3D Spatial Models for Geometric Description Of Spatial Objects

Mochammad Zuliansyah, Suhono Harso Supangkat, Yoga Priyana, Carmadi Machbub
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
zulgiva@yahoo.co.id, suhono@depkominfo.go.id, yoga@lskk.ee.itb.ac.id, carmadi@lskk.ee.itb.ac.id

***Abstract*—Whilst thematic analysis and 2D spatial analysis are well studied, research on 3D spatial analysis is still at an intensive stage. Spatial relationships are the fundament of a large group of operations to be performed in Geographic Information System (GIS), e.g. inclusion, adjacency, equality, direction, intersection, connectivity, and their appropriate description and maintenance is inevitable. Similar to 2D variants, 3D GIS should be capable to perform metric (distance, length, area, volume, etc), logic (intersection, union, difference), generalisation, buffering, network (shortest way) and merging operations. Except metric operations, most of them require knowledge about spatial relationships. The third dimension increases drastically the number and complexity of all possible spatial relationships, compared with 2D GIS. A formalism for the detection of spatial relationships based on set topology notions has already been proposed by several authors, however the description of 3D spatial relationships is not sufficiently studied. The design of a spatial query language or the extending of existing languages, updating procedures to ensure topology consistency of database, etc., these are tasks that need further development. In this paper, the definition of a new spatial model will be given. It will be referred to as Geometric Domain Spatial Model. According to the proposed definition of an object, the geometry of each spatial object can be associated with four abstractions of geometric objects, i.e. point, line, surface and body.**

***Keywords*—3D spatial analysis, 3D spatial, Geometric Domain Spatial Model**

I. INTRODUCTION

Research in the GIS community is trying to work out a conceptual model capable of integrating geometric (position, shape and size) and thematic characteristics of objects and mutual spatial relationships. These models can be considered an explicit description of cells (or objects). The concept of a 2D GIS introduced by [8,10] as "Formal Data Structure", and extended to comprise 3D information and texture in several successor models, follows an integrated approach for describing geometry, semantic and spatial relationships in one spatial model. The model has been extensively studied (see [9], [3]). The investigations into 2D and 3D spatial analysis promised results that have motivated the utilisation of 3D FDS as a starting point of this paper. A review of the 3D extensions and implementations of the model are listed below. A modification of the initial 2D FDS has led to an extension, called 3D FDS, allowing operations with body objects. An experimental 3D Vector GIS (TREVIS) has been used to test the model (see [2]). Despite the promising results, the study on suitability of 3D FDS for urban areas is still not sufficient. Developed software for visualisation is hardware-dependent, with limited possibilities, and does not offer texturing (only wire frame).

The next extension, achieved by including a tetrahedron as a fourth describing element, results in the Tetrahedral Network (TEN), which makes available new operations with volumes (e.g. in geo-sciences and environmental management). A package of programs for 3D raster conversion, creating 3D Voronoi polyhedrons and Delaunay tetrahedrons and basic query analysis have been developed and tested. Analysis of the various models based on FDS yields a generalised concept for an ndimensional data model named Simplicial Network Data Model (SNDM). The implementation work on this data model is done using an object-oriented approach. The developed software, i.e. Integrated Simplicial Network Application Package (ISNAP), allows 2D and 3D triangulation (unconstrained and constrained), graphic display (orthogonal, perspective and stereo views, wire frame, surface illumination, etc.) of surface, query of point, line and face, derivation of contour lines and derivation of a regular grid DTM. Further investigations into the applicability of the model for more complex 3D analysis (e.g. intersection, buffering) are necessary. The software for visualisation of 3D data has to be extended. 3D tools for interactive editing still have to be developed.

