

Extending Geographic Information System From Two-Dimensional To Three-Dimensional Approaches

Jing Lu LO, Mordechai HAKLAY

Department of Geomatic Engineering, University College London
Gower Street, London WC1E 6BT, United Kingdom
Email: jinglu@hotmail.com, m.haklay@ucl.ac.uk

Abstract

Geographic Information Systems (GIS) may have become a common solution for managing urban data, but unfortunately most of the commercial solutions were limited to handle only the data of two dimensions. Our life takes place in three dimensions (3D) and a two dimensions (2D) GIS is no longer sufficient to fulfill the needs of users. In geometrical terms, 2D to be a subset of 3-D and therefore this paper proposes to build a 3D solution that are sitting on top of, and working alongside with existing 2D system and the focus is on the issues of 3D data modeling, compression technique, indexing method and data query algorithms.

1. Introduction

Our human senses evolve and are geared toward perceiving the world in three dimensions (3D). In today's marketplace, sophisticated 2D GIS are available as a standard and it seems more practicable, wherever possible, to extend the 2D system to handle 3D applications rather than building a brand new 3D system from the scratch. Luckily, when considering the data representation in geometrical terms, 2D is a subset of 3D and therefore in principle, it should be possible to extend GIS from 2D to 3D and such an extension brings several challenges, particularly in constructing the 3D data model, storing data, indexing and finally the provision of an interface to visualize data. In this paper the focus is on the issues of storing and indexing 3D data as well as object query. The system that we have used to extend is the database structure of Intergraph GeoMedia (Figure 1). The paper starts with an overview of storing 3D data in the database that was initially designed to store data of 2D. The 3D data model is then defined by using Delaunay Tetrahedral Tessellation (DTT)(Chen and Murai, 1999). This is followed by a discussion of data compression, as even moderately fine tetrahedral meshes consume large storage space. The next section discusses indexing method, particularly the structure of Hybrid Tree (Chakrabarti and Mehrotra, 1999). In order to get the Hybrid Tree structure to work with the DTT model, the respective data is mapped as points into high dimensional feature space. Following the explanation of the indexing method, the algorithms for data searching inside the index tree is then explained. Finally, the results of a trial run are reported as well as suggestions for future work

2. Related Works

The topic of extending GIS to the third dimension (3D GIS) received much attention and of the literature is now extensive. However, most of the proposed systems are characterized by dealing with some specific aspect of 3D and have yet to offer a holistic approach that deals with all aspects of 3D GIS. For example, special attention has been given to 3D visualization that linked to the conventional GIS and this has resulted in the availability of commercial products such as "SiteBuilder3D" (SiteBuilder3D, 2003). This is one of the products developed by Multigen-Paradigm (Multigen-Paradigm, 2003) that provides high-end 3D visualization based on a GIS database, but does not provide analysis capabilities. Other products, such as 3D Analyst of ESRI's ArcGIS (ESRI, 2003), do provide analysis functionality, but its visualization capabilities are limited. Yet another product called GeoMedia Terrain developed by Intergraph (Intergraph, 2003) serves as Digital Terrain Model (DTM) module for the GeoMedia GIS on terrain model generation, terrain analysis and fly-through. Nevertheless it is based on a 2D datasets and should be considered as 2.5D instead of 3D solution. Apart from the traditional GIS vendors, some other companies have recently provided GIS module in their product. For example the Imagine system, originally

developed by ERDAS (ERDAS, 2003) for remote sensing applications has now a GIS module (Imagine VirtualGIS). This module enables 3D analysis to be carried out together with visualization; nonetheless, the analysis is mainly based on visual instead of object.

ID	RIVER_NAME	RIVER_TYPE	Geometry	Geometry_sk
1	Champlain Canal	Canal	Long binary data 1'Yk	
3	Hudson	Perennial - Single Line	Long binary data 1'Yk	
12	Sacramento River Deep Water Ship Channel	Canal	Long binary data 1'V	
13	Sacramento	Perennial - Single Line	Long binary data 1'w62	
21	Pecos	Intermittent - Single Line	Long binary data 1'Xf	
22	Gila	Perennial - Double Line	Long binary data 1'X_i	
24	Columbia	Perennial - Double Line	Long binary data 1'yp	
25	Columbia	Perennial - Single Line	Long binary data 1'yx	
28	Willamette	Perennial - Double Line	Long binary data 1'YkD	
29	Missouri	Perennial - Double Line	Long binary data 1'Wln9	
30	Snake	Perennial - Double Line	Long binary data 1'xy{	
31	Columbia	Perennial - Double Line	Long binary data 1'yy{	
32	Yellowstone	Perennial - Double Line	Long binary data 1'Wln{	

Figure 1: Database structure of GeoMedia for Microsoft Access. There are five columns in total. [ID] is the index column for the entire row with "Number" type, [RIVER_NAME] and [RIVER_TYPE] are the attribute columns with "Text" type, [Geometry] is spatial column with "OLE Object" type, and [Geometry_sk] is index column for spatial data with "Text" type

Whilst these developments are mainly coming from the GIT community, there are also some researches in computer science or engineering that are working on spatial data storage issues and their work is highly relevant to 3D GIS. In the following paragraphs, we review relevant work that deal with data model, storage and indexing.

