A HYBRID GIS FOR 3-D CITY MODELS

Xinhua WANG, Armin GRUEN

Institute of Geodesy and Photogrammetry
Swiss Federal Institute of Technology (ETH) Zürich
ETH Hönggerberg, CH-8093 Zürich, Switzerland
wang@geod.baug.ethz.ch agruen@geod.baug.ethz.ch

Working Group IV/1

KEY WORDS: 3-D City Model, Hybrid GIS, Relational Database.

ABSTRACT

With the development of modern cities, 3-D spatial information systems (SIS) are increasingly required for spatial planning, communication systems and other applications. Usually, several important topics are involved in 3-D urban information systems, i.e. data acquisition, modeling, management and handling. The first issue is addressed with our CyberCity-Modeler (CC-Modeler) that has been developed in order to generate structured data for city models from photogrammetrically measured points. This paper aims at reporting about the status of the development of our spatial information system CC-SIS, which is specially designed for the handling of 3-D city data, and the integration of raster images and vector data on the basis of a hybrid GIS.

1 INTRODUCTION

3-D urban GIS became an important issue in the recent past due to the increasing demands for a realistic representation of the environment. There are usually three types of data sets involved: Digital Terrain Model (DTM), objects like buildings, roads, and waterways, etc. and original images or derived orthoimages.
The DTM is the most standardised data type in a 3-D city model, and it usually is represented as regular grid, Triangular Irregular Network (TIN) or in hybrid form. Many investigations have been performed concerning the implementation of those representations (Fritsch, Pfannenstein, 1992). Considering the complexity of the terrain structure in most city areas, the TIN structure is preferred as a more suitable structure for DTM representation.
As to the representation of objects, several methods for 3-D object description have been investigated. Those models can be grossly subdivided into Wire-Frame, B-Rep, VBR, Cell-Decomposition, FBR, CSG, E-complex, FDS [Molenaar 1992, Rikkers, Molenaar, 1994]. Breunig, 1996 has discussed their suitability and insufficiency for 3-D GIS by comparing the different 3-D representations according to the following criteria: domain, validity (geometric and technical), non-ambiguity and unambiguity, close operation, efficiency of geometric algorithm, accuracy, need for storage.
In most applications, images are used as realistic texture data in order to present details, which are not modelled in the vector data set and give information about material properties. Therefore, for visualisation purposes, images may even compensate for lack of modelling of fine details. From a data structure point of view, images are expressed as 2-D raster data, which can be stored or manipulated as a special layer in our 3-D spatial information system.
This situation led us to develop an appropriate data model, which should not only represent the geometrical information, but also implicitly or explicitly describe the topological relationship. In the case of texture mapping, it also must have the ability to manipulate raster images.
Two important research topics are treated by our group: (a) the generation of the topology of 3-D objects by using photogrammetric tools; (b) the investigation of the data model and the development of a prototype system to manage the vector data and raster images based on the relational database technology. The detailed technical description of the former problem is addressed in Gruen, Wang, 1998 with our CC-Modeler (CyberCity Modeler). Here, we will present a technology for management of data, which is implemented in our CC-SIS (CyberCity Spatial Information System).
This paper is arranged in the following manner: In section 2, the conceptual model will be presented. It consists of a self-developed 3-D data structure (V3D), in which the geometrical, topological, texture and thematic information is defined. The configuration and the issue of implementation based on relational database technology are addressed in section 3. An application prototype is presented in section 4.

2 CONCEPTUAL MODEL

V3D is a hybrid data structure. It not only models 3-D objects, but also combines raster images and attribute information for each object. The objects are grouped into four different geometric object types:

- Point Objects: zero-dimensional objects, which have a position but no spatial extension.
- Line Objects: one-dimensional objects with length as the only measurable spatial extension. Line Objects are built up of connected line segments.
- Surface Objects: two-dimensional or two and half-dimensional objects with area and perimeter as measurable spatial extension, which are composed of facet patches.
- Body Objects: three-dimensional objects with volume and surface area as measurable spatial extensions, which are bordered by facets.

In V3D, each special object is identified by Type Identifier Code (TIC), referred to as PIC, LIC, SIC and BIC, respectively. Two data sets are attached to each object type: thematic data and geometric data. The image data can be attached to the surface object, body object and DTM object. In fact, the thematic data attributes are built up in a separate data table. It is linked to the object type with a related class label. The definition of the thematic data is user-dependent. The geometric data set contains the geometric information of 3-D objects, i.e. the information of position, shape, size, structure definition, and image index. The diagram in Fig. 1 shows the logical data structure.

For the four object types, four geometrical elements are designed, i.e. *Point*, *Edge*, *Facet* and *Entity*. *Point* is the basic geometric element in this diagram. The *Point* can present a point object. It also can be the start or end point of an *Edge*. The *Edge* is a line segment, which is an ordered connection between two points: begin point and end point. Further, it can be a straight part of a line object or lie on a facet. The *Facet* is the intermediate geometrical element. It is completely described by the ordered edges that define the border of the facet. One or more facets can be related to a surface object or *Entity* geometrical element. Moreover, *Facet* is related to an image patch. *Entity* is the highest level geometrical element, and it can carry shape information. An entity is completely defined by its bordering facets. Image data and thematic data are two special data sets, which are built up in two seperated data tables. Each facet is related to an image patch through a corresponding link.