The model is well studied in the spatial domain as well. In [2] have completed a study on binary topological relations that can be derived from 3D FDS. The suitability of 3D FDS and its 3D extensions has still not been sufficiently studied for 3D visualisation and remote access. Despite some experiments with urban models, the results obtained are only preliminary: the models are mostly visualised in wire frame, no texture is applied and no real-time navigation is intended. The GIS models still deal only with spatial objects (with discernible and indiscernible boundaries) and do not provide a framework for the accommodation of non-spatial objects.

In [4] presents a framework for a 3D GIS, which maintains different views per object, e.g. point objects can be displayed by different symbols. The topology handled follows Molenaar's approach with several extensions: each point knows its adjacent lines, every face knows its bounding lines, and every line knows its bounding faces. Thus the bidirectional links between

all the cells are stored. The model is intended for implementation in object-oriented DBMS as the GUI is built using OpenGL. The author warns about the large amount of data produced as a result of the explicit storage of many relationships.

In [5] presents an idea to incorporate Constructive Solid Geometry (CSG) primitives in the topological model in order to facilitate the data acquisition by CAD systems. The author introduces six ATDs for topographic objects and four ATDs for primitives and thematic classes. The CSG primitives are decomposed into defined ATD classes before storage in the database.

II. DESIGN

In this section, the definition of a new spatial model will be given. It will be referred to as Geometric Domain Spatial Model. According to the proposed definition of an object, the geometry of each spatial object can be associated with four abstractions of geometric objects, i.e. point, line, surface and body. A point is a spatial object that does not have shape or size but position is the space. A line is a type of a spatial object that has length and position. A surface is an abstraction of spatial object that has position and area. A body is a type of spatial object that has a position and a volume. All the Geometric Object (GO) are built of smaller, simpler elements, i.e. constructive objects. The model consists of two constructive objects (CnsO), i.e. node and face. The formal definition of the spatial model establishes the rules according to which an object can be composed, clarifies allowed configurations and specifies the topological primitives (closure, boundary, interior and exterior) needed for the later elaboration on neighbourhood relations. Furthermore, we assume that all the objects are embedded in Euclidean 3D-space, denoted by IR^n where $0 \le n \le 3$. The formalism employs fundamental definitions, theorems and concepts of set theory (see [6] and [7]) and linear algebra [1]. The basic category utilised in the definitions is the one of indexed sets. The index gives a unique identification of any spatial object, which facilitates many stages of the implementation.

The data structure presented here is not complete implementation of all the components of the object defined above. We concentrate mainly on **GA**, **GR** and **GB**, assuming that they represent the most important information to create VRML documents and have crucial impact on the successful communication between client and server (see Figure I).

We will as it was stated above, geometric description (**GDsc)** is influenced by a number of factors: the purpose of the application (e.g. environmental analysis or town planning), the method for data collection, the rendering engine used for visualization, etc. Boundary representations (B-reps) seem to be the best suited description for urban modeling due to 1)the prevalent attention to the surfaces of the objects, and 2) mostly surface measurements to build geometry. On another hand most of the rendering engines (VR browsers in our approach) are based on Breps. Some recent developments in 3D reconstruction of man-made objects draws the attention up to the constructive solid geometry (CSG) structures, which in practice is realizable in VRML. We consider, however, the storage of real measurements a crucial requirement for our data structure. Since it is more difficult to be realized in Constructive Solid Geometry (CSG) structures, we concentrate on B-reps description without dipper elaboration.

The **CnsO** in B-reps are points, lines (arcs) and faces, which are used in different combinations in CAD and GIS structures [2]. The model presented here is based on existence of two of these **CnsO**, i.e. points named *nodes* and *faces.* This is to say that a *surface* and *body* object will be described as a set of *faces*, while *line* object will be a set of *nodes* and *point* object will be the *node* containing the coordinates (see Figure 1 and 2).

The most significant spatial relationships for the VRML document among *geometric objects* and their *composing objects*, are the relationships object-*face* and *face-node*. The query for creation of VRML documents starts always from the object level and ends at the lowest *constructive object* level, i.e. *nodes*. This requires an *object-facenode* traverse of the data base, which is assured by storage of boundary relationships, e.g. *face* is a list of *nodes*, *body* is a list of *faces*.

