Review of Spatial Indexing Techniques for Large Urban Data Management

Review of Spatial Indexing Techniques for Large Urban Data Management

Suhaibah Azri, Uznir Ujang, François Anton, Darka Mioc and Alias Abdul Rahman

Abstract Pressure on land development in urban areas causes progressive efforts in spatial planning and management. The physical growth expansion of urban areas to accommodate rural migration gives a massive impact to social, economic and politics of major cities. Most of the models used in managing urban areas are moving towards sustainable urban development which to fulfill current necessities while preserving the resources for future generation. However in order to manage large amounts of urban spatial data, an efficient spatial data constellation method is needed. With the ease of three dimensional (3D) spatial data usage in urban areas as a new source of data input, practical spatial data indexing is necessary to improve data retrieval and management. Current two dimensional (2D) spatial indexing approaches seem not applicable to the current and future spatial developments. Therefore, the objective of this paper is to review existing spatial data indexing approaches to managing large urban area datasets. Each approach will be reviewed and discussed according to the current spatial data scenarios. In addition, a 3D spatial data indexing method will be discussed as an alternative for organizing 3D spatial data.

Suhaibah Azri
3D GIS Research Lab, Universiti Teknologi Malaysia, Malaysia, e-mail: `norsuhaibah@gmail.com`

Uznir Ujang
3D GIS Research Lab, Universiti Teknologi Malaysia, Malaysia, e-mail: `mduznir@utm.my`

François Anton
National Space Institute, Technical University of Denmark, Denmark and Universiti Teknologi Malaysia, Malaysia, e-mail: `fa@space.dtu.dk`

Darka Mioc
National Space Institute, Technical University of Denmark, Denmark, e-mail: `mioc@space.dtu.dk`

Alias Abdul Rahman
3D GIS Research Lab, Universiti Teknologi Malaysia, Malaysia e-mail: `alias@utm.my`

1 Introduction

The urbanization phenomenon and the growth of mega-cities are widespread around the world. The rapid process of urbanization and the growing number of mega-cities cause a lot of economical, social and ecological issues and risks. Affected by this phenomenon, planners or spatial professionals are challenged for making policies and strategies in managing urban cities towards sustainable development. The increasing prominence of urban growth is parallel with the increase of spatial information. Spatial information has become indispensable for numerous aspects of urban development and management. The increasing prominence of spatial data has been due to recent developments in spatial data capturing such as remote sensing, global positioning system, laser scanning and drones for aerial surveying.

Geographic Information Systems (GIS) offer a platform to store, manage, analyze and help decision taking using urban data such as buildings, road networks, surface and subsurface utilities, etc. However, the bottleneck of GIS is the management of large amounts of spatial data. Some of the issues that arise are related to software and hardware limitations in handling large spatial datasets. Furthermore, spatial data come from various sources of varying sizes such as digital maps, satellite images, topographic maps and aerial photographs. Moreover, for urban data management, total data sizes could raise up to terabytes due to the large scale, area and object density. In urban data, spatial objects such as buildings could be built in very dense areas with densities of up to thousands of buildings per square kilometer. In addition, preserving the historical development of an urban area will require a temporal data management and thus, increase the disk storage size as described in Ujang et al (2013).

This issue becomes more crucial when the trends of spatial data are evolving to three-dimensional (3D) spatial data and more specific application frameworks are moving towards using 3D data as a new data input Sharkawi et al (2008); Mohamad Yusoff et al (2010). Nowadays, acquisition of 3D spatial data is much more convenient with various scales and resolution due to acquisition technology such as TLS (Terrestrial Laser Scanning), LIDAR (Light/Laser Detection and Ranging) and drone based surveying. For a practical application, these datasets need to be managed and maintain within a 3D spatial database. Spatial query is a basic function in GIS. In fact, spatial analysis itself originates from spatial query Kun et al (2005). For large areas of urban data management, optimizing spatial queries is needed since the large volume of data will affect the duration of data retrieval. In spatial databases, spatial indexing is one of the key techniques in improving and optimizing spatial query and analysis.