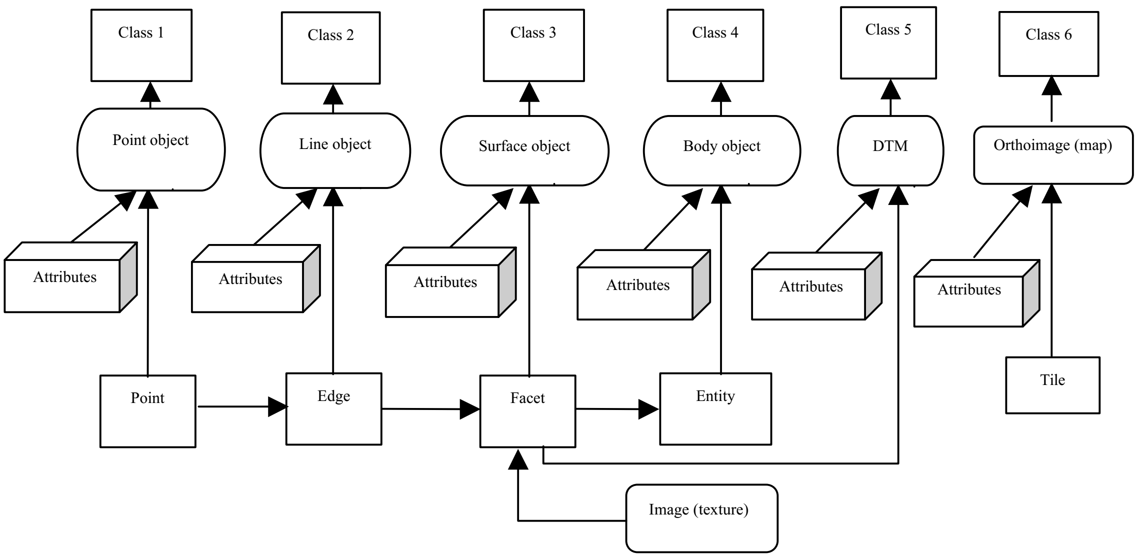
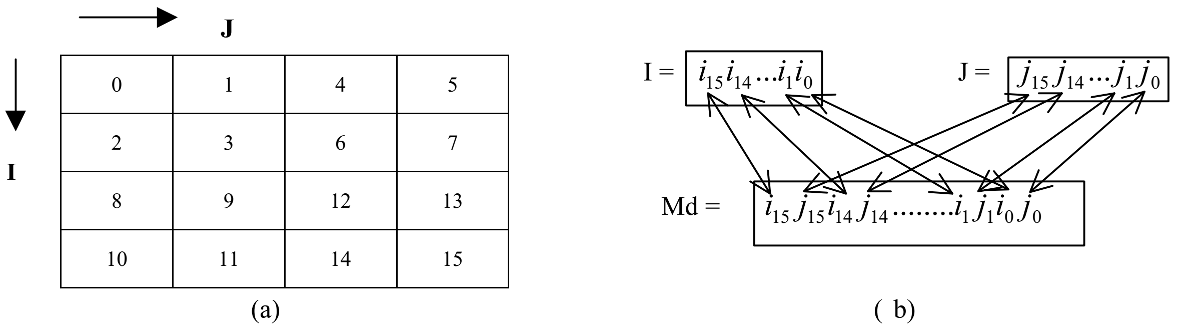


Figure 1. The conceptual model of V3D

Once the attribute table is attached and the TIC is labelled, a geometrical element becomes an object type. Obviously, the topological relationships between geometrical elements are implicitly defined by the data structure. A point object is presented by a distinct *Point* element. The line object is described by ordered *Edges*. The *Facet* with the information of image patches describes the surface object. Similarly, the body object is described by *Entity* that is defined by the facets. Thus, the topological relationships between *Point* and *Edge*, *Edge* and *Facet*, *Facet* and *Entity* are registered by the links between the geometrical elements.

DTM is treated as a special data type, which is described by a series of facets. Also, a raster image or a raster map is considered another special class. In a general 3-D urban GIS, three main classes of spatial raster data can be identified: raster images (ortho images), raster maps, and regularly spaced DTM. Since the vector TIN structure is employed to model a DTM in our system, the raster data refers to the raster orthoimages or raster maps. A common structure for the organisation of raster data is to partition the space into tiles of equal size. The raster is a two dimensional data structure where each cell has an associated two-dimensional coordinate value. Due to the one-dimensional nature of computer storage, there has to be mechanism to linearlize the two-dimension raster data. The most popular type of methods of ordering two dimension space based on spatial proximity is the space filling curves. The most popular among the space filling curves are the row order, row-prime order, Morton order, Hilbert-Peano order, cantor-diagonal order, and the spatial order. Here, the method of Morton order is employed in our system. Since the Morton code is a kind of variable position code, it can be employed not only as the index, but also as the "pointer" to build the relation between sub-raster blocks, or different data blocks (such as the orthoimages with DTM and 3-D objects).

