IMPROVING 3D SPATIAL QUERIES SEARCH: NEWFANGLED TECHNIQUE OF SPACE FILLING CURVES IN 3D CITY MODELING

U. Uznir[a], F. Anton[a b], A. Suhaibah[a], A. A. Rahman[a] and D. Mioc[b]

[a] Dept. of Geoinformation, Faculty of Geoinformation and Real Estate,
Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
mduznir@utm.my, norsuhaibah@gmail.com, alias@utm.my
[b] Dept. of Geodesy, National Space Institute, Denmark Technical University, 2800 Lyngby, Denmark
{fa,mioc}@space.dtu.dk

Commission II WG II/2

KEY WORDS: 3D Spatial Data Management, 3D Space Filling Curves, 3D City Modeling, 3D Adjacency, 3D Indexing

ABSTRACT:

The advantages of three dimensional (3D) city models can be seen in various applications including photogrammetry, urban and regional planning, computer games, etc.. They expand the visualization and analysis capabilities of Geographic Information Systems on cities, and they can be developed using web standards. However, these 3D city models consume much more storage compared to two dimensional (2D) spatial data. They involve extra geometrical and topological information together with semantic data. Without a proper spatial data clustering method and its corresponding spatial data access method, retrieving portions of and especially searching these 3D city models, will not be done optimally. Even though current developments are based on an open data model allotted by the Open Geospatial Consortium (OGC) called CityGML, its XML-based structure makes it challenging to cluster the 3D urban objects. In this research, we propose an opponent data constellation technique of space-filling curves (3D Hilbert curves) for 3D city model data representation. Unlike previous methods, that try to project 3D or n-dimensional data down to 2D or 3D using Principal Component Analysis (PCA) or Hilbert mappings, in this research, we extend the Hilbert space-filling curve to one higher dimension for 3D city model data implementations. The query performance was tested using a CityGML dataset of 1,000 building blocks and the results are presented in this paper. The advantages of implementing space-filling curves in 3D city modeling will improve data retrieval time by means of optimized 3D adjacency, nearest neighbor information and 3D indexing. The Hilbert mapping, which maps a sub-interval of the $[0, 1]$ interval to the corresponding portion of the d-dimensional Hilbert's curve, preserves the Lebesgue measure and is Lipschitz continuous. Depending on the applications, several alternatives are possible in order to cluster spatial data together in the third dimension compared to its clustering in 2D.

1 INTRODUCTION

3D spatial city models in different applications require different analyses and purposes. It is not an immense exertion if it just meant for 3D object visualization (see (van Lammeren et al., 2010) and (Nichol et al., 2010)). In most critical applications, information retrieval time is important (see (Dash et al., 2004) and (Tomaszewski, 2011)). Without a proper procedure of storing 3D spatial data, queries for each object will reduce the time efficiency of processing and information retrieval. Meanwhile for mobile applications, displaying all 3D objects in a single display consume a lot of processing-memory allocation and will decelerate device performance. Alternatively, by knowing which 3D objects are within a certain radius or distance from current location would be a useful implementation for mobile application display.

On the other hand, 3D data (spatial and semantic) consume more disk storage compared to 2D data. Requesting 3D spatial data from servers requires step by step memory disk searching and would be a hassle if the information was stored in different servers in different agencies throughout a country. Yet, again, 3D spatial data with proper data constellation will boost the retrieval time and optimize the processing procedure.

Therefore, in this paper, a new method of storing 3D spatial city models is presented. It is useful for various applications and improves data retrieval time by providing 3D adjacency, nearest neighbor information and 3D indexing. The advantages of implementing the space-filling curve in 3D city model will be explained as a new method of organizing 3D data in an opponent arrangement.

This paper is organized as follows. In Section 2, we review the recent trends in 3D Geoinformation Sciences related to 3D spatial queries. In Section 3, we review the CityGML standard. Then, we review space-filling curves and we introduce our own 3D Hilbert space-filling curve implementation in Section 4. Finally, we present the analysis and results in Section 5 and conclude this paper in Section 6.

2 TRENDS IN 3D GEOINFORMATION SCIENCES

Visualization of three dimensional (3D) objects become more widespread in developments (see (Behley and Steinhage, 2009), (Jin and Bian, 2006) and (Liu and Fang, 2009)). This can be seen from demand in 3D based applications (see (Freitas et al., 2010) and (Zhang et al., 2009)). Visualization of 3D objects can support a more realistic view as in the real environment than the two-dimensional (2D) visualization. 3D view of a building model is more realistic compared to 2D floor plan of a building in a navigation application.

In commercial software development, private companies are competing in developing tools that are capable of managing 3D cities. ESRI (CityEngine), Bentley (Bentley's Map V8i) and Google (Google Earth) offer users the capability to create, visualize and measure 3D cities in their products (Jazayeri, 2012).

Since quite a number of 3D city model formats are available, there is a need for standardizing the 3D city model format for various applications. City Geography Markup Language (CityGML) is an example of an exchange standard format for 3D city models (see Figure 1). It consists of different Levels of Details (LOD); LOD0, LOD1, LOD2 and LOD3. Different LODs reflect different 3D spatial information details. The higher the LoD, more object detail and geometry is included. This common information model is the first standard related to 3D city models (Jazayeri, 2012).

Users want to visualize and analyze their 3D city models (see (Over et al., 2010) and (Mao et al., 2011)). Among the main steps in analyzing 3D city models, there is the data retrieval for a specific computing process. To seek specific information in a huge volume of data storage, a proper data constellation mechanism is needed. Although the data is retrievable with today's computer technology, the data retrieval routine is not optimal and there is no reason to sacrifice the computational performance instead of using it for more intricate procedures.