From this paper, user especially spatial professional will get a basic knowledge on which methods is applicable and practical in managing large dataset for different situations or cases especially for 2D and 3D dataset. There are a lot of spatial indexing structures proposed for spatial query operation. However, not all are suited to 3D application in urban data management. Sometimes the processing of 3D data is based on the framework of 2D index structure which lead to data retrieval inefficiency as mentioned by Gu et al (2011). Thus, in this paper several methods will be investigated to see the potential or applicability of each structure in handling a huge amount of urban data. The effort towards 3D spatial index structure also will be discussed in this paper including several extension of index structure based on the existing R-Tree structure. From the review, some issues of the uncompleted part towards true 3D index structure will be raised.

2 Spatial Indexing

Spatial indexing is a technique to optimize query processing in spatial databases by organizing records in memory space. Query performance is increased especially when dealing with many rows in a table. Number of rows will be reduced once the appropriate indexes are created and after that indexes will be executed for query processing. In traditional database system, data is ordered (or sorted) for efficient searching by using the ordering approach such as B-Tree. But this approach is limited to one dimensional data such as text, numbers, strings and etc. For higher dimensional spatial data, memory space is utilized for organizing and sorting records in the database. According to Rigaux et al (2002), there are two main groups of index structures for spatial databases, which are space-driven structures and data-driven structures. In the next section, we will discuss and illustrate spatial index structures based on these two main groups.

2.1 Space Driven Structures

In space driven structure, objects in 2D space are partitioned into rectangular cells. Objects are mapped according to geometric criterion. The simplest space driven structure, fixed grid was proposed by Bentley and Friedman (1979). The fixed grid structure divides the space to fixed number of cell with an equal size. Each cell represents a block of secondary storage. This means every point in the cell is stored in the same block of secondary storage. The structure of fixed grid also could be implemented by using an array or hashing function. From time to time the structure of fixed grid has been extended to overcome some limitation of original fixed grid. The original fixed grid is limited to non-uniformly distributed data which lead to overflowed cell that being used by others. In this section several space driven structures will be successively presented such as grid file which is an extension of fixed grid file, Quadtree, k-d Tree and Space Filling Curve of Hilberts Curve.

Grid File

Grid file is initially designed from fixed grid for indexing points Nievergelt et al (1984). In the fixed grid, the space is decomposed into rectangular cells and the resulting grid is an $n_x \times n_x$ array of equal size cells. Each one of the cells c is associated with a disk page. Objects or points mapped in cell c are sequentially stored in the page associated with c.

The same method applied in a grid file. Every cell c is associated with one disk page. The difference is when a cell overflows, it will split into two cells. From that split, points are consigned based on the cell they fall into. This causes the size of the partition and cells to adapt to the point distribution. The construction of grid file is best described using Figure 1.

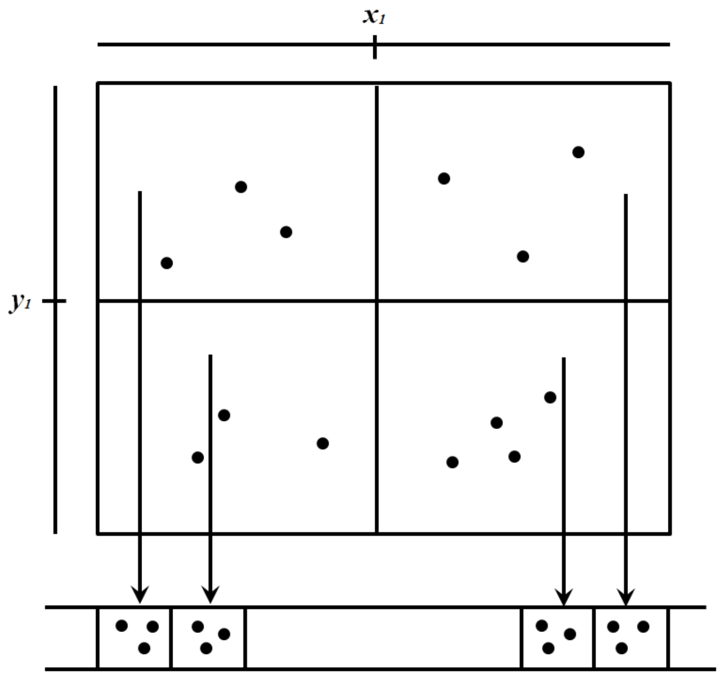


Fig. 1 Grid file for point distribution

Grid files are not limited only to point indexing. They could be utilized for rectangle indexing. In this case, rectangles are mapped by assigning the rectangles to the cell that overlaps the rectangle with the condition of contains, intersect or in contained in it. Grid files have also been extended in different ways as reported by Gaede Gaede et al (1998) such as Extendible Cell or EXCELL Tamminen (1982), the Two-Level Grid File Hinrichs (1985), and the Twin Grid File Hutflesz et al (1988).