The data model is at the core of 3D GIS and the most common approach of 3D storage is the use of a planar polygon with a height value as this is, of course, the easiest solution. However, once the goal is to store 3D spatial objects in their real, multi-facet structures, the data model becomes more complex. There are many data models have been proposed such as the DTT, the Simplified Spatial Model (SSM) (Zlatanova and Tempfli, 2000), or the 3D topology model (Pilouk, 1996). The next issue with 3D GIS is the volume of data that is usually very large in size and being updated continuously. The volume of data may incur significant disk Input/Output (I/O) overhead during query and updating operations. In order to reduce the I/O access overhead, some researchers have suggested using specific compression algorithms to remove, wherever possible, the redundant data in a dataset without information loss during the process of compression or decompression. Related discussions on the compression techniques, particularly for tetrahedral meshes, can be referred to Yang *et al.* (2002) and Gumhold *et al.* (1999).

Data indexing is another mechanism that is usually applied to accelerate the process of queries in the database by keeping some extra information. There are many types of indexing methods have been cited such as the R-tree (Guttman, 1984), the K-D-B-tree (Robinson, 1981) or the Z-ordering (Orenstein, 1986). For an extensive review of multidimensional indexing structures see Castelli (2001) and Gaede (1998). In order to further improve the performance of queries, some researchers combined mechanisms common in computer graphics with multidimensional indexing structures such as the LOD-R-tree (Kofler *et al.*, 2000), Reactive Data Structures (Oosterom, 1990) or V-Reactive Tree (Li *et al.*, 2001). In this paper the Hybrid Tree will be used as the primary search index. This mechanism was chosen because it facilitates feature similarity search in high dimensionality feature space and the index structure can be fine-tuned to handle spatial search.

3. Storing Three Dimensions Object

In the existing database structure of GeoMedia, 2D geometry data is stored as Binary Long Object (BLOB) along with the non-spatial data in one single row (Figure 1). This structure can be extended to store 3D geometry data by simply adding in another new BLOB column. Although both data and indexes are stored inside the database, the procedures for creating and updating to the relevant component are done externally using specific tools that are held up together in the procedure library for the sake of versions management. This simple extension of the existing structure is used here as it extends the capabilities of GeoMedia, without fundamentally changing its underlying data storage. While the underlying database structure was modified slightly to accommodate 3D data, the data model required specific development. In the next section, we provide a description of the data model.

4. Data Model

Although 2D data is a subset of 3D data, the content of 3D data model is far more complicated than the existing 2D model as shown in Figure 2. In the context of GIS, geometry is the most essential property of a data model and 2D system usually stores planar geometry of each feature, while attributes are stored separate fields but within the same record (Figure 3a). However, it is not enough to just extrude the planar geometry using a height value as 3D data model requires the ability to store solid geometry as well as attributes that are associated to each face of the geometry such as the texture entity (Figure 3b). Apart from the geometry property, this paper considers adding in the colour description as the second property under the data model.

Following the required properties, the data model which is used here is based on the four basic elements of geographic data: point, line, shape and volumetric object. It starts with point defines a single position on the space and two points define a line, then three adjoining lines define the simplest shape, which is a triangle. Finally four enclosing triangular surfaces define the simplest volumetric object, a tetrahedron. Theoretically any kind of shape can be formed using a combination of triangles; meanwhile, any kind of volumetric object can be formed using a combination of tetrahedrons. The term tetrahedral mesh is used to describe a complex structure that contains the four basic elements.

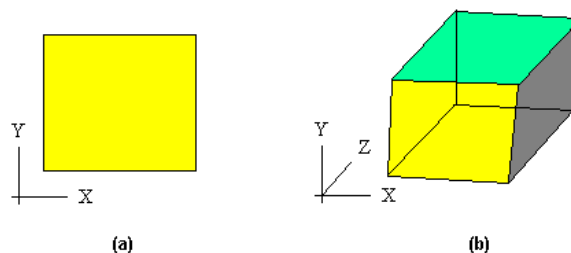


Figure 2: (a) 2D data model with planar geometry and single colour description (b) 3D data model with solid geometry and multiple colour description. 2D is a subset of 3D

For the creation of tetrahedral mesh, we have used an existing 3D tetrahedral mesh generator named TetGen (2002). If the data is either a point, line or surface, it can be paired up with the related texture value in one-to-one relationship. The only exceptional is tetrahedron, which is actually defined by four adjoining triangles where each triangle can have a different value of texture property. Thus, texture entity is attached to each face of the geometry and this kind of relationship has been considered when attempting to define the structure of data model. More explanations regarding to data model structure are given in section 5. Finally, we need to consider the properties of existing 2D data model that cannot be simply left out, such as map projection and co-ordinate transformations. Theoretically, 3D geometry shares at least one face with the 2D geometry (Figure 3); the sharing face provides an opportunity for both sides to supply each other with particular information that might be lacking at either side.