The Morton order organises raster cells in a quadrant-recursive fashion. This means that each sub-quadrant is visited (or filled) before moving to the next sub-quadrant. Morton codes or keys for raster cells are easily computed by interleaving the bits of theirs row and column values. Fig 2 shows the Morton order for 16*16 pixel raster and the computation of Morton codes.


J
I
(a)
I = $i_{15}i_{14}\cdots i_1 i_0$
J = $j_{15}j_{14}\cdots j_1 j_0$
Md = $i_{15}j_{15}i_{14}j_{14}\cdots\cdots i_1 j_1 i_0 j_0$
(b)

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 2. Morton ordering and its computational strategy

(a) Morton ordering for 16*16 pixel raster, (b) computation of Morton codes from row/column position.

In V3D, raster data is tiled according to Morton order. The tile can be considered as the basic element of raster data. The diagram in the right of Fig.1 shows the conceptual model of raster data. Also, raster image tiles are related with the terrain objects and DTM through a spatial index. For more details of spatial indexing, we refer to section 4.

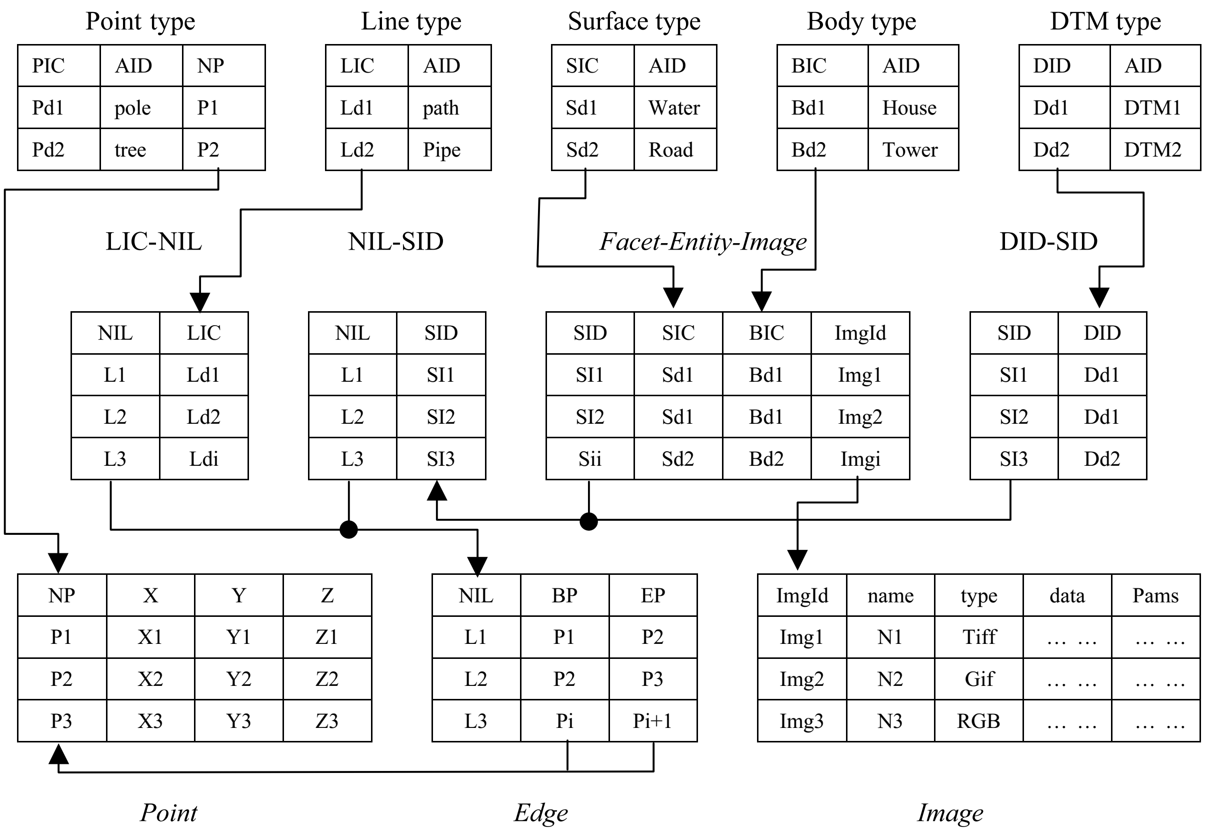
3 IMPLEMENTATION IN A RELATIONAL DATABASE

3.1 Relational model

In a relational database the most common object to be manipulated is the relation table. Other objects such as index, views, sequence, synonyms and data dictionary are usually used for query and data access. "Table" is the basic storage structure, which is a two-dimensional matrix consisting of columns and rows of data elements. Each row in a table contains the information needed to describe one instance of the entity; each column represents an attribute of the entity. The data model shown in Figure 1 is a relational model, which can be implemented by relational data base technology. Fig. 3 shows the relational model of objects in the V3D data structure.