The construction of large-scale 3D city models faces the problem that data sets are too huge, the mission is too heavy (in terms of loading, managing and editing) and the development cycle is too long (Zhang and Shen, 2008). Two major issues in using 3D city model data have been listed in previous research on measuring the impact of 3D data geometric modeling on spatial analysis (Brasebin, M. et al., 2012). First, more complex data implies an increase in time and memory usage for the analysis (and calls for more research on the efficiency of the algorithms used). Second, detailed 3D urban data are complex to produce, expensive and it is important to be well informed in order to decide whether or not to invest in such data.

Although a lot of problems arise, researchers have put determination in improving the CityGML standard. Improving existing CityGML standard platform for practicality is better rather than starting from scratch as developing CityGML standard took many years of development. The next section will describe the CityGML technical structures in brief for understanding.

3 CITY GEOGRAPHY MARKUP LANGUAGE (CITYGML)

The City Geography Markup Language (CityGML) is an open standard for 3D city and landscape modeling that is recognized by the Open Geospatial Consortium (OGC) and the ISO TC211. The usage of the CityGML standard can be seen in urban and landscape planning, 3D cadasters, environmental simulations, mobile phone navigation applications, disaster management, vehicle navigation, training simulators, mobile robotics, computer graphics, computer vision and computer games. CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models and it is implemented as an application schema for the Geography Markup Language version 3.1.1 (GML3).

CityGML implements different scales called the Levels of Detail (LOD). Each five LODs are based on specific models required in various applications. LOD1 is the well-known blocks model comprising prismatic buildings with flat roofs. Meanwhile, buildings in LOD2 have differentiated roof structures and thematically differentiated surfaces. LOD3 and LOD4 are more detailed city models designed for outdoor and indoor applications respectively, that require higher accuracy for implementations (see Figure 2).

Number of Buildings	Total	Size (KB)	Number of Lines
2 x 2	4	17	708
10 x 10	100	394	17,412
20 x 20	400	1,592	69,612
50 x 50	2,500	10,084	435,012

Table 1: CityGML Size (in KB) and number of lines comparison between different numbers of buildings

CityGML defines the classes and relations for objects in cities with respect to their geometrical, topological, semantical and appearance properties. This information is stored in a structured XML-based format with different tags for identifications. Figure 3 shows a representation of a single building in CityGML data. CityGML constructs a building with four façades by having 6 polygonal faces consisting of quadrilaterals defined by the coordinates (x, y, z) of their vertices each. The 6 polygonal faces can be identified as `gml:Polygon` tags. Whereby each tag has its own unique `gml:id` attribute name identification. For each polyhedron, there is a sub-child tags named `gml:LinearRing` and `gml:PosList`. Information regarding coordinates are stored in `gml:PosList` tags in counterclockwise coordinates sequences for faces that are facing outside and clockwise coordinates sequences for faces that are facing inside. A four sided rectangle is represented by 5 tuples of coordinates (x, y, z). Usually the first and the fifth coordinate sets refer to the same coordinate values in order to create a boundary for the polyhedron. Basically, this sequence is a cycle of coordinates (points). Polyhedrons that "belong" to a building are grouped in the same set. In the example shown (see Figure 3), the group can be identified as `bldg:Building` tags.

However, CityGML requires 3.12KB of memory / disk storage for the building in Figure refCEDS3 (102 lines of tags and elements in XML structure). Table 1 shows the storage size for different numbers of LoD1 objects in CityGML. Retrieving information from CityGML data requires an XML tag list searching. The comparison in Table 1 shows a total line of tags for each category for a number of buildings.

Meanwhile, Figure 4 shows the disk storage comparison of LoD2 objects. Data used as a benchmark test in Figure 4 are downloadable at `www.citygmlwiki.org`. Categorizations were made based on two categories which are LoD2 MultiSurface and LoD2 bounded by WallSurface, RoofSurface and GroundSurface. LoD2 bounded by WallSurface, RoofSurface and GroundSurface is a method of classifying the 3D objects into several parts such as wall, roof or ground surface of a building. This is a common way of describing CityGML data for 3D implementations. Figure 4 shows that the disk storage for LoD2 bounded by WallSurface, RoofSurface and GroundSurface rocketed more than double of LoD2 MultiSurface disk storage size. This graph indicates that it is vital to organize 3D city model data for optimizing 3D spatial queries searches.

Therefore, in the next section, we propose the data constellation mechanism by using space-filling curves for 3D city model data. The 3D city data will be stored in a more structured way for various applications. At the same time, it will provide advantages in terms of providing common spatial topology information such as 3D adjacency and it will therefore facilitate nearest neighbor queries.

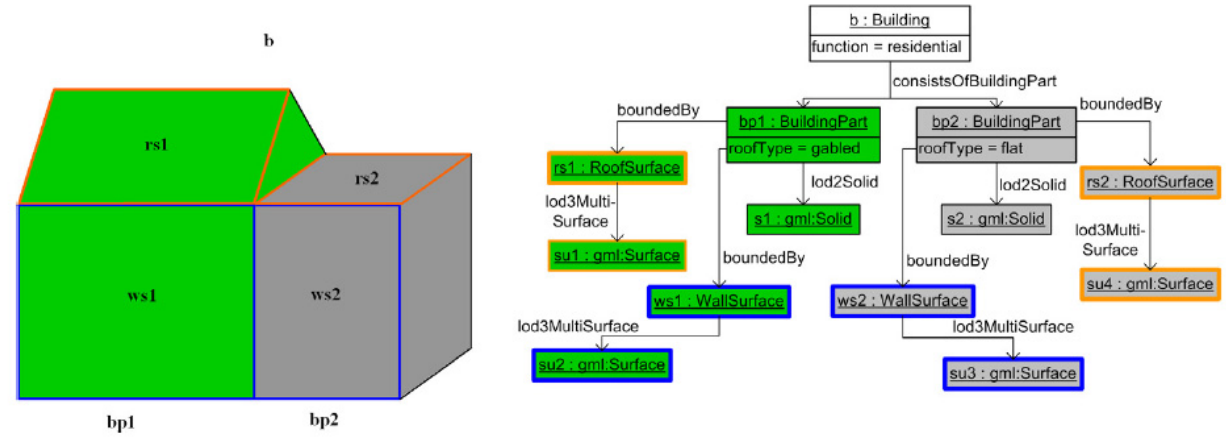

Figure 1: LoD2 building illustration: CityGML feature structure as UML instance diagram (taken from (Gröger and Plümer, 2012))