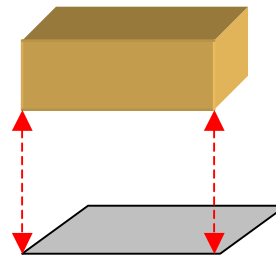
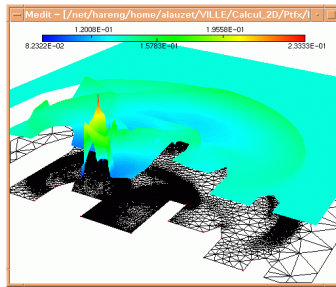


Figure 3: The relationship between 2D and 3D data model, courtesy of Medit (2003). To place a 3D model on the Cartesian coordinate system, it can be referred to the existing 2D model wherever the related information is found

5. Compression Algorithm

Whilst most compression techniques assume that the data must be fully decompressed when it is used, we propose a heuristic technique that allows partial decompression on the compressed dataset. This technique is especially useful for 3D GIS because only tetrahedrons that are located near to the query points will be extracted for next level testing. Compression technique can be classified into two categories of lossless and lossy. A Lossless algorithm prevents any loss of information during the compression/decompression process, whilst lossy algorithm does not guaranty that.

Lossless algorithms for tetrahedral compression can be found in Yang *et al.* (2000) and Gumhold *et al.* (1999) and similar algorithm is used here. The compression algorithm is using an important property of DTT: any of the tetrahedrons is sharing at least one of its faces with adjacent tetrahedrons; thus, by rearranging the vertices orders it is possible to have a compact storage model, by storing the two faces that are sharing same vertices only.

Table 1: Tetrahedral meshes generated by TetGen on cube geometry with eight vertices

Tetrahedron No.	Vertex 1	Vertex 2	Vertex 3	Vertex 4
1 (G2)	2	3	5	1
2 (G2)	4	5	3	1
3 (G1)	4	7	5	8
4 (G1)	3	6	2	5
5 (G1)	5	3	6	7
6 (G1)	4	7	3	5

Table 2: Tetrahedral meshes generated by TetGen on pentagon geometry with ten vertices

Tetrahedron No.	Vertex 1	Vertex 2	Vertex 3	Vertex 4
1 (G3)	7	9	8	4
2 (G1)	1	4	6	2
3 (G2)	4	7	2	3
4 (G1)	5	6	1	4
5 (G2)	7	4	2	6
6 (G2)	6	9	7	4
7 (G3)	4	8	7	3
8 (G1)	5	9	6	4
9 (G1)	9	5	6	10

Table 1 contains 24 entries (6 tetrahedrons x 4 vertices) and those entries that can be grouped (G), are re-arranged into two subsets {G1: 8, 4, 7, 5, 3, 6, 2} and {G2: 2, 3, 5, 1, 4}. Both subsets are now containing 12 entries in total after compression. Upon decompression, the tetrahedrons (T) can be recreated using a set of four adjacent-vertices. For example, {T1: 8, 4, 7, 5}, {T2: 4, 7, 5, 3}, {T3: 7, 5, 3, 6}, {T4: 5, 3, 6, 2}, {T5: 2, 3, 5, 1}, and {T6: 3, 5, 1, 4}. The arrangement of faces (F) become {F1: 8, 4, 7}, {F2: 8, 4, 5}, {F3: 8, 7, 5}, {F4: 4, 7, 5}, ..., {F12: 3, 6, 2}. The similar process is repeated for result that is shown in Table 2. Three subsets have then been defined {G1: 10, 9, 5, 6, 4, 1, 2}, {G2: 9, 6, 4, 7, 2, 3} and {G3: 3, 4, 7, 8, 9}. There are 18 entries in total - half of the 36 entries which are listed in Table 2.

As was stated above, the texture data (C) need to be included in the model structure, thus another subset can be defined as following: {C: [default texture value], [face no.], [new texture value], [face no.], ...}. To preserve storage space, a default texture value is assigned for every face and this value is used unless a texture value is declared for a particular face. For example above the face F2, can be coloured with colour code "000256". If we assume that colour code "000000" is the default, we can decode it as {G1: 8, 4, 7, 5, 3, 6, 2} {C: 000000, 2, 000256}.

6. Indexing Method

Indexing methods are used to facilitate queries, especially when the dataset is large. K-D-B-tree (Robinson, 1981), R-tree (Guttman, 1984) and grid files (Nievergelt *et al.*, 1984) are among the indexing methods that were first proposed to handle spatial data and they indeed work well with the conventional GIS; nevertheless they are not suitable to be applied for high dimensional features, as they are geared towards 2D data. For geographic data, three or four dimensionality of indexing method should be sufficient. However, by adding more dimensions, we can query spatial and non-spatial attributes using the same index. For example, the query "select all the buildings that are inside specific zone" can be extended to "select all the buildings that are inside specific zone and higher than 10 meters from the ground, and painted light yellow colour, and using bricks as building material". This query combines the search possibility ranging from geographic positioning to feature similarity such as colour and construction material.