The grid file structure seems to meet the requirement in terms of time complexity and space complexity. However, there are several problems that remain. Object duplication of neighbor cells will increase the index size and removing duplicates is expensive when the result size is large. For large datasets, especially large scale urban datasets, the management of this data becomes complex and this will directly degrade the query response time.

Quadtree

The quadtree is a data structure used to subdivide the 2D plane into four quadrants. The quadrants are named North West (NW), North East (NE), South West (SW) and South East (SE). In the Quadtree structure (Figure 2), the search space is recursively decomposed into quadrants until the number of rectangles overlapping each quadrant is less than the page capacity. The Quadtree could be utilized for data representation such as point, area, line and curve.

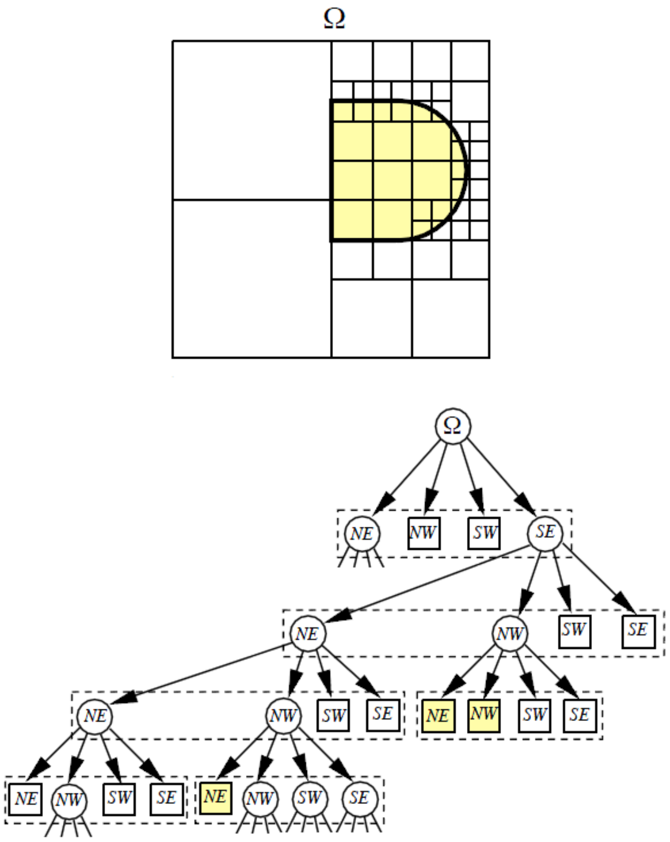


Fig. 2 Quadtree structure taken from Gomes et al (2009)

A lot of applications used Quadtrees for detecting collisions, view frustum culling of terrain data, fractal image analysis and more. In previous research, Quadtrees were applied to binary images Klinger and Dyer (1976), computation of geometric properties Ranade and Shneier (1981), edge enhancement Ranade (1981) and distance transformation Samet (1982). Based on this, several scholars Samet et al (1984), Peuquet (1984), Mark and Lauzon (1985) and Mark (1986) have proposed the use of Quadtree in GIS.

Recent work by Zhang et al (2011), has adopted the Quadtree data structure for large scale geospatial data with a new design of a Quadtree structure termed as Bitplane Quadtree (BQ-Tree). 16-bits of the NASA MODIS image were tested in Zhang et al (2011) with $22,658 \times 15,586$ cells in 190 milliseconds, which is 1.86 billion cells per second. While handling raster data seems promising for BitplaneQuadtree, a limitation on geometry and topology is reported in Mark (1986); Tobler and Chen (1986). These papers show that the Quadtree has been experimented for large areas by partitioning the space into a set of mutually exclusive and collectively exhaustive patches or frames. The concept of the quadrant is applied within the patches of space which is very easy to accommodate with GIS systems. However, from this experiment, the Quadtree does encounter a problem when it was applied on large portions or Earths surface. This problem is due to the problem of tessellated squares on a sphere surface. Thus to overcome this problem, Mark (1986)suggested for continuous or path-connected area of surface the modification concept of Quadtree is necessary. The same problem is also reported by Tobler and Chen (1986). In their experiment of they were using Quadtree for global storage information. From the report, it is mentioned that if one attempts to adapt this technique on ellipsoidal surface, there will be some complications on the data storage. This is due to the facet edges which are curves on the sphere is not coincident with the latitude-longitude lines and sub-facet edges are not even great circles.