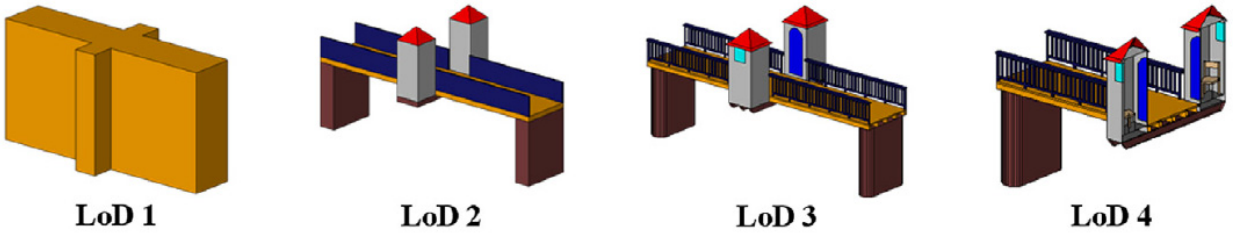

Figure 2: The four levels of detail (LoD) defined in CityGML (taken from (Gröger and Plümer, 2012))

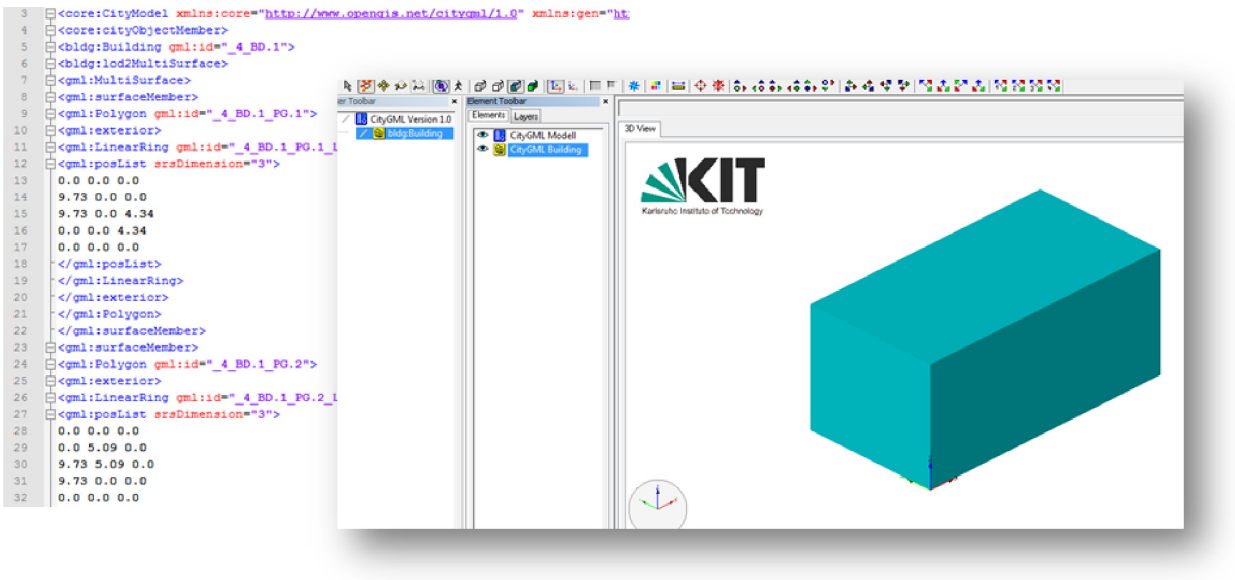

Figure 3: Representation of a building in CityGML done using Karlsruhe Institute of Technology's IfcExplorer for CityGML (Karlsruhe Institute of Technology, 2013)

