Nat. Hazards Earth Syst. Sci., 16, 123–133, 2016
www.nat-hazards-earth-syst-sci.net/16/123/2016/
doi:10.5194/nhess-16-123-2016

Natural Hazards and Earth System Sciences
Open Access
EGU

Effective 3-D surface modeling for geographic information systems

K. Yüksek[1], **M. Alparslan**[2], **and E. Mendi**[3]

[1]Turkish Aviation Academy, Turkish Airlines, Istanbul, Turkey
[2]Computer Engineering Department, Istanbul Kultur University, Istanbul, Turkey
[3]Computer Engineering Department, KTO Karatay University, Konya, Turkey

Correspondence to: E. Mendi (esmendi@ualr.edu)

Received: 18 August 2013 – Published in Nat. Hazards Earth Syst. Sci. Discuss.: 5 November 2013
Revised: 9 December 2014 – Accepted: 12 February 2015 – Published: 18 January 2016

Abstract. In this work, we propose a dynamic, flexible and interactive urban digital terrain platform with spatial data and query processing capabilities of geographic information systems, multimedia database functionality and graphical modeling infrastructure. A new data element, called Geo-Node, which stores image, spatial data and 3-D CAD objects is developed using an efficient data structure. The system effectively handles data transfer of Geo-Nodes between main memory and secondary storage with an optimized directional replacement policy (DRP) based buffer management scheme. Polyhedron structures are used in digital surface modeling and smoothing process is performed by interpolation. The experimental results show that our framework achieves high performance and works effectively with urban scenes independent from the amount of spatial data and image size. The proposed platform may contribute to the development of various applications such as Web GIS systems based on 3-D graphics standards (e.g., X3-D and VRML) and services which integrate multi-dimensional spatial information and satellite/aerial imagery.

1 Introduction

Due to the increasing importance of visualization, GIS applications are getting widespread use in different forms of development. Initially, environmental planning, natural resource management, public work services, education, research, military, urban planning, land management and telecommunications were common application areas in 2-D solutions. During the past decade, there has been a broad use and need of 3-D GIS applications. The previous approaches for GIS process 2-D data (Zlatanova et al., 2002). Nowadays, 3-D spatial data hiding frameworks have been proposed over database management systems (DBMSs). Most of those applications have been developed on already existing systems. Since GIS-based applications require the processing of a huge amount of data, these systems have heavy demands on processing and storage resources. Consequently, modeling and resource management have become important aspects of an effective GIS application system. The use of GIS in municipal services such as infrastructure and lighting based on city modeling and the related layering approach has increased dramatically (Glander and Döllner, 2009).

Given the growing use of Light Detection and Ranging (LIDAR) technology, the 3-D city modeling and analysis processing have opened a new era. In recent years, the research on methods using LIDAR technologies has exponentially increased (Tao and Hu, 2001). The main purpose of such approaches is to correctly identify the terrain and the objects on the terrain. Several techniques for precisely differentiating the terrain and the objects on the terrain have been recently proposed (Yoon and Shan, 2002). Such methods have also contributed to evolving new software and hardware architectures. For instance in Dollner and Hinrichs (2000), Saha et al. (2011), and McKinney and Cai (2002), the objects and terrain are represented as an object-oriented data structure concept and each building is assigned to an object. The properties of the building include roof type, roof polygons, height, roof surface and LIDAR point sequence. Another approach used interpolation for graphics library and smoothing (Zhou et al., 2004). Many topological models for 3-D spatial objects have been already reported in the literature to describe relationships between spatial objects. As compared to 2-D approaches, such relationships have become more complex in 3-D methods (Zlatanova et al., 2004).

In 3-D GIS modeling, type and size of data have led databases to include special add-ons in order to store and process them. Besides the classic text-based queries, geometry-based queries have become a necessity, leading to the emergence of structures known as spatial databases. This feature has then been adopted by many vendor database systems. In recent years, the structures for the storage as well as query of GIS data have reached a promising level (Kothuri et al., 2007). The increase of precision and quality in satellite and aerial imagery has brought about the tendency to use GIS applications. The need for saving and processing a huge amount of image information in databases has led to not only spatial database extensions but also multimedia database extensions being used in current database systems (Ramakrishnan and Gehrke, 2003). With more widespread use in the field of telemedicine and the media, this feature, in fact, has potential for initiative in the GIS field.

There is a significant trend in the re-modeling of systems in 3-D space. Much work has been done on 3-D processing of terrain and objects on terrain in GIS applications (Arens et al., 2005) based on computer graphics methods (Arens et al., 2005). Delaunay triangulation is widely used in spatial object and terrain representations (Goias and Dutton, 1997). A recent study proposed using polyhedrons as base structures in spatial object representation (Chandra and Govardhan, 2008). On the other hand, recent advances in game engines utilize realistic representations and models such as light, and sound effects (Noh et al., 2006).

Initially, two-tier architectures were used in GIS applications. In these systems, clients operate applications in order to receive services through the network from the database server (Coors, 2003). Storing the data on the server and sections processed on the client are important in the effectiveness of the architecture. Resource needs and the client/server layout in 3-D structures have gained increasing importance. Software complexity and user expectations are crucial in interactive GIS applications. Production of high-quality satellite and aerial images, getting ground elevation and location information accurately with a global positioning system (GPS) have contributed to the development of realistic 3-D GIS applications.

The development of GIS applications that require high-speed data and image transmission does not proceed in parallel with the computer technology. This has led to a search for other means to overcome the bottleneck in terms of computing environments. In other words, a search for new opportunities has emerged to employ high-quality 3-D GIS applications in computers with limited memory resources and processing power. In Guttman (1984), the modeled terrain is entirely processed, but only details of a specific terrain area are presented during representation instead of providing all the details.

With the spread of Internet use, research of web-based GIS solutions has initially started on 2-D (Huang and Lin, 1999). Recently, 2-D GIS research has been replaced by 3-D approaches. The majority of existing systems are based on three-tier or n-tier models. Several platforms have been introduced for the presentation of terrain and spatial objects to users using VRML (Virtual Reality Modeling Language) via web browsers (Ming, 2008; Zhang et al., 2005). However, a major drawback remains the large bandwidth requirement for processing the huge amount of image information.