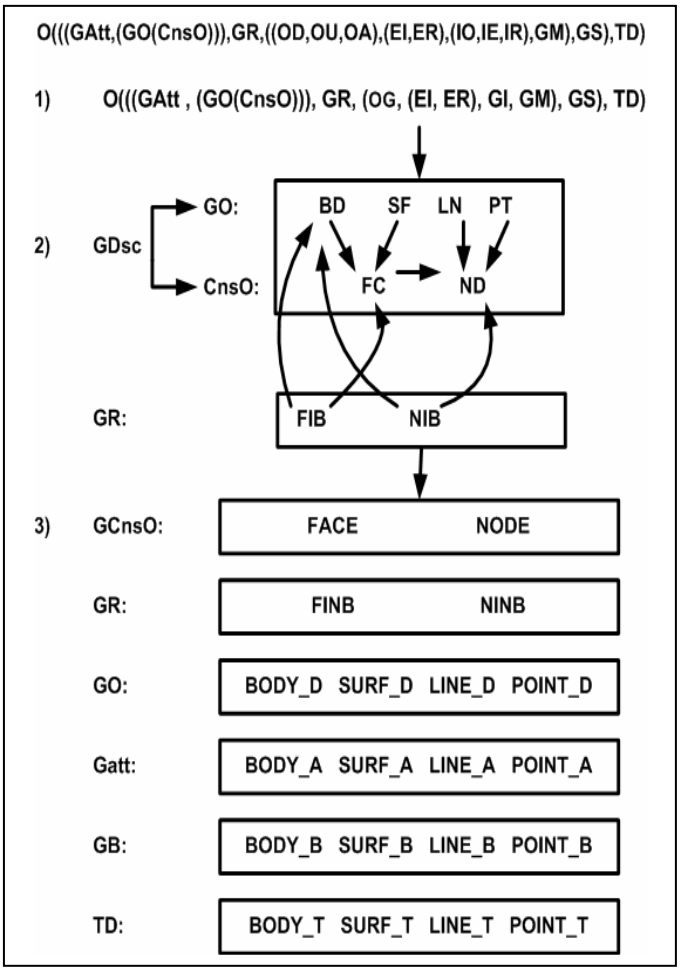

Figure 1. Implemented components: GDsc, GAtt, GB

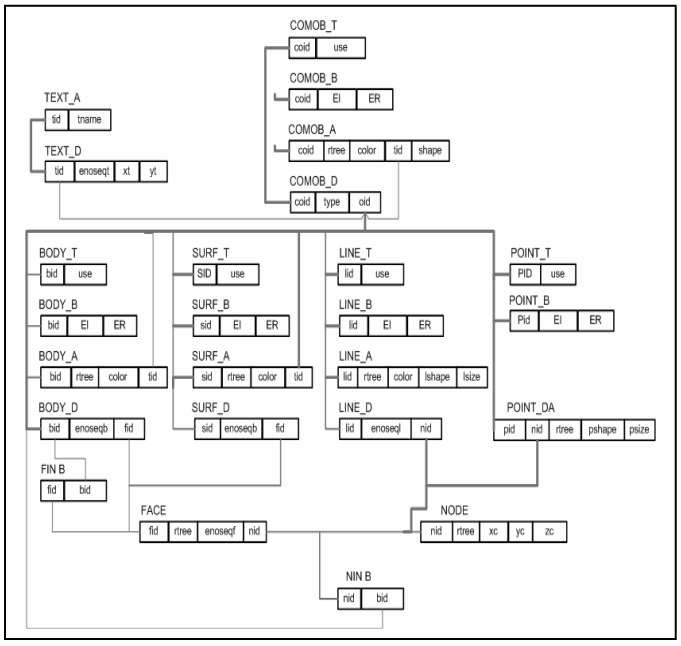

Figure 2. Relational data structure

In order to simplify the *composing rules* to create *composite objects,* we establish the following rules for composition of objects in geometry domain:

geometric objects (GO(CnsO)): The major principle is aggregation, i.e. the *composite object* is a child, containing all the primitives of parent objects.