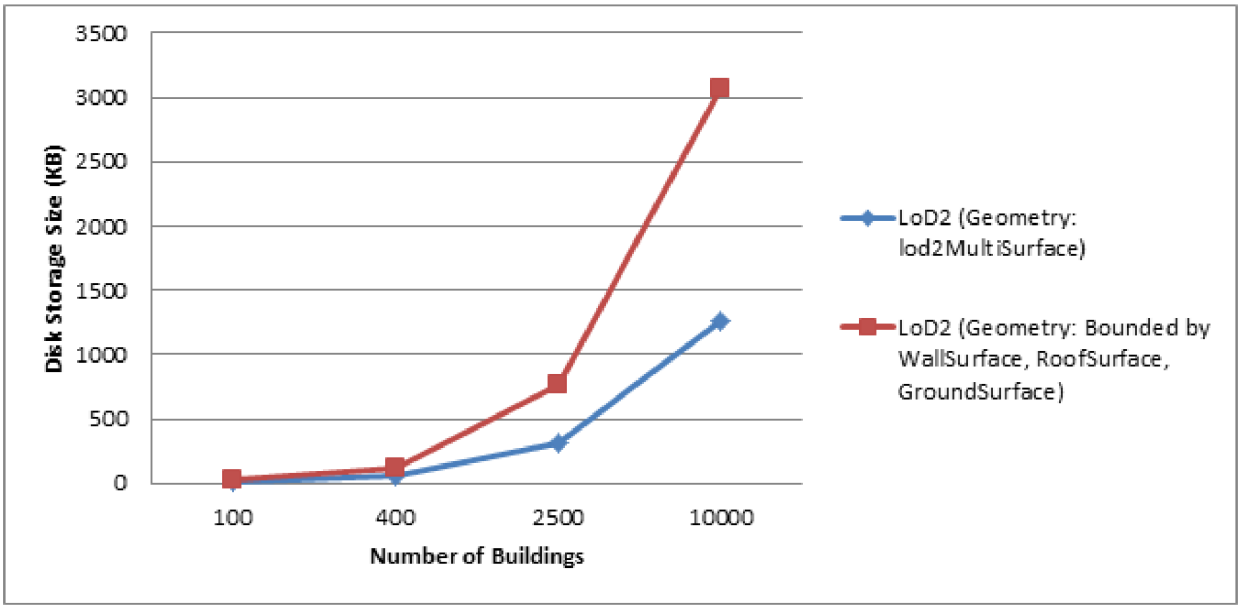

Figure 4: Disk storage comparison between two LoD2 categories

4 SPACE-FILLING CURVES AND HILBERT'S CURVE

A space-filling curve is a continuous function with its domain interval $[0, 1]$ and its range could be in any topological, uniform or Euclidean space. Space-filling curves allow a proximity problem to be reduced from higher dimensional spaces to a binary search through a one-dimensional space. Such proximity problems usually involve sorting in a one-dimensional space. Many applications can be benefited from space-filling curves, but the most common application of space-filling curves is for storage and retrieval of multidimensional data in a spatial database. A proximity problem for which space-filling curves have been used regularly is the approximate nearest neighbor search or finding closest points from one location. Until now, there are several variants of space-filling curves such as Hilbert's curve, Dragon Curve, Gosper's Curve, Moore's Curve, Z-Order and many more.

The space-filling curve was discovered by Giuseppe Peano in 1890 and called the Peano's Curve (Peano, 1890). From his finding, a curve must pass through every point on a plane at least once. But, before his finding, in 1878, George Cantor has demonstrated a one-to-one correspondence between the unit interval and the unit square (Thiele and Wos, 2002). In 1879, Netto added that any such mapping could not be continuous. In 1891, David Hilbert discovered the Hilbert Curve structure, which is a simpler variant of the same idea of Peano's space-filling curve (Kevin, 2008). The Hilbert Curve subdivides squares into four smaller squares instead of Peano's nine smaller squares. Figure 5 shows four iterations of Peano's Curve.

When David Hilbert discovered a variation of Peano's Curve, he divided each side of unit square into two parts equally. This yields the square into four smaller squares. From the divided squares, each of them is divided into another four smaller squares, and so on. For each part of this division, a curve will traverse all the squares. The Hilbert's Curve normally starts at the lower left division and ends at the lower right division. According to its movement and process, the Hilbert's curve could be expressed in base 2.

The Hilbert space-filling curve could be described as an underlying grid, a $N \times N$ array of cells where $N = 2^n$. Hilbert's enumeration of the squares is shown in Figure 6 for $n = 1, 2, 3$. Note that the first square is always at the lower left corner and the last square is always in the lower right corner.

From the divided square, we assume that the $N \times N$ cells coordinates start at $(0, 0)$ in the lower left corner. From the assumptions, Hilbert's curve has corners at integers ranging from 0 to $2^n - 1$ in both x and y and we take a positive direction along the curve from $(x, y) = (0, 0)$ to $(x, y) = (2^n - 1, 0)$.

4.1 Mapping Points Using Hilbert's curve

Figure 7 shows a set of points that will be used to demonstrate the 2D Hilbert's space-filling curve.

Figure 7: Scattered points (Kevin, 2008)

The following constructions mapped the unit interval to a unit square of the Hilbert's curve method.

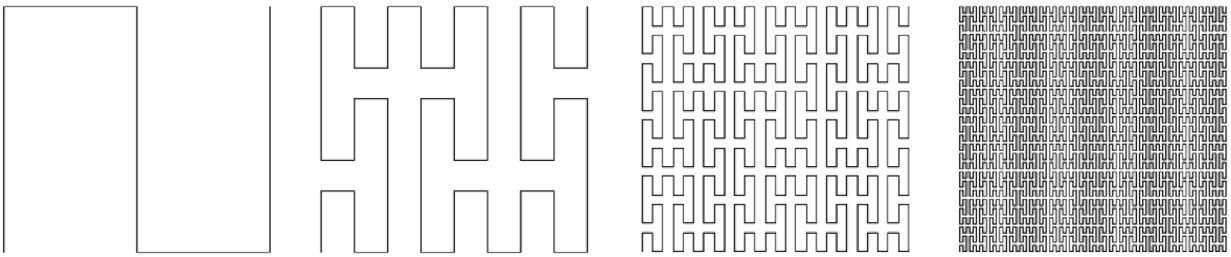

Figure 5: First four iterations of Peano's curve

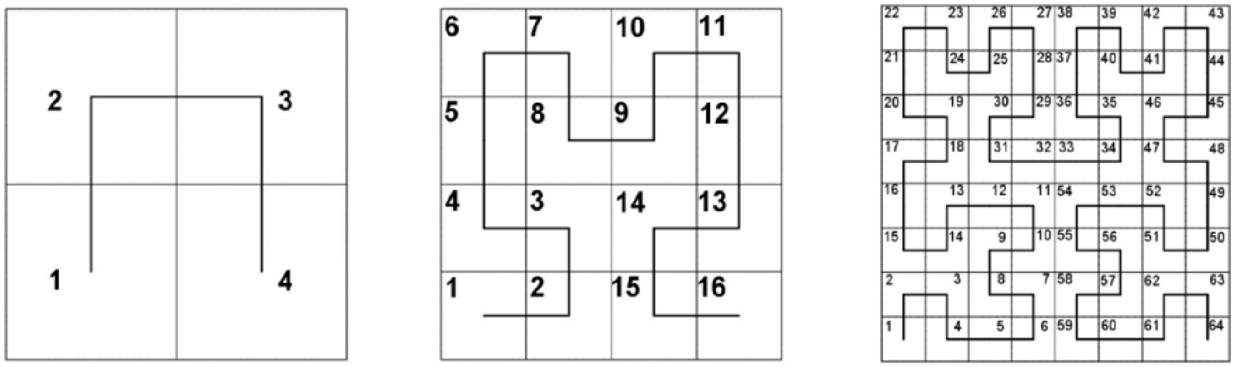

Figure 6: The first three stages in generating Hilbert's curve

- Each unit interval and unit square is divided into four intervals as shown in Figure 8. Every unit interval is assigned to one square. Figure 8 (a-c) shows the first three steps of this construction. At the limit (when the number of elements goes towards infinity), this produces a subjective, continuous map from the unit interval to the unit square.
- Figure 8 (d) shows the resulting order of scattered points in Figure 7. Since the Hilbert's curve starts in the lower left corner and ends in the lower right corner, the last edge of the round-trip (shown in Figure 8 (d) as dashed lines) from the last corner back to the first corner is much longer than its Euclidean distance.