As discussed above, the current state of research lacks a quick and affordable GIS framework with a satisfactory level of visual quality and high interaction performance for both two-tier and n-tier design models. In this work, a novel digital terrain platform is proposed by creating an architecture with the appropriate components integrated from different disciplines. The system can be used for both client–server applications and web-based solutions. With the coordinate framework and satellite/aerial imagery, a 3-D GIS tool can be generated by the system developed. The system does not require any specific GIS application to create and display the surface. A dynamic surface rendering algorithm allows continuous smooth transitions between surfaces regardless of how many data are processed. 3-D objects produced dynamically or created in 3-D graphics programs are stored in a database and visualized by adding them to merge the surface. With the graphics library used in the presentation layer, more effective interaction can be provided between the user and 3-D objects. Thanks to the multi-layer application architecture, the presentation layer of the system can be improved for web and mobile environments.

The remainder of this paper is organized as follows: Sect. 2 outlines an overview of the generation of spatial and multimedia databases. Section 3 describes the proposed data structure. Section 4 provides an evaluation of the framework. Finally, Sect. 5 presents the conclusions.

2 Spatial and multimedia databases

For quick access and storage of the data on Geo-Nodes, the principal data elements of the system, an efficient data structure model must be employed. To do this, first, data analysis is performed. The data used for creating a surface include a sequential file with surface coordinates in UTM (Universal Transverse Mercator) format, a satellite image of the surface and CAD objects represented on the surface. One challenge is the distribution and continuity of the data in Geo-Nodes. In this work, we applied an Entity-Relational (ER) approach for data modeling and used a relational database management system (RDBMS) structure for generating tables and relationships (Ramakrishnan and Gehrke, 2003). The first step of surface modeling is the transfer of system data to the tables of the corresponding spatial and multimedia database. While the data are stored in tables, methods implemented for business logic to create the surface are stored in package structures of an Oracle database. Packages are implemented using PL/SQL language (Kothuri et al, 2007; Pribyl, 2002). The

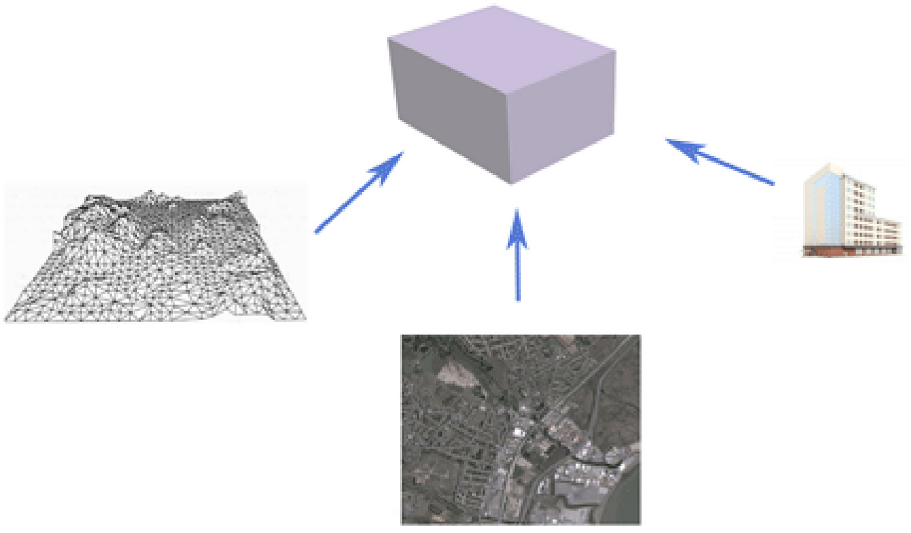

Figure 1. Geo-node structure.

processes of surface modeling described in this section are performed only once. If the surface is reconstructed when any data of the surface are modified, the surface modeling process must be repeated from the beginning. The duration of this process depends on the quantity and size of the data. In other words, the number of Geo-Node data elements storing the system information as discrete data is unique. Figure 1 shows the basic structure of Geo-Nodes. The information stored in Geo-Nodes include text, graphics and spatial data.

The database connectivity physically associated with Geo-Nodes is determined by the data model. The relational scheme that emerged from our data model including tables and their relationships is depicted in Fig. 2.

In addition to the relational model generated from data modeling, the system also has an INIT table containing packages of system parameters. This eliminates the dependence of the system with the application and achieves the business logic part of the application on the server side. The data retrieved from plain text file in UTM format are converted to SDO_GEOMETRY data type and stored in the POINTS table. Figure 3 shows a PL/SQL example of inserting spatial data into the POINTS table.

Graphic elements generating the geometric representation of the surface are stored in the POLYHEDRONS table. Separating the points set in the POINTS table into blocks, those graphic elements are closed geometries with multi-surface and are referred to as polyhedrons. In Fig. 4, a PL/SQL statement is given that generates closed geometries with multi-surface and inserts to the POLYHEDRONS table.

Since spatial querying is a time-consuming process, IDs of points in every polyhedron are stored in a POINT_POLYHEDRON table. Therefore, the relationships of polyhedrons and points are identified by spatial queries only one time, and then stored in the POINT_POLYHEDRON table. In order to perform these queries, an R-tree spatial index must be included in the SDO_GEOMETRY data fields. Figure 5 presents a code block for relationships between polyhedrons and points and inserting them to the POINT_POLYHEDRON table.

Basic triangular elements that will be used in the tessellation process are converted to SDO_GEOMETRY data type and stored in the TRIANGLES table. The Delaunay triangulation algorithm is used for the triangulation process. A PL/SQL statement that inserts triangles to the TRIANGLES table is given in Fig. 6.

The IMAGE table contains satellite images of the surface that will be modeled. The satellite images are in TIF format and can be up to 2 GB in size. The whole image is stored in the IMAGE table and divided into segments corresponding to each polyhedron in the POLYHEDRON table. Each image portion is stored in the IMAGES table. The whole image and segments stored in the IMAGE and IMAGES tables, respectively, are retained using BLOB data type field as binary.