geometric attributes (GA): The rules are aggregation and inheritance. If the *composite object* has its own geometric attributes they are dominant, e.g. a *surface*, presented as a composite object of a street, canal and parcel, can have a *geometric attribute* texture covering the three parent objects.

behavior (GB): Maintenance of behavior of composite objects is similar to the geometric attributes. In general, two distinct cases are possible: 1)the child have an individual behavior, which is dominant for the *composite object* and 2)the child does not have behavior, i.e. the parent behavior can be activated individually.

Bearing in mind the assumptions about *composing rules*, the *composite object* can be expressed as a list of composing objects.

The data are organized in a relational data model as each component (**GDsc**, **GAtt**, **GB**) is represented by one relational table (e.g. SURF_D, SURF_A, SURF_B) (see Figure 1 and 2). Thus each **GO** (*body*, *surface*, *line*, *point* and *composite object*) has three relational tales. The R-tree leaves and non-leaves are organized in relational tables containing information about **MBB** (minX, minY, minZ, maxX, maxY and maxZ coordinates of a R-tree box) and the identifiers of the three children leaves (non-leaves). The number of entries, i.e. three, was chosen among experimented 2,3,4 and 5 entries per non-leave. The aim was to achieve such groups of objects in the height *h-1*,i.e. the first non-leave level which can be used to create one LOD. This means that the **MBB** has to give an idea about the shape of the three grouped objects. As an experimented criterion for grouping was used: the minimal oblique distance, the minimal horizontal distance, and the min-max angle, between weight centers of the objects. The best results were achieved with criterion min distance and min-max angle between mass centers of the objects. An additional column with the position in the r-tree was included in the *attribute* tables of each **GO** and **CO** and **CnsO** in order to facilitate the traversal of the FACE and NODE tables. Note, the FACE and NODE table contain all the *faces* and *nodes* in one 3D model.

The method for creating LOD for visualization is expected to perform satisfactory results for buildings, however, large surface objects will cause visualization artifacts. Suppose the DTM is one object, its bounding box will cover the entire model and in case of rough relief will hide large parts of the model. Apparently either these LOD should not be applied to such surfaces, or the surfaces should be subdivided further into smaller parts.

III. PERFORMANCE

The last study examines the performance of GDSS. The results contribute to the verification of the model and the overall evaluation of the system architecture. GDSM was proposed as an alternative to 3D FDS for our system architecture. In this respect, the definition of GDSM and the logical model GDSS are conceptually related to 3D FDS. Therefore, the basic idea of the test is a proof of the improved performance of GDSS with respect to 3D FDS. Single aspect of the performance are investigated here, i.e. size of the database. The performance test concerning size concentrates on the effect of three major concepts in GDSS: 1) the elimination of arcs and modified representation of some relationships, 2) the maintenance of R-tree tables and fields for codes and 3) the storage of geometric attributes and behaviour. While the reduction in the database size due to arc removal can be predicted, the effect of modified relationships and the storage of additional data is difficult to evaluate. This test investigates whether the modified geometric description of GDSS provides a sufficient reduction to compensate for the size of the new included data. If this is the case, the tests will be considered successful, i.e. GDSS ensures more efficient data organisation than 3D FDS.

