhedron with four types.

In each model here, the last class contains geometries and the rest are non spatial relations. For example, in Type 1, the NODE class stores actual point data and each of SOLID, FACE and EDGE classes corresponds to a non-spatial table.

Figure 2 shows the UML class diagrams that describe storage of a polyhedron using the four types. Type 1 model provides full topology, which enables storing semantic data in each table (Figure 2-(a)). As we mentioned above, actual points are stored in NODE table, and spatial operation query is performed using this geometry. This model has an advantage for storing semantic attribute data in each table. However, many-to-many relationships such as node-edge and edge-face require additional joining of tables leading to inefficiency for computation due to excessive join queries. Also in type 2, only the final class, which is NODE, stores the geometries. Without "EDGE" class, we cannot expect such information as "which edge is this node belonged to?" or "which edge do these two faces share?" In a study (Stoter *et. al*., 2003), even more reduced form (Type 3) is introduced for storing 3D objects in a DBMS and visualization. While sufficing for the visualization purpose, such "de-normalized" relations as in type 2 or 3 suffer duplicated data storage and less capability of topological information retrieval. As of now, we could not find in the current literature the model having line geometry (Figure 2-(b)). Thus, we included this model (Type 2) and performed comparison tests along with other models in our study.

3. SIMULATION

3.1 Data Construction

Points to be considered in selecting 3D data models in this paper are as follows:

- Easiness of storing 3D objects
- Retrieval time of non-spatial operation query
- Retrieval time of spatial operation query
- Possibility of using spatial operation

Through performance comparison tests for the four models mentioned above, we present 3D data model suitable for 3D visualization and spatial operation. For the tests, we used a number of artificial cube objects and stored them in a spatial database as in figure 3. We used the PostgreSQL/PostGIS for the spatial DBMS.

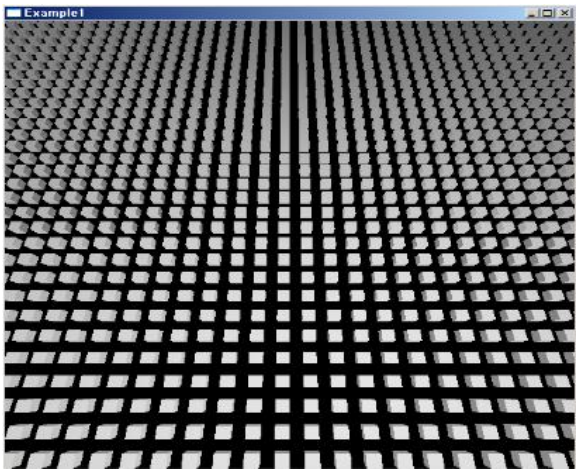

Figure 3. 3D objects used in the tests (100 × 100 size)

The first test measures the performance time for non-spatial operations. For visualizing 3D objects in VRML or OpenGL, coordinate values of polygons constituting a 3D object are needed. Thus, we built a SQL-query to get object ID, spatial feature ID and coordinate values. Table 1 is an example of queries executed for Type 1 and 3.

Another test is to measure the performance time for spatial operations. When range queries are executed, we measured time required to get objects' IDs and face information consisting the objects. We set up the range of 20×20 unit distance and obtained objects' values included in this range. Table 3 shows an example of queries for Type 1 and 3. As you can see from Table 1 and 2, spatial queries include spatial functions (i.e. intersect) in the join statements while non-spatial queries do not. We created 500,000 cube objects. And the tests were performed for the sub-ranges of 100, 1,000, 10,000 and 100,000 units. Each test was executed 10 times and the average of them was recorded.

Body-Face-Edge-Node
SELECT body.bodyid, face.faceid, edge.edgeid, node.nodeid, ST_AsEWKT(shape) FROM body INNER JOIN ((((face INNER JOIN (((face_edge INNER JOIN ((edge INNER JOIN (edge_node INNER JOIN node ON node.nodeid = edge_node.nodeid) ON edge.edgeid = edge_node.edgeid)) ON edge.edgeid = face_edge.edgeid))) ON face.faceid = face_edge.faceid)))) ON body.bodyid = face.bodyid ORDER BY body.bodyid, face.faceid, edge.edgeid, node.nodeid;
Body-Face-Node
SELECT body.bodyid, face.faceid, node.nodeid, ST_AsEWKT(shape) FROM body INNER JOIN ((face INNER JOIN (face_node INNER JOIN node ON node.nodeid = face_node.nodeid) ON face.faceid = face_node.faceid)) ON body.bodyid = face.bodyid ORDER BY body.bodyid, face.faceid, node.nodeid;