k-d Tree

According to Bentley (1975), the k-d Tree stores k-dimensional points in subdivided rectangular parallelepipeds. Each rectangular parallelepiped corresponds to one node of the k-d Tree. The whole region of interest is the root of the tree. The internal nodes are subdivided by rectangles orthogonal to one of the d axes of coordinates (for each level i of the k-d-tree, the axis of coordinates orthogonal to the dividing rectangle is the axis of coordinate x_i) into two rectangular parallelepipeds. The points of the original rectangular parallelepiped that have the x_i coordinate lower than or equal to the x_i coordinate of the dividing rectangle are represented by the left subtree, while the others are represented by the right subtree (see Figure 3). For each node in the k-d Tree, it will consist of records and two pointers. Pointers are either null or point to other nodes. The insertion of a new point will result into the splitting of an existing node into two new nodes. Range searching starts from the root, and then checks whether the stored node overlaps with the right or the left region of the subtree. For the overlapping search region, the procedure is repeated until the leaf level.

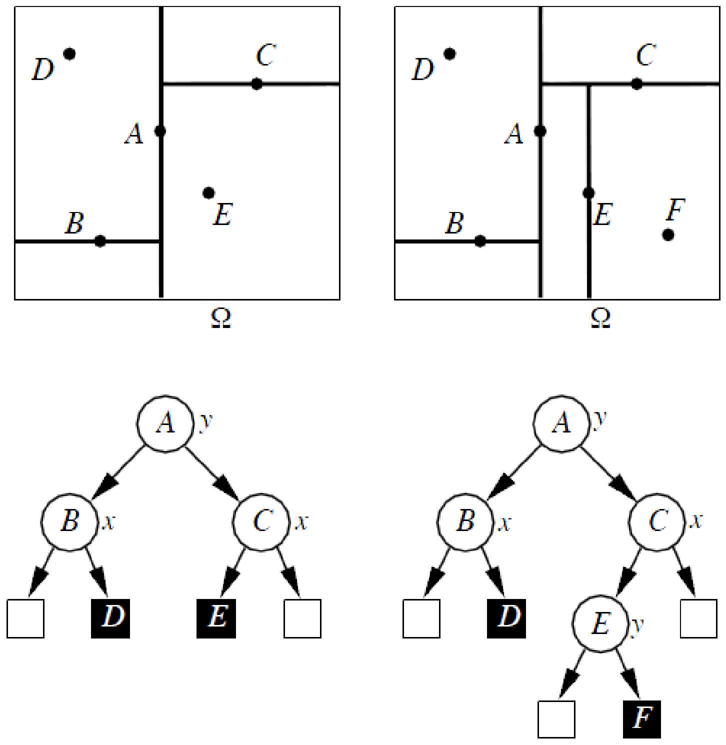


Fig. 3 k-d Tree by Gomes et al (2009)

k-d Trees are efficient in performing nearest neighbor searches such as list of points at distance lower than D from point X or find the nearest points of P from X. When executing such a query, the corresponding leaf node of point X will be detected by traversing the k-d-tree from the root until the leaf. The point stored in the leaf node will be processed and nearby leaf will be scanned. When the distance from X to the leaf is higher than the point found so far, it is the time to stop searching. Algorithm of k-d Tree is the most widely used algorithm for nearest neighbor search Friedman et al (1977). However, the search implementation is ineffective and degrading when space dimensionality is increased Oosterom (1999), Sample et al (2001) and Nene and Nayar (1997). Another problem of k-d Tree mentioned by Bærentzen et al (2002) is that its structure depends on the order of inserted points. In the case of k points require k levels; a searching will took a linear time to complete the process. A solution to this problem is by balancing the tree structure using adaptive k-d Tree. Adaptive k-d Tree splits each set of points in the whole region of interest into two equal regions. By having this division, it is ensure that the adaptive k-d Tree is a balance tree (see Figure 4).However, the balancing enforcement by the adaptive k-d Tree took the flexibility of the structure to insert or delete points. This is due to the prohibitions of standard balancing strategies based on rotation. The rotations could not be used because it would destroy the rule at a given depth of tree. However, most of the recent extension of k-d Tree such as generalized k-d Tree has loosens the rule. Therefore, insertion or point deletions in adaptive generalized k-d Tree are allowed and in the same time satisfy the balancing property. The algorithm of k-d Tree has been improved from time to time to suit applications such as retrieving images from a medical image collection Robinson et al (1996) and mobile robot navigation Li et al (2008).