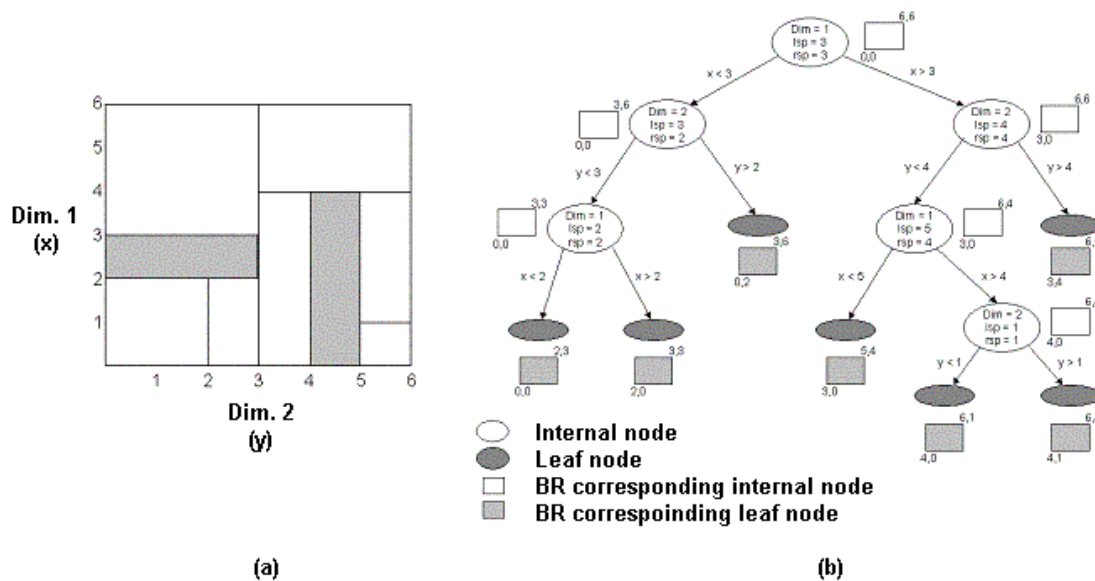


Figure 4: (a) A 2D representation of Space partitioning within Hybrid Tree index node. Partitioning may not always be mutually disjoint as requested in K-D-B-tree (b) Corresponding representation. Internal nodes of KD-trees maintain 2 split positions (left split line and right split line) instead of one to represent overlapping splits

Hybrid Tree (Chakrabarti and Mehrotra, 1999) is a multidimensional data structure for indexing high dimensional feature space, and its main advantage is in performing feature based similarity search. The technique works by mapping data items as points in a high dimensional space, which is then indexed using the mechanism of Hybrid Tree. The structure of Hybrid Tree is similar to K-D-B-tree but it allows overlap between index nodes to increase space utilization (Figure 4a), at the same time, the tree operation is similar to R-trees, where indexed subspaces are treated as bounding rectangle (Figure 4b).

The Hybrid Tree is designed for feature based similarity search and as it is using point-based structure, it may not be suitable to index spatial data, which is normally consisting of lines and polygons. For example in Figure 4a, if a 2D feature that is represented by rectangle from (1, 1) to (5, 5) is inserted to the index space which has been already partitioned we might end-up with four new vertices (1, 1), (5, 1), (5, 5) and (1, 5) that are distributed into four separate nodes. Due to the separation, it probably will cause problems for some spatial queries such as overlap. However, due to the use of DTT, the point-based index structure of the Hybrid Tree is adequate, because inside DTT, all tetrahedrons are mutually disjoint and therefore it is possible to collapse its representation to a point, as we explain in the following sections.

6.1 Feature Mapping As Point

The method to map the feature as point in the Hybrid Tree space is based on analysing the minimum and maximum value on each of its dimension. For example, a rectangle can be described by its minimum and maximum co-ordinates (X1, Y1) – (X2, Y2). Therefore, the feature is ranging from X1 to X2 on the Dimension X and Y1 to Y2 on the Dimension Y (Figure 5). The (x', y') is the mapped coordinate respectively for Dimension X and Y, whereas $y' < x'$, meanwhile $z1'$ and $z2'$ are added to prevent these two sets of data to mixed up in Hybrid Tree space, as $z1' \neq z2'$. More explanation on the proposed mechanism is provided in Table 3.