3 Surface modeling

The 3-D surface modeling of a large urban scene requires numerous topographic points. As the surface model expands, the number of points will naturally increase. The method proposed in this study is based on the management of surface data by partitioning into specific data subsets. This section presents the generation of graphics elements, and employment as well as dynamic representation of appropriate data structure.

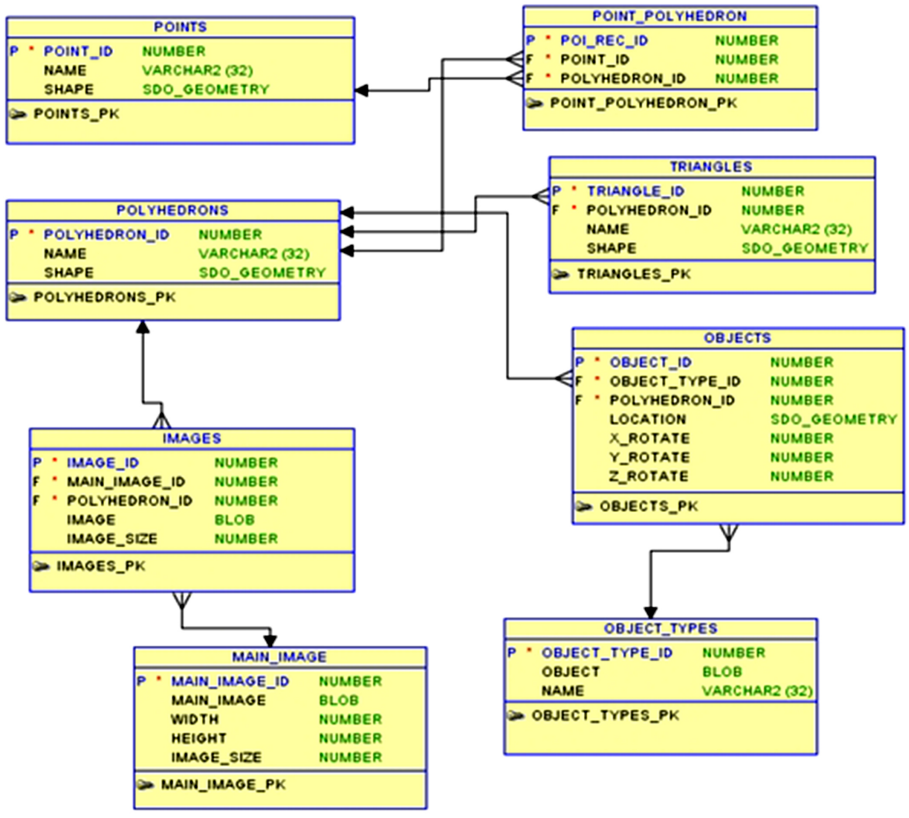

Figure 2. ER diagram of the data stored in tables.

```
INSERT INTO POINTS (SHAPE) VALUES(
SDO_GEOMETRY(3001,NULL,SDO_POINT_TYPE(Px,Py,Pz),NULL,NULL));
```

Figure 3. PL/SQL statement inserting point geometry to the POINTS table.

3.1 Modeling components

Using the UTM coordinate system, Earth coordinates of a surface region are read from a sequential file. The set of points is shifted to the origin of the Cartesian coordinate system by subtracting the x, y, z values of the smallest point from the other points in the data set. 3-D point geometries using SDO_GEOMETRY data type are produced from the records of each row and stored in the POINTS table. Then, the R-tree spatial index is generated using SDO_GEOMETRY data type in the SHAPE field of the table for topological associations. Finally, a point set is obtained in 3-D space (Fig. 7) . Prior to the surface modeling, the set of points is divided into specific groups. In order to partition the point set, the minimum bounding multi-surface geometric solid (polyhedron) is constructed that fully encloses the set of points (Fig. 8). The minimum bounding polyhedron containing the point set in 3-D space is then decomposed into surfaces according to the X_PART and Z_PART variables of the INIT table in the spatial database (Fig. 9). These closed surfaces are saved in the POLYHEDRONS table. The SHAPE field of the table stores the geometries using SDO_GEOMETRY data type. It makes use of the R-tree spatial index which supports topological queries.

Interior points of each polyhedron, in other words intersection points, can be determined via spatial queries using the R-tree spatial index. For two geometries as input variables, a spatial Boolean SDO_ANYINTERACT query returns true when they have an intersection relation. The duration of this process depends on the number of points and polyhedrons. Figure 10 shows a query for the determination of intersection points with a polyhedron. Although R-tree indexing is the most efficient spatial access method, it is not as fast as a standard SQL query on a database table using the B-tree index structure (Ramakrishnan and Gehrke, 2003).

Since the determination of intersection points in a polyhedron via topological query is very time consuming, this will be a drawback in real-time surface representation. One pos-

```
INSERT INTO POLYHEDRONS (SHAPE)VALUES(SDO_GEOMETRY(3008,NULL,NULL,
             SDO_ELEM_INFO_ARRAY(1,1007,3),
        SDO ORDINATE ARRAY(X min,Y min,Z min,X max,Y max,Z max)));
```

Figure 4. PL/SQL statement generating closed geometries with multi-surface and inserting to the POLYHEDRONS table.

```
...
  CURSOR crsPolyhedrons
  IS SELECT * FROM POLYHEDRONS ORDER BY POLYHEDRON_ID ASC;
  rowPolyhedron crsPolyhedrons%ROWTYPE;

  CURSOR crsPoints(polyhedronShape SDO_GEOMETRY)
  IS SELECT P.* FROM POINTS P WHERE
SDO_ANYINTERACT(P.SHAPE, polyhedronShape) LIKE '%TRUE%';
  rowPoint crsPoints%ROWTYPE;
BEGIN
    OPEN crsPolyhedrons;
    LOOP
      FETCH crsPolyhedrons INTO rowPolyhedron;
      EXIT WHEN crsPolyhedrons%NOTFOUND;

      OPEN crsPoints(rowPolyhedron.Shape);
      LOOP
        FETCH crsPoints INTO rowPoint;
        EXIT WHEN crsPoints%NOTFOUND;

        INSERT INTO POINT_POLYHEDRON(POLYHEDRON_ID,POINT_ID)
               VALUES(rowPolyhedron.Polyhedron_Id,rowPoint.POINT_ID);
      END LOOP;
      CLOSE crsPoints;

    END LOOP;
    CLOSE crsPolyhedrons;
END;
```