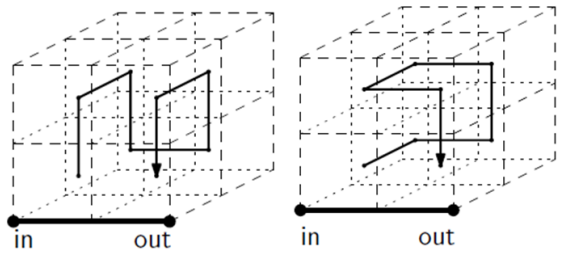

Figure 9: Possible approach of 3D Hilbert order where n = 2 (Haverkort and Walderveen, 2008)

In this research, we adopt space-filling curves for the ordering in 3D city models. In mathematical analysis, a space-filling curve is a curve whose range contains the entire 2-dimensional unit square (or more generally a n-dimensional hypercube). This space-filling curve visits every point in a square grid with a size of n^2. Space-filling curves in this research were used to compress the three-dimensional models to a one-dimensional ordering. Based on previous research in implementing 3D Hilbert's curve for spatial structures, there are 1536 diverse methods of 3D Hilbert's curve implementation (Haverkort and Walderveen, 2008). Meanwhile, Figure 9 shows the 3D Hilbert's curve implemented in this research with the same size 2^n and the rationalization of the test data based on simple 3D city buildings. We believe that this 3D Hilbert curve will benefit transportation and air pollution applications, where most of the important spatial adjacencies happen in a single horizontal layer. Results from the implementation of 3D Hilbert's curve for 3D city model will be discussed in the next section while showing the advantages of smaller data retrieval time and processing.

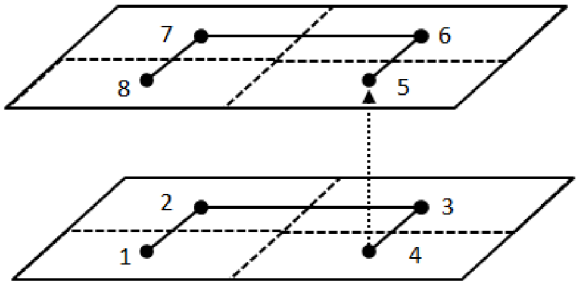

Figure 10: 3D Hilbert Order

5 ANALYSIS AND RESULTS

In critical applications, information retrieval time is important. Without a proper arrangement of 3D spatial data, a single query iteration needs to be executed through each object and it will reduce time efficiency for processing and information retrieval. The search routine should be optimized and there is no reason to sac-

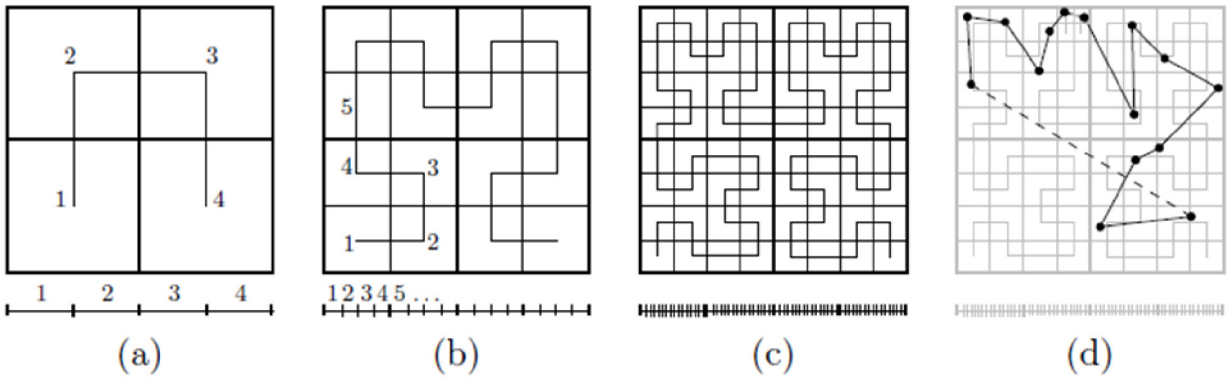

Figure 8: Hilbert's curve and Order (taken from (Kevin, 2008))

rifice the computational performance instead of using it for more intricate procedures.

Since indexing improves the data retrieval capabilities, two types of search analyses were performed. Those analyses are single object search and nearest object from target search. The single object search usually is implemented in cases where the user wants to identify a single object from the 3D city model data. For an example, search based on building names or ID. Meanwhile the nearest object from target search is meant for queries like finding the adjacent buildings or pinpointing nearest facilities from the current location. In comparison, two analyses were prepared for different data sets of buildings (10, 50, 100, 200, 400, 600, 800 and 1000 buildings). Data retrieval times were measured in milliseconds and it was tested in a system using a single Intel Core i7 running at 2.2GHz with 8GB of Random Access Memory (RAM).

The graph shown in Figure 11 shows two types of 3D spatial data sets used in the single object search: CityGML data and 3D spatial data with space-filling curve implementation. The ordering for space-filling curve is based on our 3D Hilbert's curve implementation. Eight data sets were prepared for each type respectively. From the figure shown, it indicates that for single object search, there is not much difference in time measured between both types of sets. This is due to the single object search routine is based on line tags list searching. Times measured in Figure 11 included times for displaying tags that had been traversed before getting to the target object.