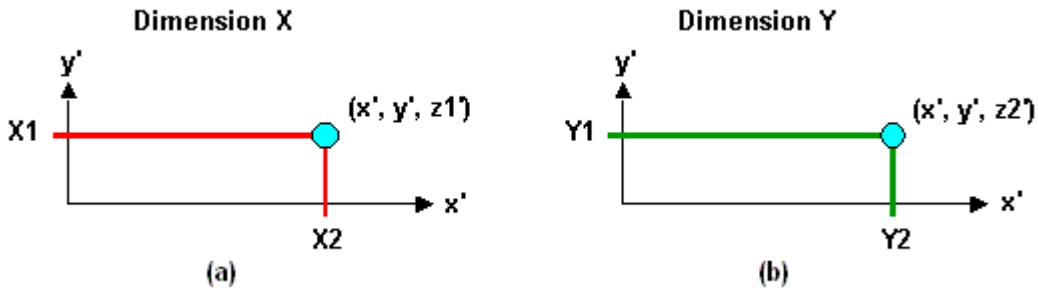



Figure 5: Feature mapped as point (a) on Dimension X, and (b) Dimension Y. False coordinate of the projected point on Hybrid Tree space is given $(x', y', z1')$ for Dimension X and $(x', y', z2')$ for Dimension Y. If it is a point feature, then $x' = y'$, otherwise $x' > y'$. The (x', y') on Dimension X could be same as (x', y') on Dimension Y but $z1'$ definitely not equal to $z2'$. Similar mechanism is applied to other existence dimensions

Table 3: Sample results of feature mapping in Hybrid Tree space as point

Element		Feature			Mapped Point		
		Dim.	Min.	Max.	x'	y'	z'
Shape (Total 4 vertices)		X	1000	5000	5000	1000	1
		Y	4000	5000	5000	4000	2
		Z	0	10	10	0	3
Line (Total 5)		X	3000	8000	8000	3000	1
		Y	2000	6000	6000	2000	2

vertices)		Z	0	0	0	0	3
Point (Total vertex) 1		X	1000	1000	1000	1000	1
		Y	3000	3000	3000	3000	2
		Z	0	0	0	0	3

6.2 The Hybrid Tree index

Points for Dimension 1 will be represented as a set of false coordinate $\{(x1', y1'), (x2', y2'), \dots, (xn', yn')\}$. One of the problems with an indexing method that is based on such false co-ordinates is that mapped points for Dimension 2 could be having the similar set of false coordinate. For example given a rectangle (1, 1) – (5, 5), the mapped coordinate for both dimension 1 and 2 would be (1, 5) on the Hybrid Tree space. In order to distinguish them a third value is assigned to each dimension. Thus, dimension X is assigned a third value (“1”) and dimension Y another (“2”), and therefore the false coordinate for the dimension X is (1, 5, 1) and (1, 5, 2) for Dimension Y. similar approach is being used to differentiate the other dimensions (Figure 6). Apart from the three geographical dimensions, additional information is added into the Hybrid Tree space in order to maintain the connectivity between dimensions. For example if the mapped point ① is selected in Dimension 1, the respective points ① in other dimensions should be selected. In order to identify point ① in all the dimensions, a unique ID is assigned to all tuples that relate to the same object. This is similar to B-tree search operation that scans more than one index and hash-join on the “ROW_ID” before going back to the heap.

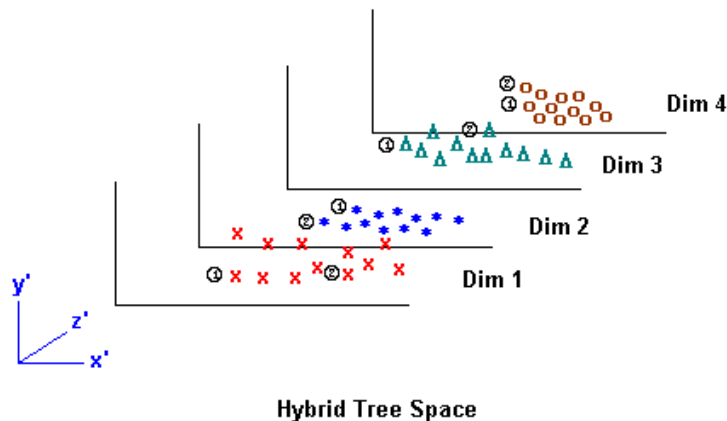


Figure 6: An overview on Hybrid Tree space with four dimensions, which each dimension is distinguished by false z' value

7. Algorithm of Index Tree Search

Although Hybrid Tree comes with its mechanism of data search but still lacking of ability in dealing with spatial query. This paper introduces an old fashion but effective B-tree (Bentley, 1975) like algorithm for the Hybrid Tree to handle spatial search whilst retaining the capability of feature based similarity search. Query results are obtained through several steps. The first step includes sorting the corresponding result from each dimension that has been specified in the query. For example if the query criteria is “X > 10 and Colour = Red”, the searching process will only go through Dimension “X” and Dimension “Colour” and skip the rest. The next step is using operator “AND” or “OR” to undergo comparison between the corresponding results of each dimension with referring to the unique number of “ID”. In the query criteria above, let assume that the records ID that satisfy the condition “X > 10” is {ID: 1, 10, 11, 12, 14, 20} and for the criteria “Colour = Red” is {ID: 6, 11, 12, 16, 17, 20}. As we apply the operator “AND”, the combined

set would be {ID: 11, 12, 20}, otherwise if “OR” is applied, the combined set would be {ID: 1, 6, 10, 11, 12, 14, 16, 17, 20}.