A. *Size performance*

To evaluate the impact of R-tree tables on the data volume, appropriate calculations of their size are provided separately (see Table 1). The leaf table is only one and contains the identifiers of the objects and minimum maximum co-ordinates of the bounding per object (body, surface). Non-leaf tables have non-constant numbers and depend on the height of the R-tree, respectively on the number of object stored. The total number of objects in the Enschede data set is 26, in Vienna 1600. Consequently, the height of the R-tree for Enschede data is three and for Vienna data seven. A record in the non-leave table contains the identifier of the current non-leave, three identifiers of the sub-tree and the min-max co-ordinates of the MBB. Since the number of records is different for each R-tree table, the total number of the records in all the non-leave tables is given. According to the position in the R-tree, geometric objects and constructive elements receive a code, which is recorded in an extra field in the _A tables (for objects), FACE

and NODE. Since they do not introduce new records, the total number of records is given by the sum of R-tree tables (see Table 1).

TABLE I. SIZE OF R-TREE TABLES AND CODES

		Enschede			Vienna		
	b/r	Num. tab.	num. Rec.	bytes	Num. Tab.	num. rec.	bytes
R-tree leaves	26	1	26	26	1	1600	41600
R-tree non-leaves	32	3	13	416	7803	25696	
Code body_A	4	0	11	44	0	1600	6400
Code surf_A	4	0	19	76	0	0	0
Code face	4	0	4834	19336	0	92268	4E+05
Code node	4	0	960	3840	0	30756	1E+05
Total	*	4	39	23738	8	2403	6E+05

Finally, Table 2 summarises the size cost 3D FDS, the content of GDSS equal to 3D FDS (denoted with GDSS-). GDSS without R-tree tables (denoted by GDSS) and GDSS including R-tree tables (GDSS+).

TABLE II. SIZE COMPARISON: GDSS VS. 3D FDS

Name	Byte/ record	Enchede Record number	Vienna byte	Record number	byte
FDS	161	9763	138522	168205	2401576
GDSS-	156	7365	79974	143202	1638956
GDSS	411	7445	81879	144802	1690156
GDSS+	*	7484	105617	147205	2255948

GDSS in all three variants presented requires less disk space than 3D FDS. A comparison of size between 3D FDS and GDSS- reveals that 3D FDS is almost twice as large. The volume of data for GDSS- is 42% and for 3D FDS 32% respectively for Enschede and Vienna. The table NODE is the same in both models, the two groups of tables BODYOBJ & BODY_T and SURFOBJ & SURF_T are almost identical. The number of surface and body objects is different but influence on the volume of data is minor. Clearly, the biggest difference comes from 1) the number of CnsO maintained (face in GDSS versus arc, face and edge in 3DFDS), and 2) the manner of representing the GO (surface and body) by CnsO, i.e. FACE, BODY_G and SURF_G tables in GDSS, FACE and EDGE tables in 3D FDS. In the following analysis, we will assume that the size of GDSS- is approximately 35% of the size of 3D FDS and we will evaluate the influence of the ARC table and different geometric representation separately.

The ARC table occupies about 20% (Enschede) and 13% (Vienna) of the total storage space of 3D FDS. The fewer ARC records in the Vienna data set are caused by the lack of DTM. The ratio node:arc:face, which is usually quite stable for TIN (1:3:2), is 1:2.5:1.6 for Enschede and 1:0.8:0.6 for Vienna. This is to say that the Enschede data set is an example of almost completely triangulated surfaces. In contrast, the Vienna data set contains only faces with four and more nodes. These figures are an indication that the size of the ARC table can vary from data set to data set but cannot decrease below 10-12% and cannot increase above 20-25%. Hence, the average "cost" of arc's existence is evaluated at about 18% of the total size of 3D FDS.