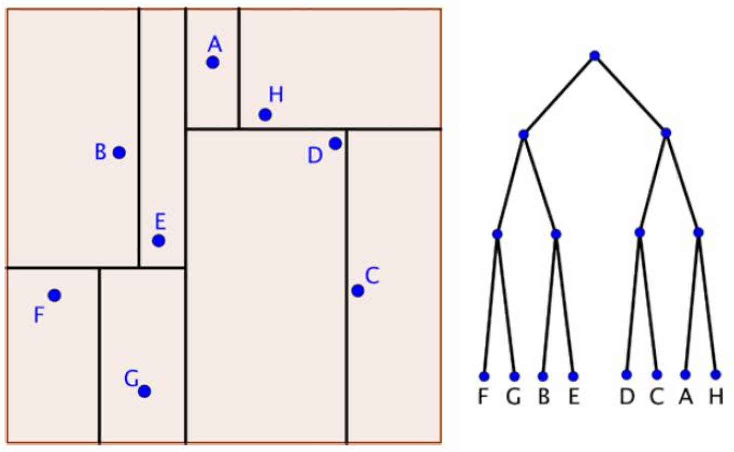


Fig. 4 An Adaptive k-d Tree by Bærentzen et al (2002)

Space filling curve

A space filling curve is a continuous function with domain the unit interval $[0,1]$ and a range that could be lying in a nD topological, uniform or Euclidean space. Space filling curves allow a proximity problem to be reduced from higher-dimensional spaces to the one-dimensional real line. Such a problem usually involves a query search and sorting in one-dimensional space. Many applications can benefit from space filling curves, but a common application of space filling curves is the storage and retrieval of multidimensional data in the database. Proximity problems for which space filling curves have been used regularly, are approximate nearest neighbor search or finding closest points from one location. Until now, there are a lot of variants of space filling curves such as the Hilbert Curve, Dragon Curve, Gosper Curve, Moore Curve, Z-Order curve and many more.

The space filling curve was discovered by Giuseppe Peano in 1890 and called the Peanos 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 unit interval and unit square Thiele and Wos (2002). In 1879, Netto added that any such mapping could not be continuous Bocarra (2007). In 1891, David Hilbert discovered a Hilbert Curve structure which is a simpler variant of the same idea of Peanos Kevin (2008). The Hilbert Curve subdivides squares into smaller squares instead of Peanos, nine smaller squares. Figure 5 shows four iterations of Peanos Curve.

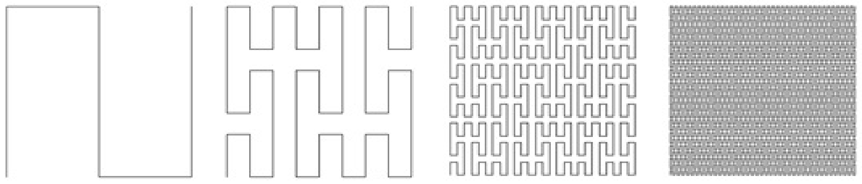


Fig. 5 First four iterations of Peano Curve

When David Hilbert discovered a variation of Peanos Curve, he divided each side of the 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. Hilberts Curve normally starts at the lower left division and ends at the lower right division. According to its movement and process, the Hilbert Curve could be expressed in base 2.

The Hilbert Space Filling Curve could be described as an underlying grid, an $N \times N$ array of cells where $N = 2^n$. Hilberts 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.