Figure 7 demonstrates data selection in each dimension based on three types of query: overlap, nearest-neighbourhood and bounding region. Overlap and nearest-neighbourhood query can be based on point or non-zero dimension feature, meanwhile bounding region query only based on non-zero dimension feature. The algorithm for overlap and bounding region query seems straightforward but is more complex when combined nearest-neighbourhood query. For example if the query is searching for twenty buildings that are located closest to a given position, the search process might have to be iterated for a few round until twenty features are selected into the result pool. The iteration of nearest-neighbourhood search starts with the search for overlapped features because distance between two overlapped features should be considered as zero unless a more complicated relationship type is defined. In Figure 7a, since it is referred to feature with extension and assuming that the false coordinate of reference feature is (START', END'), so {START' < END'}, everything that is START' before a and END' after a, START' before b and END' after b, whereas {a ≤ END' and b ≥ START'} would be selected. It is similar to overlap query based on point (Figure 7b), but {START' = END'}. Figure 7c and 7d show that the nearest-neighbourhood query based on feature with extension and point are similar to overlap query but iteration is imposed for different zone if the previous search did not get enough quantity of results. Figure 7e shows that the bounding region search is the simplest, as {START' < END'}, everything that is START' after a and END' before b, whereas {a ≤ END' and a ≥ START' and b ≤ END' and b ≥ START'}

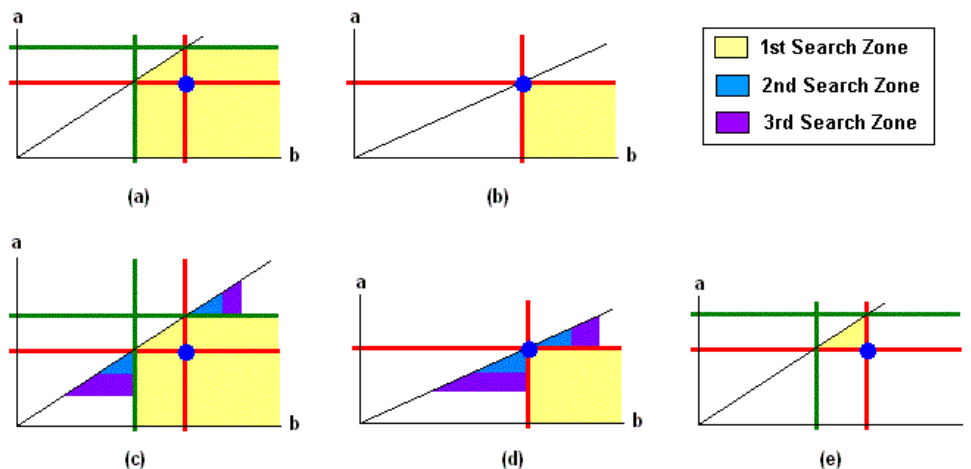


Figure 7: (a) Overlap query based on non-zero dimension feature (b) Overlap query based on point (c) Nearest-neighbourhood query based on non-zero dimension feature (d) Nearest-neighbourhood query based on point (e) Bounding region query based on non-zero dimension feature. Search only concentrates in right side of the centreline ($a \leq b$)