Figure 5. PL/SQL code block identifying the relationships between polyhedrons and points, then inserting to the POINT_POLYHEDRON table.

sible way of saving computational effort is to store the intersection points in a table, namely POINT_POLYHEDRON. The B-tree index is stored in a POLYHEDRON_ID field of the table. As a result, significant performance differences can be achieved in accessing the intersection points. Figure 11 shows an example of topological query using a B-tree index.

Using the Delaunay triangulation method, a surface can be generated from the points within polyhedrons that are stored in the POLYHEDRONS table. Thus, a triangular network is formed from each polyhedron. Triangles of the network are stored in the TRIANGLES table. All polyhedrons together comprise a triangular surface. Neighbors of the polyhedrons are produced using doubly linked list and the final surface representation is obtained by combining surfaces of polyhedrons with their neighbors.

Problems arise when the surface model is formed by partitioning the polyhedrons into triangular elements. Merging the triangular surfaces of each polyhedron with its neighboring surfaces may result in surface overlap, shared sharp edges of surfaces or surface gap. Figure 12 shows surfaces of polyhedrons in which each of four squares is the projection of a polyhedron. These problems are tackled by introducing artificial points (marked by crosses) on the vertices of the polyhedrons (Fig. 13). Artificial points may not be in the spatial data set forming the surface. Therefore, close attention should be directed toward the surface characteristics while adding artificial points, in order to preserve the original surface shape. The "y" height values of the artificial points added to the polyhedron vertices must be accurately computed to preserve the surface characteristics. Since polyhedrons are stored in the POLYHEDRONS table of the spatial database using SDO_GEOMETRY data type, in the horizontal plane the "x" and "z" values of the points located on polyhedron vertices can be found easily. In order to determine the heights of new vertices, the intersections of each new vertex with the other four polyhedrons are identified. If a new vertex added to the polyhedron lies on the corner of the surface formed by other polyhedrons, then it will be in intersection with only its own geometry. If it lies on the edge of the entire surface, then it will be intersecting two polyhedrons, otherwise it will be in intersection with four polyhedrons. After the characterization of intersection state, the closest point to each new vertex is determined. In calculation of height values of new points that lie on the x–z axis, if any points already exist on the x–z axis, the height of the existing points is used. Otherwise, the height and proximity of the closest points are

```
INSERT INTO TRIANGLES(POLYHEDRON_ID,SHAPE) VALUES (Polyhedron_Id,
 SDO_GEOMETRY(3003, NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
              SDO_ORDINATE_ARRAY(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X1,Y1,Z1));
```

Figure 6. PL/SQL statement inserting a triangle to the TRIANGLES table.

Figure 7. A sample of point set.

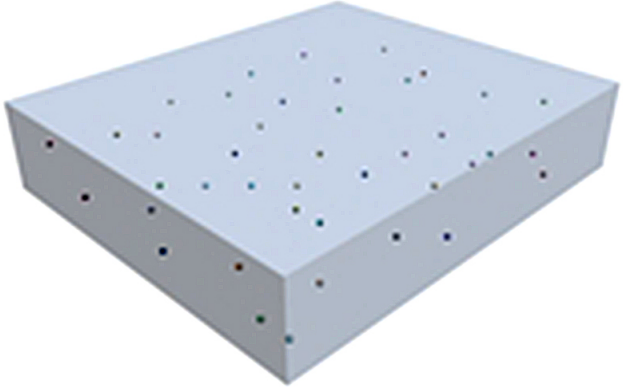

Figure 8. Multi-surface closed geometry containing point set.

taken into account. Thus, the height of an artificial point can be greatly influenced by the height of its closest point. To compute the height of an artificial point, first, the height ratio of the point is calculated as

$$\mathrm{pR}_i = 1 - \left(\mathrm{pD}_i + \mathrm{sD}\right) \quad i = 1, \ldots, 4, \tag{1}$$

where pD_i is the distance of the point i to the artificial point and sD is the sum of distances of all points to the artificial point. Applying Eq. (1) to all other points, the height of the new artificial point pH is computed as

$$\mathrm{pH} = \frac{\left(\left(\mathrm{pR}_1 + \mathrm{pH}_1\right) + \left(\mathrm{pR}_2 + \mathrm{pH}_2\right) + \ \ldots \ + \left(\mathrm{pR}_i + \mathrm{pH}_i\right)\right)}{\left(\mathrm{pR}_1 + \mathrm{pR}_2 + \ \ldots \ + \mathrm{pR}_i\right)} \quad i = 1, \ldots, 4, \tag{2}$$

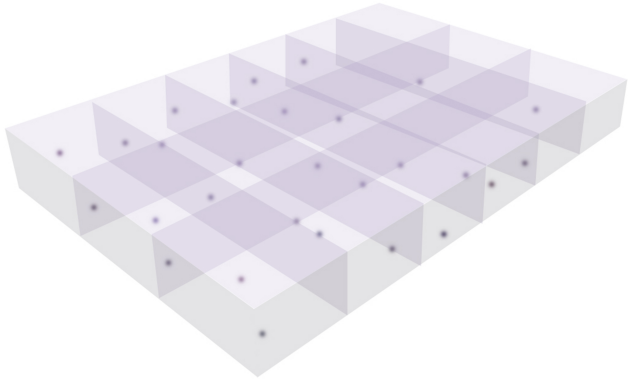

Figure 9. Decomposed geometry containing point set.

where pH_i is the height of the ith closest point to the arbitrary point. Finally, the height of the arbitrary point pH is obtained from the values of heights pH_i and height ratios, pR_i. This computation is applied to all vertices of the polyhedrons stored in the POLYHEDRONS table. Thus, all artificial points introduced in the vertices of the polyhedrons are defined in 3-D space. Defined point geometries are stored in the POINTS table.