On the other hand, the advantage of space-filling curve operations can be seen in the consistency between neighboring pixels. In this case, the adjacencies between spatial objects were conserved and retrievable. The example shown in Figure 12 identifies objects (buildings) that are located "closed to" the building of interest. Since the Hilbert's curve is organized into a one-dimensional structure, finding the nearest building to a specific building (n) is done by first finding the nearest arc length to n and then checking whether this corresponds indeed to the closest neighbor. Indeed, the ordering induced by the difference (Δ) of arc length does not correspond to the ordering of the distance between the projection of the object onto the Hilbert's curve and building n.

Another experiment conducted in this research is to test the performance in data retrieval time for finding the nearest building from the target building. Figure 13 illustrates flowcharts in both implementations for comparison assessment. For this experiment, displaying the traversed tags is excluded and only finding the nearest building is the sole objective. Table 2 and Figure 14 show

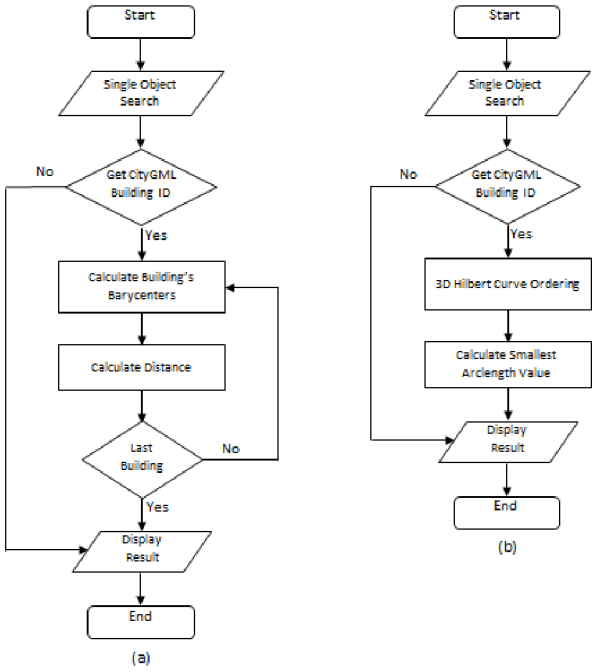

Figure 13: Flowcharts for finding the nearest building for CityGML data (a) and with 3D Hilbert's curve implementation (b)

the running times for queries performed and measured for finding the nearest building to the target building.

Since CityGML data are based on XML structured ordering, searching the target objects is based on tags list and calculations are performed as in Figure 13 (a). Results shown in Table 2 can be interpreted as buildings located farthest (i.e., building ID 1 is the closest meanwhile building ID 1000 is the farthest in XML structure) require more time for data retrieval procedure. Contrariwise, the highlight should be given at the time elapsed between 3D city model with and without space-filling curve implementation. On average, the space-filling curve implementation boost up more than 90% faster compared to CityGML data. It shows that 3D Hilbert's curve in 3D city model is capable of organizing data in a more efficient way. Hence, space-filling curve implementation in large data sets will optimize the data retrieval time while preserving the nearest neighbor information which is significant in geospatial analysis.