Each object type is defined as a table, shown as the upper row. A point type table includes three terms. The Point Identification Code (PIC) is an identification code for a point type object. The Attribute Identification (AID) is coded to relate an attribute table. Different types of objects may have different attribute tables. For example, the attribute tables of "tree" may have different thematic definitions than "pole". The Name of Point (NP) is the identification of a geometric point, which is used to relate it with a distinct element in a point geometric element table. The *Point* table is the most basic geometrical element table, which defines the coordinate position of the geometrical points.

The Line type table has similar content as the Point type table. The difference is that a line type object is identified by the Line Identification Code (LIC) which is not directly linked with the geometric element table *Edge*, but linked with an intermediate relational table LIC-NIL and then indexed to the *Edge* table. The *Edge* table defines the geometrical element *Edge*, in which each edge (NIL) is described by the beginning point (BP) and the end point (EP). The Surface type table and Body type table have similar terms as the Line type table. For each type of object a distinct identification code (SIC or BIC) is labelled. Both SIC and BIC are linked with a merging geometrical element table, *Facet-Entity-Image*, in which the topological relationships between *Facet* and Surface, *Facet* and *Entity*, *Facet* and *Image* are defined. *Facet-Entity-Image* table has two links: one is related to the *Image* table; the other is related to the NIL-SID table. *Image* table is a basic table, which describes all attributes of images, such as the image name, format, pixels, camera parameters, orientation parameters etc. The NIL-SID table is another intermediate table, which defines the corresponding relationships between *Facet* and *Edge*. Its NIL column is related to the *Edge* table. The DTM is treated as a special class, which is related to the NIL-SID table through an intermediate table DID-SID.



Point type

PIC	AID	NP
Pd1	pole	P1
Pd2	tree	P2

Line type

LIC	AID
Ld1	path
Ld2	Pipe

Surface type

SIC	AID
Sd1	Water
Sd2	Road

Body type

BIC	AID
Bd1	House
Bd2	Tower

DTM type

DID	AID
Dd1	DTM1
Dd2	DTM2

LIC-NIL

NIL	LIC
L1	Ld1
L2	Ld2
L3	Ldi

NIL-SID

NIL	SID
L1	SI1
L2	SI2
L3	SI3

Facet-Entity-Image

SID	SIC	BIC	ImgId
SI1	Sd1	Bd1	Img1
SI2	Sd1	Bd1	Img2
Sii	Sd2	Bd2	Imgi

DID-SID

SID	DID
SI1	Dd1
SI2	Dd1
SI3	Dd2

Point

NP	X	Y	Z
P1	X1	Y1	Z1
P2	X2	Y2	Z2
P3	X3	Y3	Z3

Edge

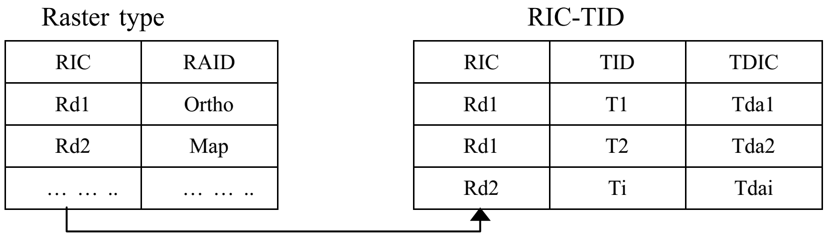
NIL	BP	EP
L1	P1	P2
L2	P2	P3
L3	Pi	Pi+1

Image

ImgId	name	type	data	Pams
Img1	N1	Tiff	… …	… …
Img2	N2	Gif	… …	… …
Img3	N3	RGB	… …	… …

Figure 3. The relational model of the V3D data structure

The raster model is also implemented in a relational DB. Raster model is defined in a two term relational table shown in Fig.4, in which Raster Identification Code (RIC) is the raster identification code. The raster attribute identification defines the raster type related the RIC, and is pointed with correspondent type description table (RAID). For example, type description table "Orthoimage" is an exact relational table, in which the general description of orthoimage is defined, such as, scale, resolution, location, format, colour type, etc. At the same time, RIC is related with RIC-TID table. The relational table RIC-TID has three items. The Tile Identification (TID) is the unique identification of tiles, and the Tile Data Identification Code (TDIC) is the identification of tile data, which serves as the key for access to the tile data.



Raster type

RIC	RAID
Rd1	Ortho
Rd2	Map
… … ..	… … ..

RIC-TID

RIC	TID	TDIC
Rd1	T1	Tda1
Rd1	T2	Tda2
Rd2	Ti	Tdai

Figure 4. The relational model of V3D raster structure

Based on the relational structure shown on the diagram in Fig.3, the query of a geometrical description of a distinct object type is easily realised. For example, the query "Select the geometrical description of an object with the identification code BIC = 202", will first index all Facet identification in the *Facet-Entity-Image* table by its BIC, and then get all edge name identification (NIL), finally index the position information of structure points with the help of *Edge* table and *Point* table.