Applying Delaunay triangulation with generated points followed by combining surfaces in the polyhedrons, piecewise surfaces (Fig. 14) can be merged into a whole surface model. Although the new points on the polyhedron vertices do not disrupt the surface characteristics, the vertices of new triangles can distort the surface shape. This is the case when two of the points on the polyhedron vertices are two of the points producing the same triangle. If the surface is too smooth, the distortion may not be distinguished. However, if the surface is sloped and composed of a variety of shapes, the base edge of the outermost triangle containing two vertices of polyhedron lies improperly on a straight line from one polyhedron vertex to another. This situation results in the visibility of connection points (junctions), and hence decreases realistic appearance. In order to avoid this, we create new artificial points reflecting surface characteristics between the new points added on the polyhedron vertices at regular intervals. The number of artificial points added along the vertices of polyhedron is the value stored in the EDGE_POINT column of the INIT table of the spatial database. The heights of the artificial points are calculated using Eq. (1). New added points allow a smooth transition that preserves surface fea-

```
SELECT P.* FROM POINTS P WHERE SDO_ANYINTERACT(P.SHAPE, (SELECT R.SHAPE FROM
POLYHEDRONS R WHERE R.POLYHEDRON_ID=10)) = 'TRUE';
```

Figure 10. Query determining intersection points with a polyhedron.

```
SELECT P.* FROM POINTS P INNER JOIN POINT_POLYHEDRON R ON
P.POINT_ID=R.POINT_ID WHERE R.POLYHEDRON_ID=10;
```

Figure 11. A topological query with B-tree index.

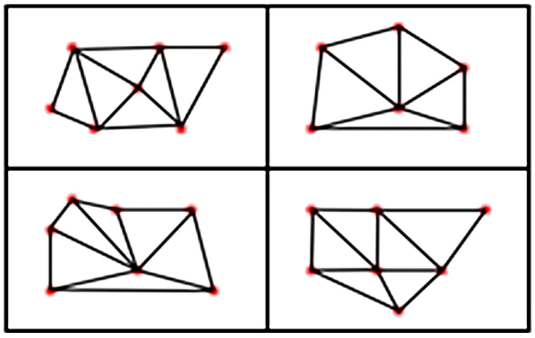

Figure 12. Surfaces of polyhedrons.

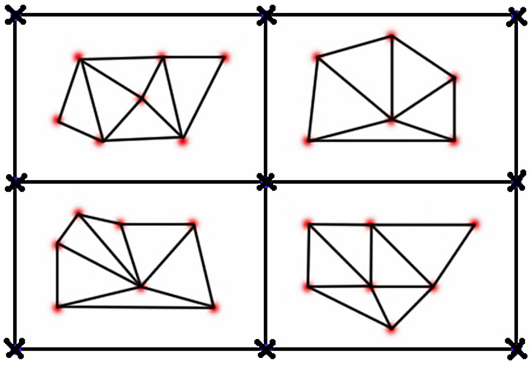

Figure 13. Artificial points, shown by x, added to merge the surface.

tures between inner surfaces. Figure 15 depicts a smooth view of the whole surface model.

The next step after surface modeling is texturing. In this study, satellite imagery is used as a texture overlay to generate terrain. The satellite image is cropped to the values in X_PART and Z_PART fields in width and height, respectively. Each cropped segment is stored in the IMAGE field of the IMAGES multimedia database table in binary format. A unique identity number in the POLYHEDRON_ID field determines which image belongs to which polyhedron. Figure 16 shows the effect of seamless texturing between surfaces.

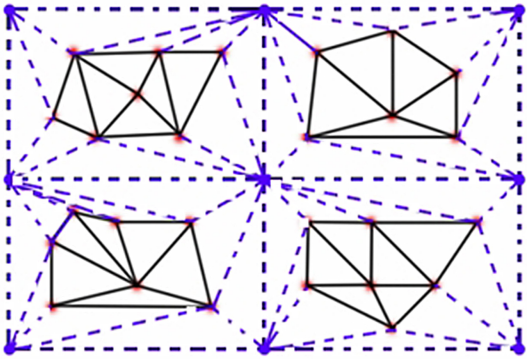

Figure 14. A whole surface model merged with new dotted triangles.

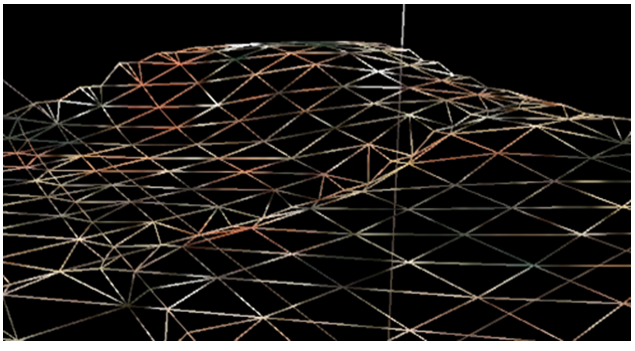

Figure 15. Smooth view of whole surface model.

The proposed surface modeling technique provides a 3-D representation of the topographic ground surface and many types of ground and non-ground objects such as buildings, houses and infrastructure (e.g., roads and bridges including noise barriers, lamp posts and sign boards as well as buried objects such as drums and pipes). The system allows representation of any object on terrain. An object is associated with a 3-D geometry model together with a texture image and integrated to the model with its world coordinates. In order to facilitate the storage and indexing of 3-D objects represented on the surface model, two database tables have been created: (i) OBJECT_TYPES, and (ii) OBJECTS. The OB-

Figure 16. Textured model with continuous smooth transitions (top panel) and without continuous smooth transitions (bottom panel) between surfaces.