The second factor that contributes to the improved performance of GDSS is the different geometric representation of body and surface. The table FACE (GDSS) is conceptually similar to the table EDGE (3D FDS), i.e. both of them represent the relationship between face and the next low dimensional CnsO: arc (3D FDS) and node (GDSS). They differ in the relational implementation: 10 bytes in GDSS against 13 bytes in 3D FDS. This is an indication for the more expensive face_arc than the face_node relation. Table FACE (3D FDS), which represents the co-boundary relationships face_body and face_surf, does not have an equivalent in GDSS. BODY_G and SURF_G are the two new tables, which contain the boundary relationships body_face and surf_face. In general, the information that can be extracted from FACE and EDGE table in 3D FDS is almost identical to the information of BODY_G, SURF_G and FACE in GDSS. Consequently, we should evaluate them together, i.e. the size of FACE+EDGE versus FACE+BODY_G+SURF_G tables. Despite the slight difference between EDGE (3DFDS) and FACE (GDSS), they can be ignored to show the space needed for the relations among face, surface and body only (see Table III).

TABLE III. FACE+EDGE (3D FDS) VS. FACE+BODY_G+SURF_G (GDSS)

	Relational Table	Scheme	Enschede Byte	Vienna Byte
1	FACE + EDGE	FDS 3D	93502	1574044
2	FACE + BODY_G + SURF_G	GDSS	63670	1108460
	Difference 1-2	FDS 3D-GDSS	29832	465584
3	FACE	FDS 3D	30660	371560
4	BODY_G + SURF_G	GDSS	15330	185780
	Difference 3-4	FDS 3D-GDSS	15330	185780

As can be seen, the explicit boundary (body-face, surface_face) representation of these relations is much "cheaper". The differences in the volumes of data obtained from the two representations of surface, body and face, i.e. the tables containing information about them, is denoted as difference 1-2. The difference in representations of surface and body is denoted as difference 3-4 (see Table II). Table III shows the impact (in %) of the discussed volumes of data on the size of 3D FDS.

TABLE IV. THE COST OF ARC TABLE AND THE GEOMETRIC REPRESENTATION

	Enschede Byte	Vienna Byte	Enschede % dari FDS 3D	Vienna % dari FDS 3D
FDS 3D	138552	2401576	100%	100%
GDSS-	79974	1638956	57%	68%
ARC	28836	300036	21%	12%
Difference 1-2 (Table II)	29832	465584	21%	19%
Difference 3-4 (Table II)	15330	185780	11%	7%

It can clearly be seen that the sum of the data contained in GDSS-, the ARC table and the difference in geometric representations (i.e. difference 1-2) are approximately equal to the size of data in 3D FDS. Thus, the tests and the analysis have verified that geometric representation of the GDSS is more efficient than 3D FDS. Moreover, the better performance is due to reversal of geometric representations (from co-boundary to boundary) and elimination of the ARC table.

TABLE V. THE COST OF GB, GA AND R-TREE TABLES

	Enschede byte	Vienna byte	Enschede Enlargement in% of GDSS-	Vienna Enlargement in% of GDSS-
GDSS-	79974	1638956	100%	100%
GDSS	81879	1690156	2%	3%
GDSS+	105617	2255948	32%	37%

The enlargement of GDSS with additional information (behaviour, colours, and textures) and corresponding R-tree tables and codes, still does not exceed the size of the 3D FDS (Table I). GB and GA increase the size of database by only 2-3% (see Table IV). It should not be forgotten that the size of the images for texturing is not considered. Here, only the parameters maintained in GDSS are considered. The disk space occupied by GDSS+, i.e. GDSS including the R-tree and the codes is about 30% larger than GDSS. This number includes the size of the R-tree tables and the additional fields for the codes in the tables for CnsO and GO. The impact of the R-tree tables is minor, i.e. about 2% of the total size of GDSS+. The enlargement is a result of the codes introduced. The main contribution gives the FACE table. Since the type of relations kept there is 1:m, further normalisation of the FACE table will improve the performance. The test verified that the supplementary information including the R-tree representation lead to a size that is compatible (even smaller) with the size of 3D FDS.