The queries of topological relationships are divided into two types: relationships between the geometrical elements of an object and those between objects themselves. The relationships between the geometrical elements are implicitly defined in the above data structure. Though the internal topology is not directly supplied, users can flexibly deduce the relationships, such as joint, adjacency, left or right, etc. The queries of topological relationships between objects are not considered in the above data structure because they are application-dependent.

Moreover, the query between raster data and spatial objects are available based on the following spatial index mechanism.

3.2 Spatial index

A spatial index is the ordered data structure based on the position and distribution of the spatial objects, which is implemented by using the object identification.

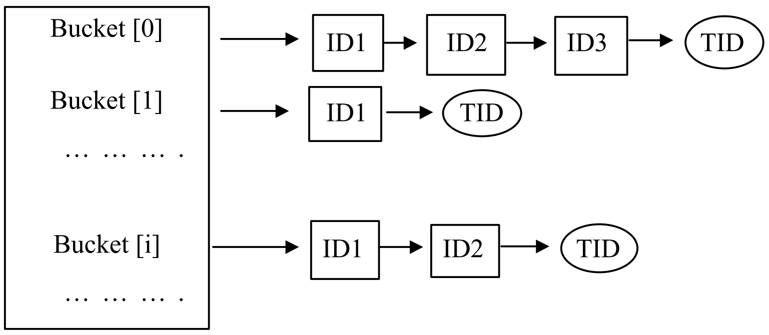


Figure 5. The index data structure

Assume that an exact urban area is partitioned in $n \times n$ blocks, expressed as *block[i, j],* $(0 \le i, j \le (n-1) \times (n-1))$ and data of every block is recorded in a correspondent storage bucket, the *k*th storage bucket is expressed as *Bucket[k]* with the row and column *(i, j)*. The bucket and the block have the following correspondence relation:

$$Bucket[M_d] \Leftrightarrow Block[i, j]$$

Where M_d is Morton code that is computed as indicated in Fig.2.

Spatial indexing is down with spatial index files, which defines the covering of spatial area of geometry model, which are included in a certain space. Considering one or more than one spatial objects go through or beside in a block and the spatial are labelled an exact identification, a variable link pointer is employed to relate the storage bucket with the identification of those objects. If no spatial object is included in the block, the storage bucket is NULL. In the case of raster (orthoimage or map) available in the area, the corresponding raster tile identification is pointed, which follows the variable link pointer. Fig.5 shows the index data structure.

Since Morton codes inherit the 2-D position information, it is directly used as the index of the bucket array to relate spatial position and index table. In GIS, spatial objects are always related with spatial distribution. In our system, the 2-D spatial index is realised to maintain a $n \times n$ bucket table, in which a variable link table is linked with every bucket.

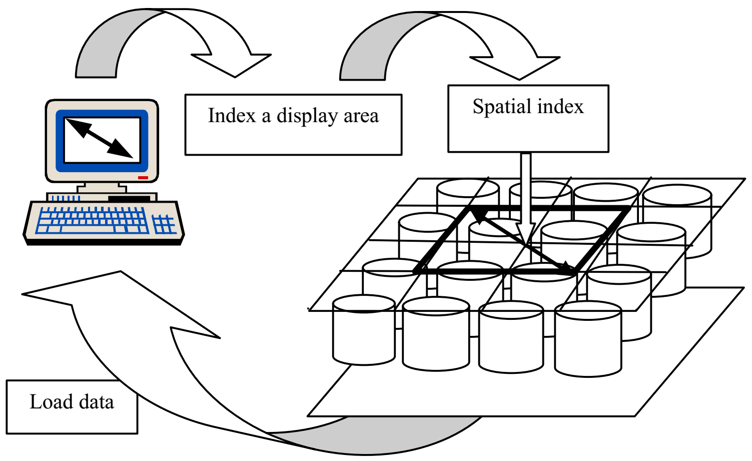


Figure 6. The procedure of spatial querying in a rectangular area

Based on the above index structure, it is easy to query, insert or delete a spatial object. Moreover, since raster data (such as orthoimages or maps) are tiled and coded based on Morton order. It makes it possible to index and relate raster data with vector objects, such as orthoimages with DTM. Fig. 6 shows the procedure of querying the spatial data in a rectangular area.

4 PROTOTYPE SYSTEM

Based on the above data model and structure, a special information system, CC-SIS, has been successfully developed and implemented on a workstation (Sun SPARC) under X-Windows, OSF/Motif, OpenGL as well as ORACLE database. Although its main purpose is scientific investigations, it can also be used in applications like photorealistc representation with possibilities for navigation through the 3-D city model, creation, storage, analysis and query of a city object. In combination with our topology generator CC-Modeler, it builds up a system for geometrical information generation, storage and manipulation.