Table 1. Non-spatial query for Type 1 and Type 3

Body-Face-Edge-Node
SELECT Distinct body.bodyid, face.faceid FROM body INNER JOIN ((((face INNER JOIN (((face_edge INNER JOIN ((edge INNER JOIN (edge_node INNER JOIN node ON node.nodeid = edge_node.nodeid) ON edge.edgeid = edge_node.edgeid)) ON edge.edgeid = face_edge.edgeid))) ON face.faceid = face_edge.faceid)))) ON body.bodyid = face.bodyid WHERE ST_Intersects(GeomFromEWKT(node.Shape), GeomFromEWKT('POLYGON((0 0 0,20 0 0,20 20 0,0 20 0,0 0 0))'))=true;
Body-Face-Node
SELECT Distinct body.bodyid, face.faceid FROM body INNER JOIN ((face INNER JOIN (face_node INNER JOIN node ON node.nodeid = face_node.nodeid) ON face.faceid = face_node.faceid)) ON body.bodyid = face.bodyid WHERE ST_Intersects(GeomFromEWKT(node.Shape), GeomFromEWKT('POLYGON((0 0 0,20 0 0,20 20 0,0 20 0,0 0 0))'))=true;

Table 2. Spatial query for Type 1 and Type 3

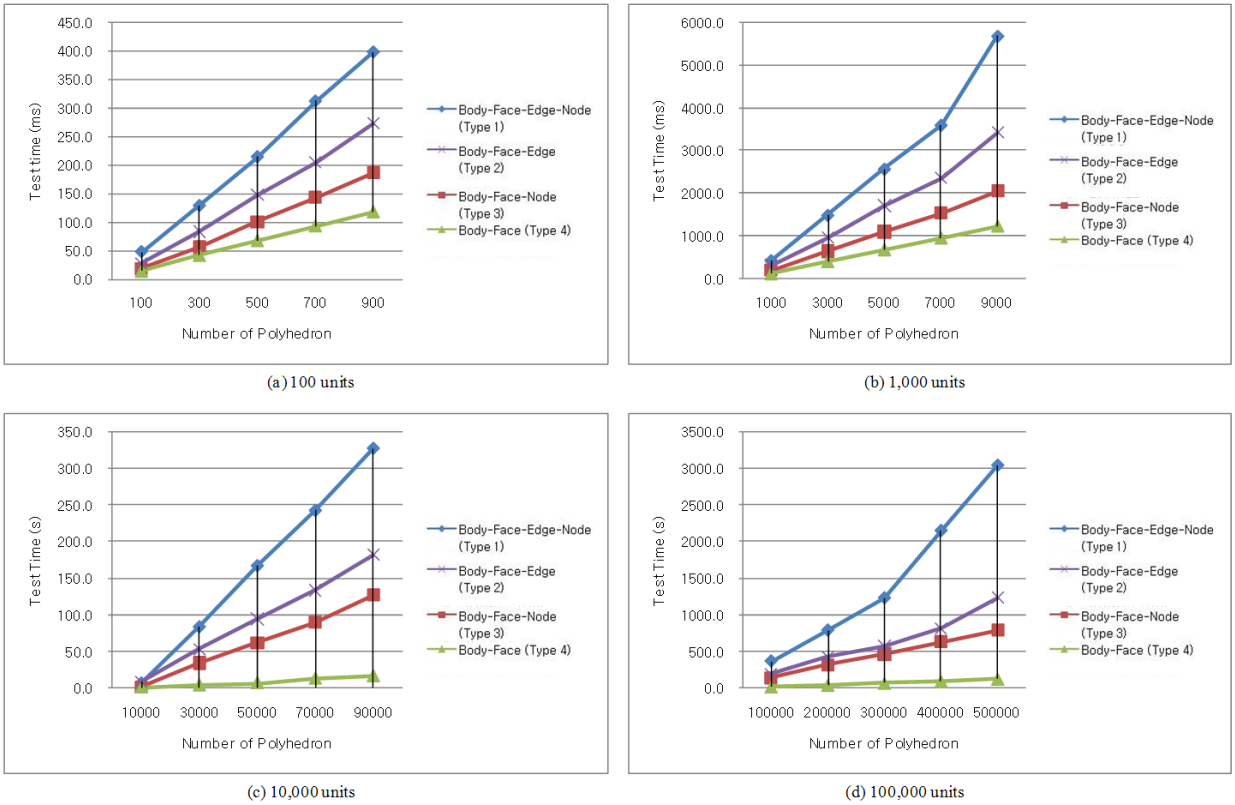

Figure 4. Performance results for non-spatial queries

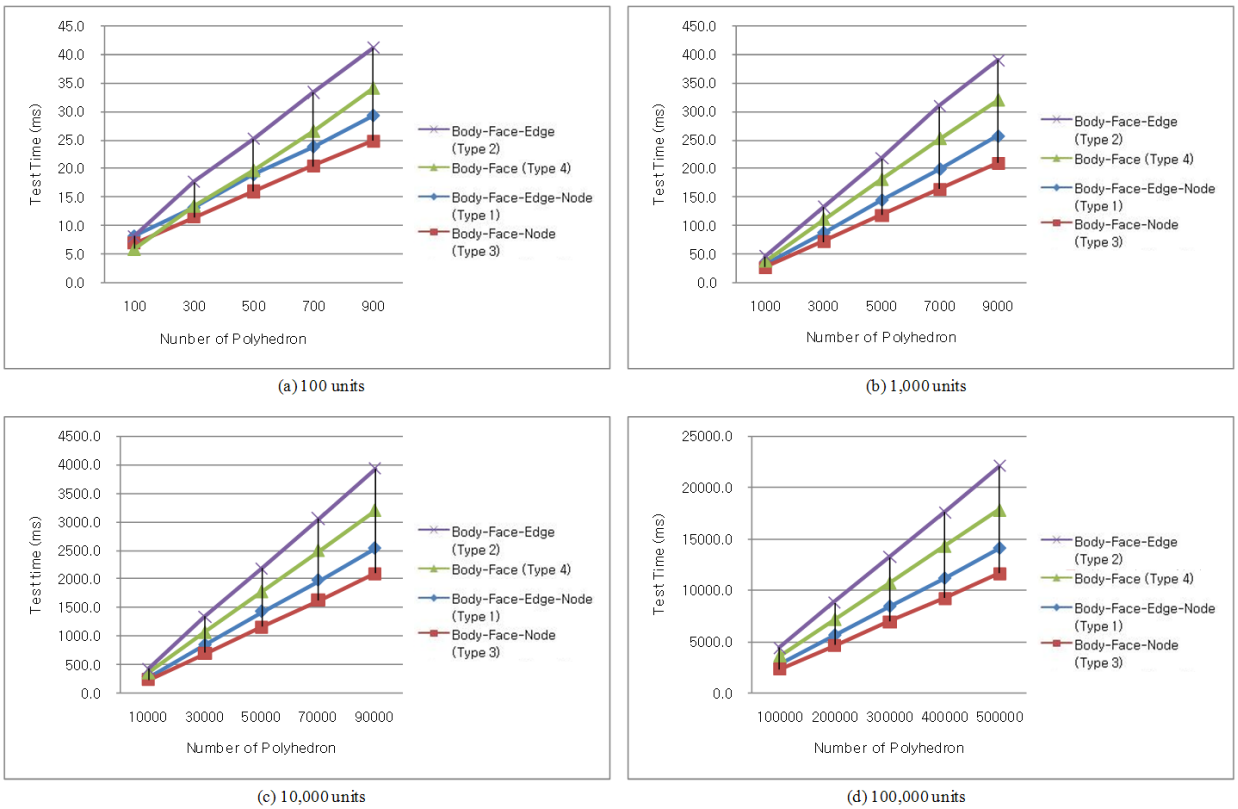

Figure 5. Performance results for spatial queries

3.2 Test Results

Figure 4 and 5 show the results of performance comparisons of the four types. For non-spatial queries, Type 4's retrieval time is fastest in performance followed by 3, 2 and 1 in order. As the number of 3D objects increases, query performance for all four models decreases. While Type 4 shows the smallest increasing rate, Type 1 shows the greatest. As can be seen, with less than 1,000 units, four models show differences up to 6 seconds, implying that all four models are suitable for visualization for the size of urban model including hundreds of building objects. However, with over 1,000 objects, the retrieval performances show significant gaps among the models, implying that Type 4 is the only alternative for visualization.

In the test for the non-spatial queries for 100,000 units in the PostGIS, the retrieval time of each model shows as follows; Type 1 – 369.3 seconds, Type 2 – 202.4 seconds, Type 3 – 143.0 seconds, Type 4 – 19.1 seconds. In the spatial query tests, Type 3's retrieval time is fastest in performance followed by 1, 4 and 2. All four models show steady increase as the number of objects increases. The reason that spatial queries show less time than non-spatial queries is we experimented using a portion (20 × 20) from entire objects, while non-spatial treats all the objects. The number of objects from the range query was around 10. From the result, we can see that spatial range queries can be applied to all four models for retrieving relatively small number of objects from less than 100,000 units, since the gaps between them area 5 seconds at most.