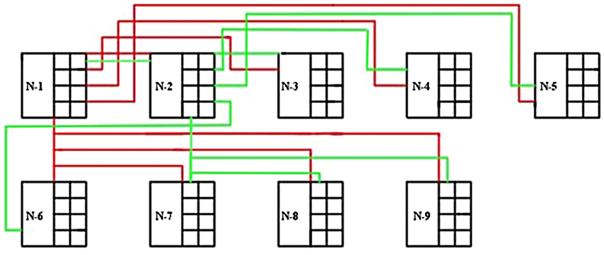

Figure 17. Eight-connected linked list.

JECT_TYPES table is used to incorporate different types of objects with the terrain surface. The OBJECTS table contains location information and terrain conditions for placing an object referenced in the OBJECT_TYPES table. After model construction, the object is inserted into the OBJECT field of the OBJECT_TYPES table in binary format. Then, the object model is scaled to the desired size, rotated to the desired orientation and ultimately translated to the final destination on surface. This makes it possible to use multiple references (instancing). That is, once the object model is abstracted, re-instancing it into several representations could be possible. Once the position of the object model is identified, the polyhedron that bounds the object is determined from the spatial database using topological queries. An R-tree index structure is built during the process of unfolding polyhedrons using topological queries on the LOCATION column of the OBJECTS spatial table. The polyhedron is then assigned a unique identification number in a POLYHEDRON_ID field of the OBJECTS table.

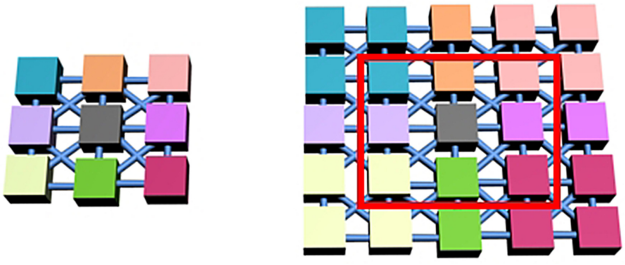

Figure 18. Illustration of eight-connected linked list composed of Geo-Nodes.

3.2 Data structure

A Geo-Node, as a stand-alone element, has no meaning itself. Its relations with other Geo-Nodes must be known. Figure 17 depicts proposed eight-connected double-linked list, in which *N* denotes the objects stored in linked lists.

Each Geo-Node in the eight-connected double-linked list stores geometry entities including triangles that form multi-surface geometry, satellite image segments that drape over the surface and 3-D objects represented on the terrain.

The main use of the proposed dynamic data structure is that it manages the movement of persistent Geo-Node data in secondary storage and buffer during interactive display of the surface model by the user. In this context, the linked list corresponds to a grid structure whereby accessing one element leads to traversing through all elements in the linked list. Figure 18 illustrates the eight-connected linked list structure composed of Geo-Nodes. The right-hand side of Fig. 18 shows all Geo-Nodes allocated in the memory at a specific time, and the red frame indicates the subset of Geo-Nodes available to the user at that time.

The RECTANGLES table of the spatial database consists of TOP, LEFT, RIGHT, BOTTOM, RIGHT_TOP, RIGHT_BOTTOM, LEFT_BOTTOM, LEFT_TOP, LEFT_BOTTOM columns of numeric data with rows. The data contain numeric values defining the 3-D multi-surface geometry stored in the same row with all neighboring geometries in the eight-connected double-linked list. A motion at the client side via any I/O device triggers the buffer manager based on directional replacement policy (DRP). This module provides smooth and continuous retrieval of data by utilizing effective connectivity periods when the user's view is moving. A simple mechanism of the DRP algorithm is given in Fig. 19. *P* denotes the Geo-Node to be fetched using pre-fetching and *X* denotes the Geo-Node to be replaced using the replacement policy in the movement of the user's view.

3.3 Dynamic representation of the surface model

The proposed multi-layer system offers a user-friendly interface that is easy to understand and provides interactive vi-

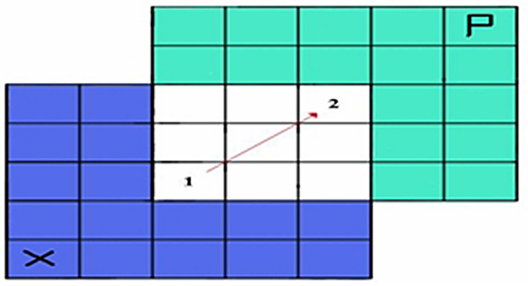

Figure 19. The DRP mechanism.

sualization. The position and orientation of the user's view on the surface are continuously updated. When the current view of the user is within a polyhedron that is on the corner and edge vertices of the surface, new surface segments (plane surfaces) are identified to be visualized. These are the planes in neighboring polyhedrons of the current polyhedron. Polyhedrons together with data of corresponding triangular planes, satellite images and CAD objects in the database are prepared for visualization. The surface of each polyhedron can then be reconstructed from these data. Figure 20 shows the reconstruction of the surface model when reaching the boundary of the display.

The future viewing direction of the user on the surface can be estimated based on the path of the navigation, and hence new surface segments to be visualized can be incorporated into the buffer pool in the memory before reaching a boundary region. Whenever the position of the user within a polyhedron is placed upon another polyhedron, a neighboring polyhedron of the current polyhedron is identified using the eight-connected double-linked list, and corresponding planes and objects of this polyhedron are sent to the buffer. In query processing, when the database needs to access a data item of interest, it first requests the buffer manager. If the desired surface in the polyhedron is already in the buffer, the buffer manager simply returns it. Otherwise, the buffer manager brings the desired polyhedron and corresponding information (triangles, objects and satellite image) from the database to the memory. This process is repeated for any displacement on the surface until the view of the user reaches a boundary. Before displaying an edge surface on the current polyhedron, the surface segment to be rendered is reconstructed. In the dynamic representation mechanism, the buffer manager is first called to fetch the data needed for reconstruction. If the buffer manager has not already cached a requested block, the surface to be visualized is reconstructed reading the corresponding block from the database to the buffer pool. After surface reconstruction, the buffer manager allocates space in the buffer by throwing out objects that are no longer required. Due to the huge amount of data, accessing a sector in the database is much slower than accessing a buffer space in the memory. Therefore, the buffer management module reduces