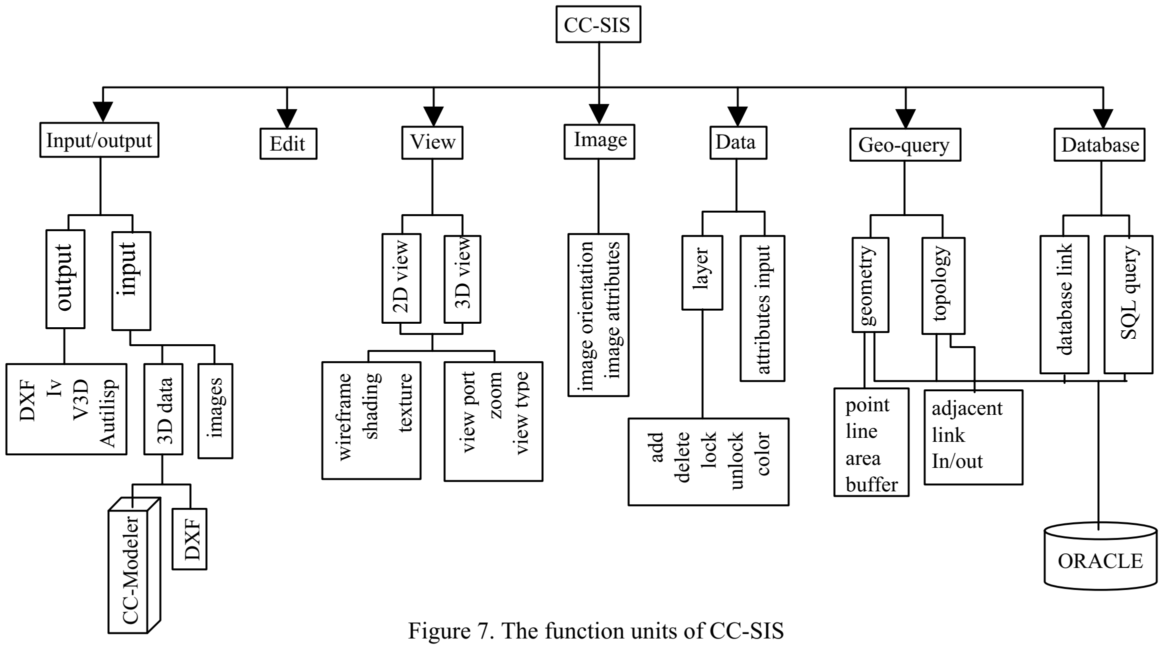


Figure 7. The function units of CC-SIS

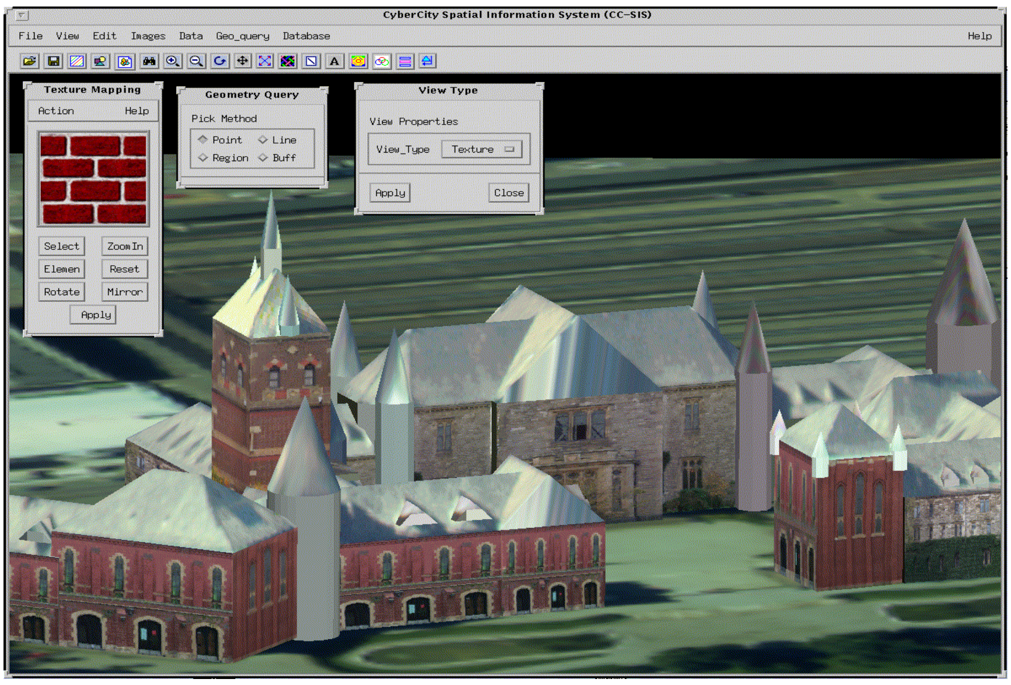
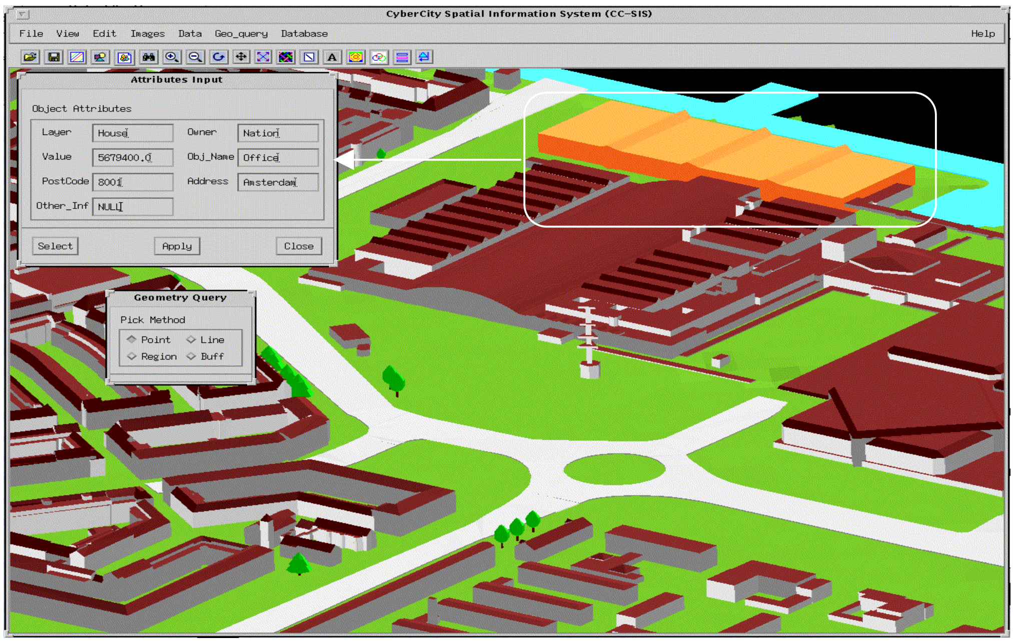