8. Trial Run

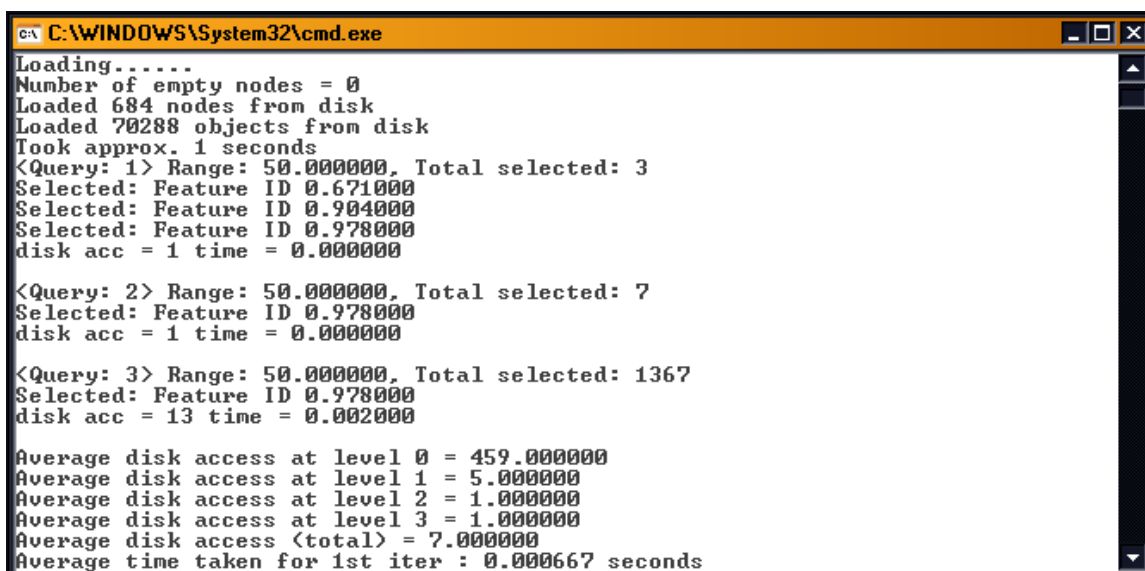
The trial run system is running on Windows XP Home Edition Version 2002 and Pentium 4 Mobile 1.6 GHz system with 256 MB of RAM. Geometry data was extracted from the graphics files of Open Inventor and Virtual Reality Mark-up Language (VRML), and data about colour was added randomly. The geometry data was compressed and mapped as discussed in previous sections, and the index was created and all information were finally stored in Access database. The index structure of Hybrid Tree was defined as following: {x', y', z', GROUP_ID, FEATURE_ID}. Tetrahedrons that are contained by DTT were grouped after the compression procedure and the indexing method was tuned to sort out the most possible results upon query and the index was addressed to group instead of every single tetrahedron.

Total 991 buildings were inserted into database and the most complicated building feature has 2139 tetrahedrons that were generated from 730 building nodes. After compression, tetrahedrons were grouped into 645 subsets and the total number of elements was 4074, which is 4482 smaller than the original total (2139 tetrahedrons x 4 vertices for each tetrahedron = 8556 vertices). Since separator needs to be added to distinguish two groups, the ratio of compression becomes $(4074 + 645) \div 8556 \approx 0.55$. The compression ratio for the entire dataset of 991 buildings ranged from 0.49 to 0.55, as less complicated feature has better compression ratio due to the ability to group the tetrahedrons into fewer subsets. Total 23,390 groups were generated for the entire set of buildings feature and four dimensions were indexed: X, Y, Z and COLOUR. Due to one group probably has more than one type of colour, so that the total mapped points in Hybrid Tree space was $(23390 \times 4) +$ total faces with different colour. When performing a search for buildings that overlap with a given point, it took less than 1 second (Figure 8).

9. Conclusion & Future Work

Data model is the core of GIS and any modification on the data model structure or design could incur changes in other respective components, nevertheless, the proposed data model needs to be further extended in the future to accommodate more information such as photogrammetric facets and landscape scenes. At the other hand, the performance of data mining is highly dependant on the mechanism that has been used by both compression technique and indexing method. This paper has presented the vertices reorganizing approach of compression technique and Hybrid Tree for the indexing method, and despite the trial run results has proven the combination is working well in handling the spatial as well as feature based similarity queries, there still have plenty of rooms for improvement. The trial run system is currently being tested on single response time but testing on multiple response time will be concluded as one of the main agendas for future work.

In the meantime, an interface should be built to visualize and manipulate the 3D dataset. Nowadays, open source visualization applications could help to cut down the workload as well as time-scale of development and at the other hand, direct hardware support for volume rendering despite has recently available for regular grids at fixed resolutions. The available of direct hardware support for the time being may not much beneficial to the tetrahedral mesh rendering, nevertheless, some researchers have proven the tetrahedral mesh rendering is already begun to achieve interactive frame rates by using carefully optimised algorithms for visibility sorting and by taking full advantage of triangle-rendering hardware (King et. al., 2001).



```
C:\WINDOWS\System32\cmd.exe
Loading.....
Number of empty nodes = 0
Loaded 684 nodes from disk
Loaded 70288 objects from disk
Took approx. 1 seconds
<Query: 1> Range: 50.000000, Total selected: 3
Selected: Feature ID 0.671000
Selected: Feature ID 0.904000
Selected: Feature ID 0.978000
disk acc = 1 time = 0.000000

<Query: 2> Range: 50.000000, Total selected: 7
Selected: Feature ID 0.978000
disk acc = 1 time = 0.000000

<Query: 3> Range: 50.000000, Total selected: 1367
Selected: Feature ID 0.978000
disk acc = 13 time = 0.002000

Average disk access at level 0 = 459.000000
Average disk access at level 1 = 5.000000
Average disk access at level 2 = 1.000000
Average disk access at level 3 = 1.000000
Average disk access (total) = 7.000000
Average time taken for 1st iter : 0.000667 seconds
```

Figure 8: Result shows average disk access and time taken for loading the index structure from disk and performing search for buildings that overlap with a given point

10. Acknowledgement

The authors express thanks to the Department of Computer Science, University College London for providing the data used in this research, also, the Database Research Group (2003) in University of California at Irvine and Dr. Kaushik Chakrabarti for providing the source code of Hybrid Tree, and finally Hang Si in the Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Berlin, Germany for providing the source code of TetGen.