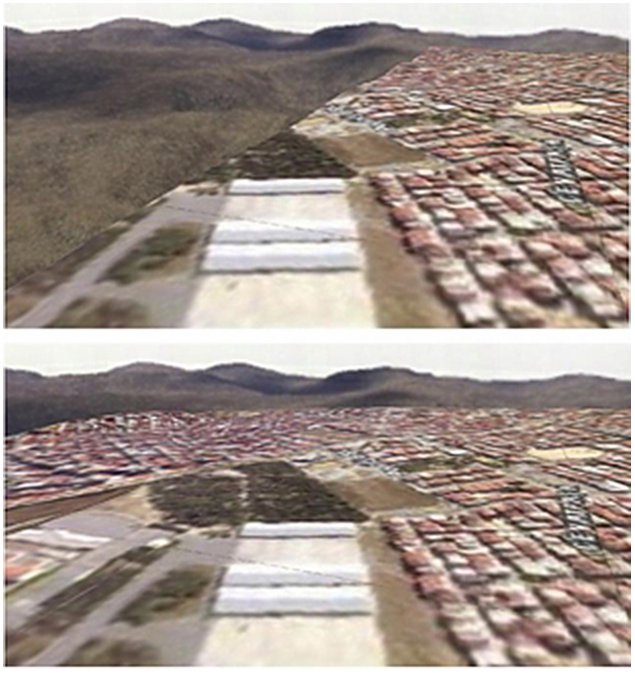

Figure 20. Reconstruction of the model in the direction of the movement upon reaching the boundary of the surface at a specific time.

the number of accesses to the database and I/Os needed, and improves the completion time of queries.

4 Experimental results

The performance of the proposed surface modeling technique is evaluated using frame per second. Frame per second is influenced by several factors such as the number of rendered points, size of the satellite image draping over the surface, the number of rendered polyhedrons at a specific time as well as graphics card and refresh rate. Performance results are depicted in Fig. 21. Frame per second in the fifth column gives the maximum and minimum interval values according to various parameters. The first column (the number of points) represents the number of points to create surface model. These points are processed by dividing the subsets by the number of polyhedrons given in the second column. If the same number of points is used for more polyhedrons, the number of points required for each polyhedron will decrease. Neighborhood in the third column denotes the number of polyhedrons in which a surface is formed during rendering. Thus, the more neighbor relationships that occur, the greater the number of polyhedrons that contain data of rendered surfaces. As a result, the number of points, polyhedrons and neighbor relationships determine the amount of data required for surface modeling at a specific time. The surface is overlaid by a satellite image after the data processing procedure. The fourth column of the table shows the size of

The number of Points	The number of Polyhedron	Neighbourhood	The size of the satellite image	FPS
10.000	25	2	1 MB	75–69
10.000	25	2	170 MB	75–63
10.000	100	2	1 MB	75–72
10.000	100	2	170 MB	75–66
10.000	100	3	1 MB	75–72
10.000	100	3	170 MB	75–64
100.000	25	2	1 MB	75–54
100.000	25	2	170 MB	75–39
100.000	100	2	1 MB	75–65
100.000	100	2	170 MB	75–43
100.000	100	3	1 MB	75–64
100.000	100	3	170 MB	75–61

Figure 21. Performance results.

the satellite images. Higher resolution images mean higher size of satellite image files. Furthermore, the larger the size of a satellite image is, the larger the size of an image segment draping over the surface of a polyhedron. Hence, the large size of these image files will require higher performing hardware – i.e., faster graphics cards, more powerful processors, etc. – in order to render the scene.

We also measured the memory consumption of the proposed surface modeling method. Figure 22 presents memory usage during rendering. When the user moves along from one polyhedron to the other during dynamic navigation over the surface, the terrain data, which is not in the main memory, according to the current position of the user is brought to the buffer space by reading from the spatial database. Since the DRP-based buffer manager loads data from the database to the main memory buffer pool, overall memory consumption increases over time as long as the user moves over the surface, as shown in Fig. 22. When the user's view is at the boundary of the terrain, the surface is reconstructed using the data in the buffer pool, and finally data are released from the buffer. The downward regions of memory requirement in Fig. 22 indicate the reconstruction of the model. In practice, fitting huge spatial data entirely inside the main memory is infeasible. Therefore, such buffer management will provide significant performance savings.

Experiments were conducted on an AMD Athlon 64 X2 4200+ dual-core processor having a speed of 2.20 GHz and 3 GB of main memory under Windows XP 32 bit operating system. The graphics card used was an nVIDIA GeForce 9500 GT with 128 MB of on-board memory and the monitor was set to a refresh rate of 75 Hz.

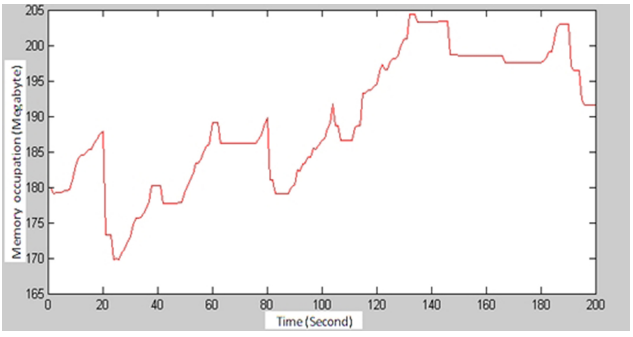

Figure 22. Memory usage (MB) vs. scene time (s).

5 Conclusions and discussions

In this paper, we presented an interactive multi-purpose platform for web-based or client/server-based 3-D GIS applications with superior graphics support. The system utilizes main data elements, called Geo-Nodes, constituted from a flexible data structure and supported by an effective buffer management. The data transfer between secondary storage and main memory is optimized with a "most recently used" strategy. Thus, fast access is provided to the anticipated uses of Geo-Nodes that store spatial data and corresponding images. The capabilities of data and query processing in spatial database and handling multimedia components in the multimedia database are integrated with an efficient data structure. Using an interpolation method, surface anomalies are eliminated. Furthermore, a game engine was used in the rendering pipeline in order to improve the realism of visualization. The experimental results show that visualization of an urban scene is successful with high performance. The proposed platform can offer new capabilities to build 3-D GIS applications in two tier or n-tier (client/server) architectures.