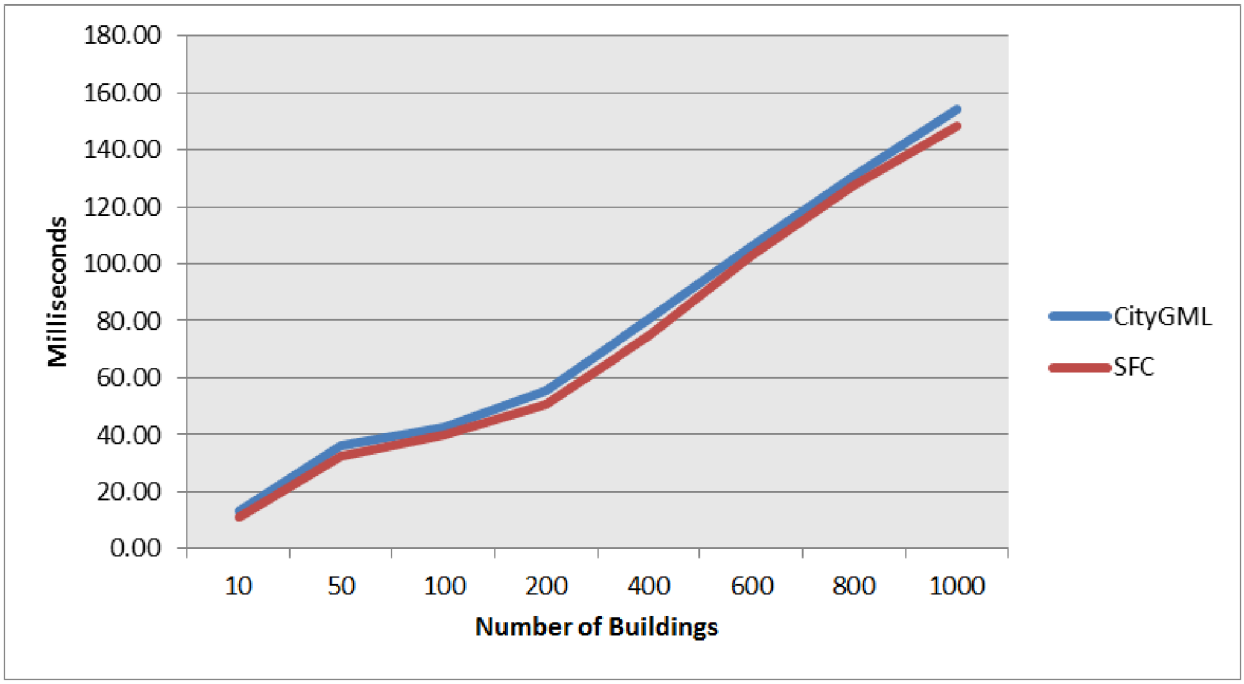

Figure 11: Data retrieval time for single object from target search

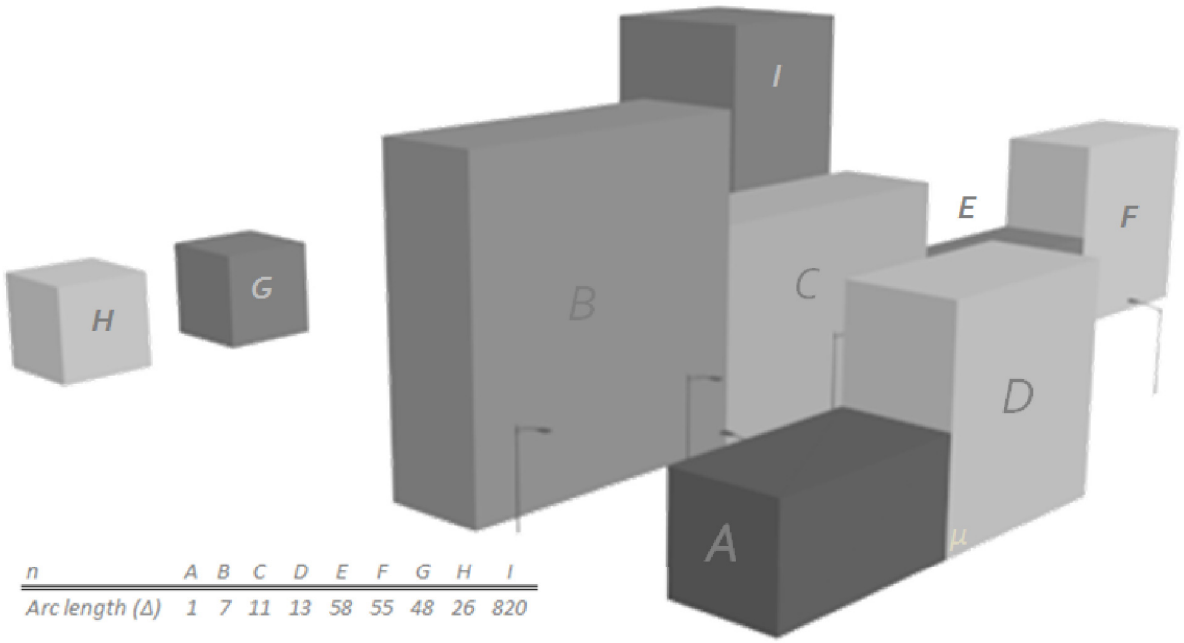

Figure 12: The nearest arc lengths (Δ) from building A are B, C and D respectively

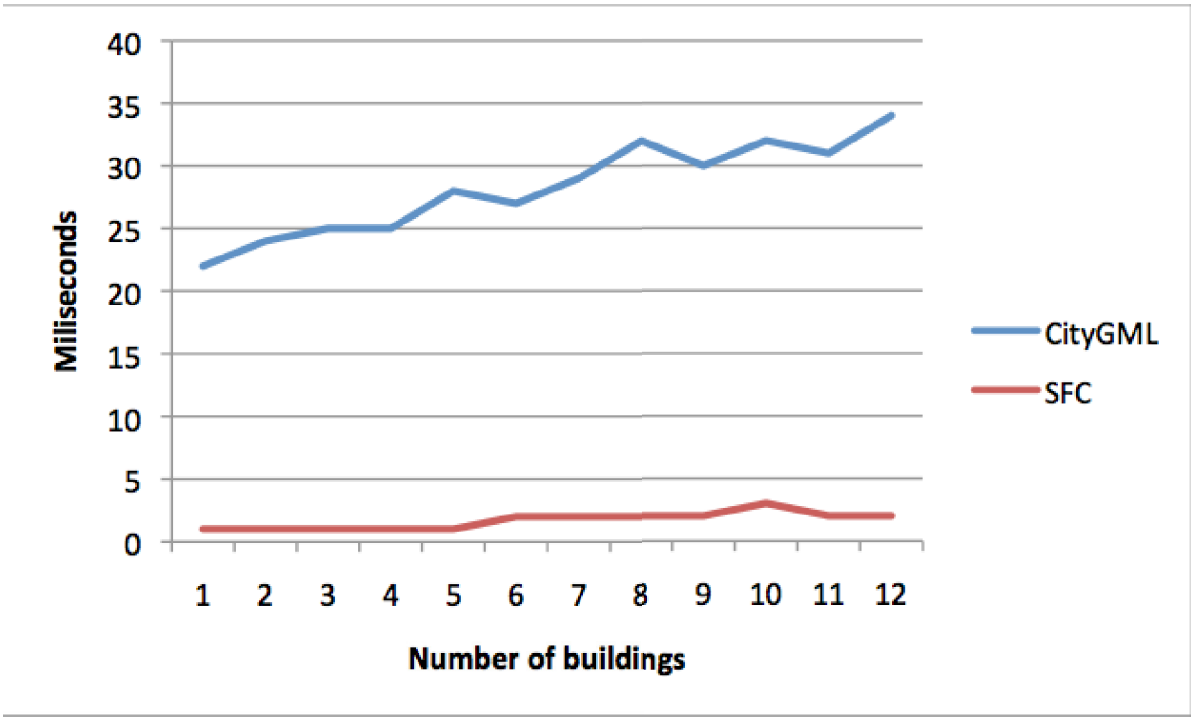

Figure 14: Data retrieval time for nearest object from target search

Target Building ID	Nearest Building ID	CityGML (ms)	space-filling curve (ms)	Elapsed (%)
46	56	22	1	95.5
440	540	24	1	95.8
445	545	25	1	96.0
498	598	25	1	96.0
571	539	28	1	96.4
704	800	27	2	92.6
799	753	29	2	93.1
816	817	32	2	93.8
883	884	30	2	93.3
900	804	32	3	90.6
928	963	31	2	93.5
952	951	34	2	94.1