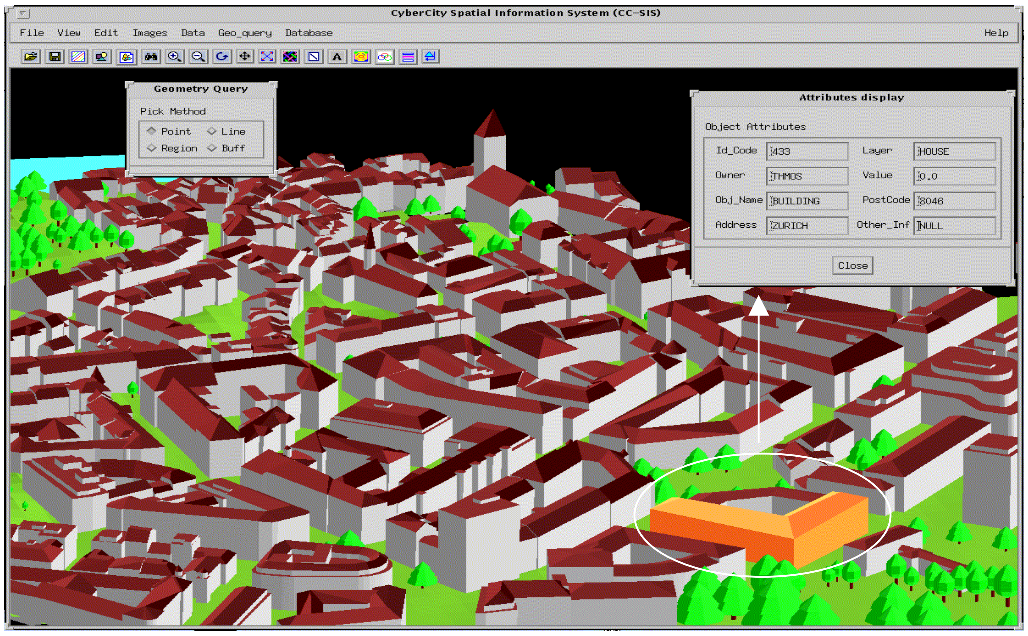
Figure 8. Artificial wall texture produced with the Image function

There are currently seven function units in CC-SIS shown in the Fig. 7. CC-SIS can directly read the data file generated by CC-Modeler and output results in the format DXF, Autolisp, Inventor and V3D. The Edit function is used for graphic editing, which is to be developed in the future. The View function supplies the tools for 2-D or 3-D viewing, such as dynamically selecting a view port, zooming, etc. Further, three types of rendering are available, wireframe, shading and texture mapping. The Image function supplies the tools for interior orientation of the images in order to map natural texture from images. Also, artificial texture can be mapped. An example of this function is shown in Fig. 8.