References

1. Bentley, J., 1975, "Multidimensional Binary Search Trees Used for Associative Searching", *Communications of the ACM*, Vol. 18, No. 9, September 1975, pp. 509-517
2. Castelli, V., 2001, "Multidimensional Indexing Structures for Content-Based Retrieval", IBM Research Report, RC 22208 (98723) February 13, 2001, Chapter 4-6, pp. 19-26
3. Chakrabarti, K. and Mehrotra, S., 1999, "The Hybrid Tree: An Index Structure for High Dimensional Feature Space", In *Proceedings of IEEE International Conference on Data Engineering*, March 1999
4. Chen, X. and Murai, S., 1999, "Analysis and Visualization of 3D Geospatial Data by Using Delaunay Tetrahedral Tessellation", In *Proceedings of Asian Conference on Remote Sensing (ACRS)*, Sessions: Measurement and Modeling
5. Gaede V. and Günter, O., 1998, "Multidimensional Access Methods", *ACM Computing Surveys*, June 1998, Vol. 30, No. 2, pp. 170-231
6. Gumhold, S., Guthe, S. and Straßer, W., 1999, "Tetrahedral Mesh Compression with the Cut-Border Machine", In *Proceedings of the 10th Annual IEEE Visualization Conference*, October 1999
7. Guttman, A., 1984, "R-trees: A Dynamic Index Structure for Spatial Searching", In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Boston, MA, pp. 47-57
8. King, D., Wittenbrink, C. and Wolters, H., 2001, "An Architecture for Interactive Tetrahedral Volume Rendering", International Workshop on Volume Graphics, June 21-22, 2001, Stony Brook, New York
9. Kofler, M., Gervautz, M. and Gruber, M., 2000, "R-trees for Organizing and Visualizing 3D GIS Database", *J. Visualization and Computer Animation*, Vol. 11, pp. 129-143
10. Li, J., Jing, N., Sun, M., 2001, "Spatial Database Techniques Oriented to Visualization in 3D GIS", In *Proceedings of 2nd International Symposium on Digital Earth*, 24-28.06.2001, Fredericton, New Brunswick, Canada
11. Nievergelt, J., Hinterberger, H. and Sevcik, K., 1984, "The Grid File: An Adaptable, Symmetric Multikey File Structure", *ACM Transactions on Database Systems (TODS)*, Vol. 9, No. 1, pp. 38-71, March 1984
12. Oosterom, P.V., 1990, "Reactive Data Structures for Geographic Information Systems", PhD thesis, Department of Computer Science, Leiden University, December 1990
13. Orenstein, J., 1986, "Spatial Query Processing in An Object-oriented Database System", In *Proceedings of 1986 ACM SIGMOD International Conference on Management of Data*, pp. 326-336
14. Pilouk, M., 1996, "Integrated Modeling for 3D GIS", PhD thesis, ITC publication, The Netherlands
15. Robinson, J., 1981, "The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes", In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 10-18
16. Wade, A. E., 1998, "Hitting The Relational Wall", White paper, Oracle white paper library, 1 January 1998
17. Yang, C., Mitra, T. and Chiueh, T., 2000, "On-the-Fly Rendering of Losslessly Compressed Irregular Volume Data", In *Proceedings of IEEE Visualization '2000*, October 2000

18. Zlatanova, S. and Tempfli, K., 2000, "*Modeling for 3D GIS: Spatial Analysis and Visualization Through The Web*", IAPRS, Vol. XXXIII, Working Group IV/2, Amsterdam
19. Database Research Group, 2003, <http://www-db.ics.uci.edu/pages/index.shtml>, Information & Computer Science, University of California at Irvine
20. ERDAS, 2003, <http://gis.leica-geosystems.com>, Leica Geosystems GIS and Mapping, August 2003
21. ESRI, 2003, <http://www.esri.com>, Environmental Solutions Research Institute, August 2003
22. GeoMedia, 2003, Version 5.1, <http://www.intergraph.com/geomedia>, Intergraph Mapping and Geospatial Solutions (IMGS), May 2003
23. Intergraph, 2003, <http://www.intergraph.com>, Intergraph Inc., August 2003
24. Medit, 2002, Version 2.2, <http://www-rocq.inria.fr/gamma/medit>, Gamma project, INRIA-Rocquencourt, September 2002
25. Multigen, 2003, <http://www.multigen.com>, Multigen-Paradigm Inc., August 2003
26. SiteBuilder3D, 2003, <http://www.sitebuilder3d.com>, Multigen-Paradigm Inc, January 2003
27. TetGen, 2003, Version 1.2c, <http://tetgen.berlios.de>, Weierstrass Institute for Applied Analysis and Stochastics (WIAS), 8 May 2003