3.3 Visualization

VRML supports diverse base solid features and allows the combination of multimedia such as animation or sound with applications, which makes visualizing 3D models in VRML relatively easy. Low-level graphic libraries such as OpenGL and Direct X can describe objects more in detail. In the test, we used real 3D building models of around 500 stored in the PostGIS. Although both techniques showed similar quality, VRML showed significant decrease in refreshing speed as the number of objects in the display increases. Thus, it is viewed that OpenGL is more suitable for 3D visualization applications although development with OpenGL requires more time than VRML (Figure 6 and 7).

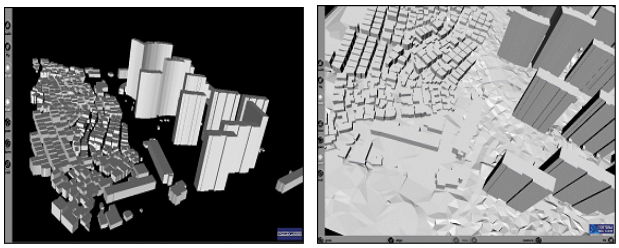

Figure 6. An example of visualization test using VRML

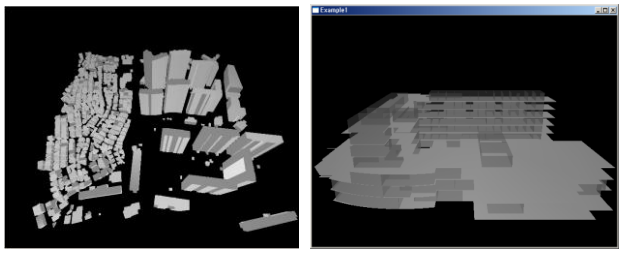

Figure 7. An example of visualization test using OpenGL

4. CONCLUDING REMARKS

Although 3D models are getting used increasingly in many areas including architecture, urban planning and environmental analysis, they mostly have been used as visualization purposes without using topological structure or semantic information. 3D topology models have been studied over the last decade and some of them were connected with DBMS-based implementations. However, we haven't found any attempts for comparing the models in the viewpoint of performances as of now. In this study, we categorized previously proposed 3D models and compared them including our own model using different queries to the SDBMS. We tested the retrieval time by non-spatial and spatial queries and suggested the most computationally favorable models for LADAR simulations. Also, we compared visualization performances between VRML and OpenGL. The results imply that choosing a 3D modeling should be done according to the problem requirement types. Type 1 is shown to be proper for visualization, while type 3 is for spatial queries. However, types of 3D model may additionally be determined by the level of details of the classes where the required semantic information resides. For example, when modeling indoor spaces, we many need a more decomposed model like Type 1 which fully implements the topological relationships between objects.

ACKNOWLEDGEMENTS

This research was supported by the Seoul R&BD Program (10561), Korea.

REFERENCES

Ellul, C. and Haklay, M., 2008. Using a B-rep structure to query 9-intersection topological relationships in 3D GIS – reviewing the approach and improving performance. In J. Lee and S. Zlatanova, (eds.), *3D Geo-information Sciences*, Springer-Verlag, Berlin, pp. 127 -151.

Gröger, G., Reuter, M. and Plümer, L., 2004. Representation of a 3-D city model in spatial object-relational databases. In: *The 20th Congress of International Society for Photogrammetry and Remote Sensing*, Istanbul, Turkey.

Kolbe, T., 2008. Representing and exchanging 3D city models with CityGML. In J. Lee and S. Zlatanova, (eds.), *3D Geo-information Sciences*, Springer-Verlag, Berlin, pp. 15 -31.

Stadler, A. and Kolbe, T., 2007. Spatio-semantic coherence in the integration of 3D city models. In: *The 5th International ISPRS Symposium on Spatial Data Quality ISSDQ 2007*, in Enschede.

Stoter, J. and van Oosterom, P., 2002. Incorporating 3D geo-objects into a 2D geo-DBMS. In: *ACSM-ASPRS 2002 Annual Conference*. Washington DC.

Stoter, J. and Zlatanova, S., 2003. Visualising and editing of 3D objects organised in a DBMS. In: *EUROSDR Workshop : Rendering and Visualisation*, Enschede, The Netherlands, pp. 14-29.

Zlatanova, S., 2000, 3D GIS for urban development, Ph.D. thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Austria, ITC, the Netherlands.