Layer	House	Owner	Nation
Value	5679400.0	Obj_Name	Office
PostCode	8001	Address	Amsterdam
Other_Inf	NULL		

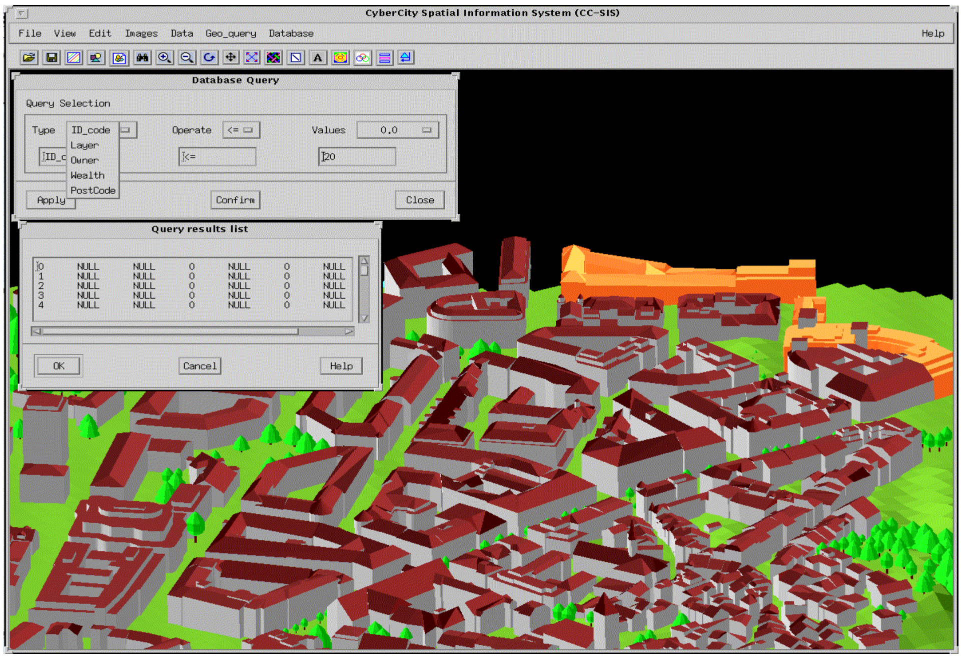
Figure 9. An example of attribute input



Id_Code	433	Layer	HOUSE
Owner	THMOS	Value	0.0
Obj_Name	BUILDING	PostCode	8046
Address	ZURICH	Other_Inf	NULL

Figure 10. The geometrical query of CC-SIS

The manipulations of data are supplied by the Data function. It includes two sub-modules: one is used for the operation on layers (objects are defined as different layers in CC-SIS, such as building, DTM, waterway, pathway, tree, etc.); the other is to input the attributes for the selected object. An example of attribute input is shown in Fig.9. The Geo-query function includes two tools: geometry query and topology query. The former is used to query the separated object by the point, line or entity selection; the latter is employed to query topological relationships between different objects. Figure 10 shows the geometric query of CC-SIS. The user can mark an object (e.g. building) with a cursor. This triggers and displays the corresponding attributes and geometrical/topological information. The operations on a database are defined in the Database function, including database link and SQL-query. SQL-query is a sub-menu, in which standard SQL queries are supplied. An example of SQL query is shown in Fig. 11.



Type: ID_code; Operate: <=; Values: 0.0; ID_c; <=; 20

```
0      NULL   NULL   0   NULL   0   NULL
1      NULL   NULL   0   NULL   0   NULL
2      NULL   NULL   0   NULL   0   NULL
3      NULL   NULL   0   NULL   0   NULL
4      NULL   NULL   0   NULL   0   NULL
```

Figure 11. An example of SQL query in CC-SIS

5 CONCLUSIONS

Given an efficient method for 3-D data acquisition, the generation of a powerful 3-D spatial information system becomes even more mandatory. Our prototype system CC-SIS (CyberCity Spatial Information System) has proven to represent a suitable concept that is worth developing further.

Based on our proprietary V3D vector data structure, a relational database has been created. The data to be operated on can be logically separated into vector data, images and thematic information. In this paper the focus is on the geometrical part of the database (vector data and images). Our pilot applications show that V3D is a suitable structure for the representation of 3-D objects, images and thematic data. It is possible to answer most of the questions about topology, position and shape of objects by means of geometric or SQL queries. CC-SIS has been proven to be used for the following applications:

- Photorealistic representation with possibilities for navigation through the 3-D city model
- Abilities to create, store, design, analyse and query city objects

REFERENCES

Breunig, M., 1996. Integration of spatial information for geo-information systems. Springer Corp., Berlin.

Fritsch, D., Pfannenstein, A., 1992. Integration of DTM data structures into GIS data models. In: Int. Archives of Photogrammetry and Remote Sensing, Washington, Vol. XXIX, B3, pp. 497-503.

Gruen, A., Wang, X., 1998. CC-Modeler: A topology generator for 3-D city models. ISPRS Journal of Photogrammetry & Remote Sensing, Vol.53, No.5, October, pp.286-295.

Gruen, A., Baltsavias, E., Henricsson, O., (eds.), 1997. Automated extraction of man-made objects from aerial and space images(II). Proceedings of the Monte Verita Workshop, May 1997, Birkhauser Verlag, Basel.

Molenaar, M., 1992. A topology for 3D vector maps. ITC Journal, No. 1, pp. 25-33.

Rikkers, R., Molenaar, M., 1994. A query oriented implementation of a topologic data structure for 3-dimensional vector maps. International Journal of Geographical Information Systems, Vol. 8, No. 3, pp. 243-260.