As the proposed system detects each 3-D object defined on the 3-D surface as a layer and visualizes it this way, the area of utilization can be very broad. To give an example, the system can be used for 3-D cadastral applications. There are studies of 3-D cadastral systems available, carried out by integrating 3-D objects, made in applications like CAD, to 2-D cadastral data (Sorensen, 2011; Hassan and Rahman, 2011). 3-D objects and cadastral information, used in the cadastral systems, can be positioned as individual layers on the application database and can be displayed interactively on the 3-D surface. Further, the proposed system is suited to applications of a full 3-D cadaster (Guo et al., 2013). Again, pipeline networks can be defined as a layer to the system, and pipe faults underground and aboveground could be displayed together with the surface (Li et al., 2010). Like pipelines, highway and railway networks can also be added to the system, forming their models as an individual layer.

Edited by: R. Lasaponara
Reviewed by: D. Liu and three anonymous referees

References

Arens, C., Stoter, J., and Oosterom, P.: Modelling 3-D spatial objects in a geo-DBMS using a 3-D primitive, Comput. Geosci., 31, 165–177, 2005.

Chandra, N. S. and Govardhan, A.: Design and Implementation of Polyhedron as a Primitive to Represent 3-D Spatial Object, J. Theor. Appl. Inf. Technol., 4, 212–218, 2008.

Coors, V,: 3-D-GIS in networking environments, Computers, Environ. Urban Syst., 27, 345–357, 2003.

Dollner, J. and Hinrichs, K.: An object-oriented approach for integrating 3-D visualization systems and GIS", Comput. Geosci., 25, 67–76, 2000.

Glander, T. and Döllner, J.: Abstract representations for interactive visualization of virtual 3-D city models, Comput. Environ. Urban Syst., 33, 375–387, 2009.

Goias, N. A. and Dutton, R. W.: Delaunay Triangulation and 3-D adaptive mesh generation, Fin. Elem. Anal. Design, 25, 331–341, 1997.

Guo, R., Li, L., Ying, S., Luo, P., He, B., and Jiang, R.: Developing A 3-D Cadastre for the Administration of Urban Land Use: A Case Study of Shenzhen, China, Comput. Environ. Urban Syst., 40, 46–55, 2013.

Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching, ACM SIGMOD International Conference on Management of Data, 47–57, 1984.

Hassan, M. I. and Rahman, A. A.: Unique Identifier for 3-D Cadastre Objects Registratio, 2nd International Workshop on 3-D Cadastres, 16–18 November 2011, Delft, the Netherlands, 137–148, 2011.

Huang, B. and Lin, H.: GeoVR: a web-based tool for virtual reality presentation from 2D GIS data, Comput. Geosci., 25, 1167–1175, 1999.

Kothuri, R. V., Godfrind, A., and Beinat, E.: Pro Oracle Spatial for Oracle Database 11g, Springer, New York, 2007.

Li, Z., Li, P., Wu, M., and Wang, W.: Application of ArcGIS Pipeline Data Model and GIS in Digital Oil and Gas Pipeline, 18th International Conference on Geoinformatics, 18–20 June 2010, Beijing, China, 1–5, 2010.

McKinney, D. C. and Cai, X.: Linking GIS and water resources management models: an object-oriented method, Environ. Model. Softw., 17, 413–425, 2002.

Ming, W.: A. 3-D Web GIS System Based On VRML And X3-D, Second International Conference on Genetic and Evolutionary Computing, 25–26 September 2008, Jingzhou, Hubei, China, 197–200, 2008.

Noh, S. S., Hong, S. D., and Park, J. W.: Using a game engine technique to produce 3-D Entertainment contents, 16th International Conference on Artificial Reality and Telexistence-Workshops (ICAT'06), 29 November–1 December 2006, Hangzhou, China, 246–251, 2006.

Pribyl, B.: Learning Oracle PL/SQL, O'Reilly, California, 2002.

Ramakrishnan, R. and Gehrke, J.: Database Management Systems, 3rd Edn., McGrawHill, New York, 2003.

Saha, K., Wells, N. A., and Munro-Stasiuk, M.: An object-oriented approach to automated landform mapping: A case study of drumlins, Comput. Geosci., 37, 1324–1336, 2011.

Sorensen, E. M.: 3 Dimensional Property Rights in Denmark: 3-D Property Design and Registration is Working – Visualization not, 2nd International Workshop on 3-D Cadastres, 16–18 November 2011, Delft, the Netherlands, 521–530, 2011.

Tao, V. and Hu, Y.: A review of post-processing algorithms for airbone LiDAR data, ASPRS Annual Conference, 23–27 April 2001, St. Louis, Missouri, USA, 2001.

Yoon, J. S. and Shan, J.: Urban DEM generation from raw airbone LiDAR data, CD-ROM Proceedings of the annual ASPRS (American Society for Photogrammetry and Remote Sensing) Conference, 19–25 April 2002, Washington, D.C., USA, 2002.

Zhang, L., Yang, C., Liu, D., Ren, Y., and Rui, X.: A web-mapping system for real-time visualization of the global terrain, Comput. Geosci., 31, 343–352, 2005.

Zhou, G., Song, C., Simmers, J., and Cheng, P.: Urban 3-D GIS from LiDAR and digital aerial images, Comput. Geosci., 30, 345–353, 2004.

Zlatanova, S., Rahman, A. A., and Pilouk, M.: Trends in 3-D GIS Development, J. Geospat. Eng., 4, 1–10, 2002.

Zlatanovaa, S., Rahmanb, A. A., and Shi, W.: Topological models and frameworks for 3-D spatial objects, Comput. Geosci., 30, 419–428, 2004.