B. Time performance

The tests are performed under the several assumptions and simplification listed below:

- Since the key issue of our approach is visualisation of 3D spatial analysis, the performance test related to time focuses only on queries, which result in a VRML document
- Even though the outcome of the query might be a **CnsO**, the VRML document is to be created including the **GO** (**GO**s), which contains this particular **CnsO**. In this respect, the visualisation of spatial queries passes two compulsory phases. First, the data needed to complete the user query is specified and, second, the data to create the VRML document is extracted. The objects included in the VRML document may vary considerably depending on the preferred manner for representation. Irrespective their number and way of representation, all the objects require the set of standard parameters for scene design structured according to the VRML syntax orientation, texture, texture co-ordinates, colour and a number of minor variable parameters. We will refer to the query that extracts data for a VRML creation as a *visualisation query*.
- The queries are simplified to extract only geometric description (the colour is constant). Since the parameters for visualisation might be organised in a similar way in 3D FDS, the issue is not relevant for testing.
- The tests conducted here refer to visualisation queries as the result of simple user queries. The first argument for this restriction is the specifics of the visualisation queries, i.e. they require traverse of all the tables concerning geometric description. The second argument is that the eventual bad performance of such queries will be an indication of even worse performance of complex user queries. The last argument refers to the variety of user queries, which may be quite significant and require special schema for investigations.
- The experiments are based on representative queries that are embedded SQL statements. The geometric description in VRML differs significantly from the geometric description in both the conceptual models. This is to say that an SQL query cannot extract the needed subset of data. However, a particular subset of data extracted in a certain sequence can be formulated in an SQL query and further reorganised to match the VRML syntax. Thus, the visualisation query in our system is composed of two distinct steps: first, extraction of the data by an SQL statement (the data are the ID of the faces of a particular object (body or surface), the order of the nodes in a face and co-ordinates of the nodes; second, further reorganisation of the data by a host language (in our case Perl, the language used to write CGI scripts)
- The visualisation queries are typical select operations and the SQL operator SELECT is therefore used to extract the needed data from the database. The SELECT SQL operator may or may not include the two phases (i.e. user and visualisation query) in one statement. For example, the query "visualise the buildings inside certain area" can be expressed by one SQL statement while the query "check for duplicated points" cannot be completed with one SQL statement

The time for completion of the query is tested first internally at a database level and second externally at the client site. The first experiments are pure database SQL queries executed on the server inside the RDBMS. The time for data extraction is

provided automatically by the RDBMS at the completion of the query. The time for creation, transmission and parsing of a VRML document is registered manually. The time considered is between the moment of starting CGI scripts and the complete display of the result in VR browsers.

TABLE VI. ENCHEDE TEST SITE: INTERNAL AND EXTERMAL TEST

Objects	3D FDS	GDSS Internal test	GDSS External test	Number of Vertices	Number of faces	Number of database records
One building	14sec	0.2 sec	2 sec	16	10	48
One surface	4sec	0.06 sec	2 sec	11	1	12
Composite object	20sec	0.2 sec	2 sec	24	15	72
DTM	15min	30 sec	50 sec	703	1399	4197
Entire model	-	40 sec	60 sec	842	1533	4293

TABLE VII. VIENNA TEST SITE: INTERNAL AND EXTERMAL TEST

Number Buildings	3D FDS	GDSS	Number of vertices	Number of Faces	Number of Database records
1	7 min	15 sec	22	13	66
2	13 min	30 sec	42	25	126
10	47 min	3 min	138	89	414
20	-	6 min	366	223	1098
50	-	13 min	1072	636	3216
200	-	27 min	4028	2414	12084
400	-	56 min	7930	4765	30938
600	-	-	12046	7223	36138
1600	-	-	30756	18578	92196
BID 818		40 sec	62	33	186
BID 773		50 sec	80	42	240

The results demonstrate faster traverse of GDSS tables compared with 3D FDS tables. The better performance of GDSS, however, is not sufficient for real work in a client-server environment. The results obtained for the Enschede data set (small data set) are satisfactory for small subsets and disappointing for large ones (e.g. DTM needs 50 sec external time). The traverse seconds increase drastically in the case of large models (Vienna), e.g. 200 buildings (about several neighbourhoods) already need 27 minutes internal time and 40 minutes external time. As mentioned before, the external time is influenced by a broader spectrum of factors (server occupation, Internet connection, host programming language), the internal time is precisely the traversing time of the tables.