Table 2: Data retrieval time for nearest object from target search

6 CONCLUSIONS

In spatial applications, to load the entire 3D city model into a viewer will consume computational processing time, and in some applications it might not be necessary at all. Researchers are facing problems in finding which visualization technique is best suited for displaying 3D city data (Métral, C. et al., 2012). As an example, in an indoor environment, a room is bounded by doors, windows, walls, corridors, and floors. To make the whole 3D city model reachable at one single processing time is not feasible nor necessary. Practically, by using space-filling curves implementation, knowing which objects are within view of sight will improve the visualization and optimize the memory. Meanwhile, for mobile applications, displaying all 3D objects in a single display consumes a lot of processing-memory allocation and will decrease the device performance. By knowing which 3D objects have their projection on the 3D Hilbert curve within some arc-length from the current location's projection would be a great idea for mobile display.

We have proposed a new 3D Hilbert's curve as a space-filling curve technique, that could be seen as an alternative to CityGML for optimizing query performance. We also demonstrated the query performance comparison of 3D building data set with and without 3D Hilbert's curve implementation. From the test, we could see that the implementation of space-filling curve method shows advantages in fast acquiring and information retrieval for 3D city model data. The advantages of space-filling curve in preserving neighboring information can be exploited for 3D city model application. However, further studies need to be conducted. This research only focused on buildings as the 3D city object. Since 3D city objects are becoming more sophisticated, other categories of city objects and their relationships are needed. Experiments on street furniture and any other objects included in the CityGML standard should be implemented.

7 ACKNOWLEDGEMENTS

Major funding for this research was provided by the Ministry of Higher Education Malaysia and partial funding was provided by the Land Surveyors Board of Malaysia. This work was done while the second author was Visiting Full Professor at the 3D GIS Research Group, Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia, for which he is very thankful.

REFERENCES

Behley, J. and Steinhage, V., 2009. Generation of 3D City Models Using Domain-Specific Information Fusion. In: Fritz, M and Schiele, B and Piater, JH (ed.), Computer Vision Proceedings, Lecture Notes in Computer Science, Vol. 5815, pp. 164–173.

Brasebin, M., Perret, J., Mustière, S. and Weber, C., 2012. Measuring the impact of 3d data geometric modeling on spatial analysis: Illustration with skyview factor. In: Usage, Usability, and Utility of 3D City Models, p. 02001.

Dash, J., Steinle, E., Singh, R. and Bhr, H., 2004. Automatic building extraction from laser scanning data: an input tool for disaster management. Advances in Space Research 33(3), pp. 317 – 322.

Freitas, M., Sousa, A. and Coelho, A., 2010. A visualization paradigm for 3d map-based mobile services. Communications in Computer and Information Science 68 CCIS, pp. 89–103.

Gröger, G. and Plümer, L., 2012. Citygml - interoperable semantic 3d city models. {ISPRS} Journal of Photogrammetry and Remote Sensing 71(0), pp. 12 – 33.

Haverkort, H. and Walderveen, F. V., 2008. Space-filling curves for spatial data structures.

Jazayeri, I., 2012. University of Melbourne, chapter Trends in 3D land Information Collection and Management, pp. 81–87.

Jin, B. and Bian, F., 2006. Study on visualization of 3d city model based on grid service. Jisuanji Gongcheng/Computer Engineering 32(4), pp. 217–219+235.

Karlsruhe Institute of Technology, 2013. Ifcexplorer for citygml. http://www.iai.fzk.de/www-extern-kit/index.php?id=1570L=1.

Kevin, B., 2008. Organizing Point Sets. PhD thesis, Freie Universitat Berlin.

Liu, L. and Fang, J., 2009. An accelerated approach to create 3d web cities. In: Geoinformatics, 2009 17th International Conference on, pp. 1–4.

Mao, B., Ban, Y. and Harrie, L., 2011. A multiple representation data structure for dynamic visualisation of generalised 3d city models. {ISPRS} Journal of Photogrammetry and Remote Sensing 66(2), pp. 198 – 208.

Métral, C., Ghoula, N. and Falquet, G., 2012. An ontology of 3d visualization techniques for enriched 3d city models. In: Usage, Usability, and Utility of 3D City Models, p. 02005.

Nichol, J. E., Wong, M. S. and Wang, J., 2010. A 3d aerosol and visibility information system for urban areas using remote sensing and {GIS}. Atmospheric Environment 44(2122), pp. 2501 – 2506.

Over, M., Schilling, A., Neubauer, S. and Zipf, A., 2010. Generating web-based 3d city models from openstreetmap: The current situation in germany. Computers, Environment and Urban Systems 34(6), pp. 496 – 507.

Peano, G., 1890. Sur une courbe, qui remplit toute une aire plane. Mathematische Annalen 36(1), pp. 157–160.

Thiele, R. and Wos, L., 2002. Hilbert's twenty-fourth problem. Journal of Automated Reasoning 29(1), pp. 67–89.

Tomaszewski, B., 2011. Situation awareness and virtual globes: Applications for disaster management. Computers Geosciences 37(1), pp. 86 – 92.

van Lammeren, R., Houtkamp, J., Colijn, S., Hilferink, M. and Bouwman, A., 2010. Affective appraisal of 3d land use visualization. Computers, Environment and Urban Systems 34(6), pp. 465 – 475.

Zhang, J. and Shen, T., 2008. Research on construction of large-scale 3d city models based on skyline prototype system. Vol. 7143.

Zhang, Z., Fang, J. and Jing, R., 2009. Interactive visualisation of 3d city models based on adaptive streaming of 3d-gis data. Vol. 2, pp. 417–420.