IV. THUS THE DATA NEEDED FOR VRML DOCUMENTS ARE CONSTANT, I.E. CO-ORDINATES, FACES, CONCLUSIONS

Since the scope of the paper was restricted to the geometric domain, the geometric characteristics of objects, i.e. attributes, description, relationships and behaviour, were further elaborated. This thesis advocated the hypothesis that a 3D topological model can be adopted for visualisation and real-time navigation. The presented review and comparison of three 3D topological models has revealed deficiencies concerning either 3D modelling, or spatial analysis or 3D visualisation and interaction. Therefore a Geometric Domain Spatial Model (GDSM) to describe 3D geometry and represent spatial relations was proposed. The model is similar in some concepts to 3D FDS and differs mainly in the number of constructive objects. The model maintains two constructive elements, i.e. nodes and faces (arbitrary number of nodes, convex, planar, oriented, without holes), which are used to build four geometric objects, i.e. body, surface, line and point. Nodes constitute faces, points and lines, faces constitute surfaces and bodies. Since the model is not a complete subdivision of the space, two relations, i.e. face-in-body and node-in-body, are explicitly described. In order to prove the capability of GDSM to perform 3D spatial analysis, the ability of GDSM to distinguish binary topological relations (according to the 9-intersection model) between geometric objects was extensively analysed. Since the possible relations between simple objects in 3D space were not fully studied, negative conditions applicable for 1D, 2D and 3D space were first derived and then the possible relations were computed. An alternative approach to determine possible relations as well as sketches of all the object configurations has verified the obtained results. Sixty-nine topological relations between two simple objects (regardless of the dimensions of the object and the space) can be identified by the 9-intersection model. Afterwards GDSM was estimated for its potential to detect all the possible topological relations. The operations needed to identify all the relations were formally described using set theory notions.

REFERENCES

[1] H. Anton, "Elementary Linear Algebra with Applications," 9th Edition, John Wiley&Sons, Inc., New York, USA, 864 p, 2005

[2] C. Ellul, and M. Haklay, "Deriving a Generic Topological Data Structure for 3D Data," presented at Topology and Spatial Databases Workshop, Glasgow, UK, May 5-6. , 2005

[3] J. Jin, N. An, and A. Sivasubramaniam, "Analyzing range queries on spatial data," In ICDE'00. Washington, DC, February 28 - March 3, 525-534, 2000

[4] J.F. George, D. Batra, J.S. Valacich, J.A. Hoffer, "Object-Oriented System Analysis and Design, Prentice Hall, 2004

[5] R. Reis, M. Egenhofer, and J. Matos, "Topological relations using two models of uncertainty for lines," 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences, 286-295, 2006

[6] S. Lipschultz, "Theory and problems of set theory and related topics," Schaum Publishing Co., New York, 1998

[7] A.V. Arkhangel'skii, V.I. Ponomarev, "Fundamentals of General Topology: Problems and Exercises," Kluwer, 2001

[8] M. Molenaar, T. Cheng, "Fuzzy spatial objects and their dynamics," ISPRS Journal of Photogrammetry and Remote Sensing, Pages 164-175, 2000

[9] P. Wanning, P. Dragan, and C. Crawford, "Handling Large Terrain Data in GIS," XXth ISPRS Congress, Istanbul, Turkey, p. 281, 2004

[10] M. Zuliansyah, C. Machbub, S.H. Supangkat, dan Y. Priyana. "Linear Approaches for using Geometric Primitives for Calibration and 3D Modelling", Proceedings International Conference on Instrumentation, Communication and Information Technology, Bandung, p: 407-412, 2005