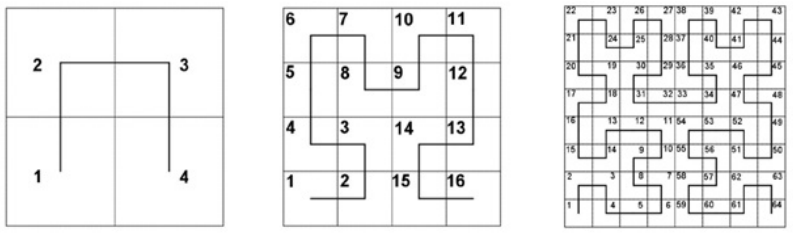


Fig. 6 The first three stages in generating Hilbert Curve

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, the Hilbert Curve has corners at integers ranging from 0 to $2^n - 1$ in both x and y. And if we took a positive direction along the curve the location from (x,y) = (0,0) to ($2^n - 1$, 0).

In a large data management Hilberts Curve responded positively on the performance.Castro and Burns (2007) have experimented with the Hilberts Curve with the visualization of a large datasets. From their finding, it indicates that Hilberts curve produces acceptable quality of mapping and in the same time, achieving faster visualization especially for online visualization of very large data sets. Apart from Hilberts Curve performance, its limitation should be considered in developing applications. For an instance Anders (2009) experimented a data vector with length of 1.8M to demonstrate the use of Hilberts Curve. In this experiment, the data vector is plotted with its whole length condensed to width of plot. The obvious drawback of Hilberts Curve in this experiment is that it is rather hard to relate a position on the plot back to a position of the vector. This limits applicability when absolute positions are of interest but is not an issue if one is interested in judging relative positions, i.e. spacing, homogeneity, etc. Another drawbacks of Hilberts Curve mentioned by Chung et al (2007); Kamata et al (1996) is the representation of Hilberts Curve. Based on its representation some homogeneous segments may be rather long. In their experiments, seven bits are needed for encoding the length one single segment.

2.1.1 Data-Driven Structures

Data Driven structures is a structure that based on partitioning the set of objects and thus adapt to the distribution of these objects. In this section R-Tree data structure has been chosen as an example of data driven structures. The R-Tree structure itself is adapted to the rectangle distribution in the plane which is a concept of data driven structures.

R-Tree

The R-Tree is a balanced tree found by Guttman (1984). The structure of R-Tree is applicable to arbitrary spatial objects by aggregating minimum bounding regions (*mbr*) and organizes the aggregates in a tree structure. Each node in the R-Tree structure corresponds to a disk page. Every leaf node contains an array of leaf entries. And every leaf entry must be a pair (*mbr, objectid*). *mbr* is of the form $(\underline{x_1}\overline{x_1}\cdots\underline{x_d}\overline{x_d})$, where d is the dimension of the search space. Non-leaf nodes contain an array of node entries. According to Guttman (1984) R-Tree structures must satisfy the following properties:

- Every leaf node in the tree contains between m and M index records, where $m \in [0, M/2]$ except for the root.

- Each index record in a leaf node contain entries of the form (*I, tuple-identifier*), where I is the smallest bounding rectangular parallelepiped of the *n*-dimensional data object.
- Non-leaf nodes contain entries of the form (*I, child-pointer*), where *I* is a *n*-dimensional rectangular parallelepiped. Every non-leaf node has between *m* and *M* children unless it is the root.
- For each leaf entry (*I, child-pointer*), *I* is the minimal bounding box or smallest rectangle of object stored in the child node.
- The root node has at least two children unless it is a leaf.
- All leaves appear on at most two different levels. That means the tree is balanced.

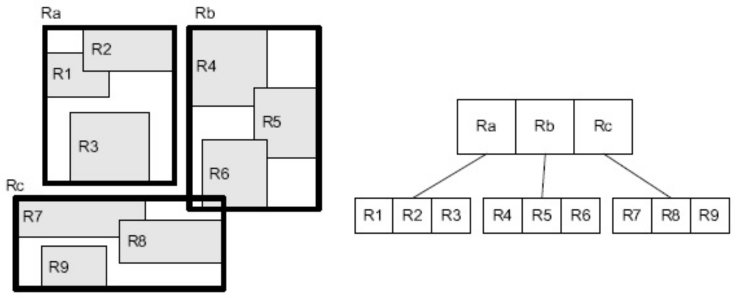


Fig. 7 R-Tree representation from Manolopoulos et al (2006)

Figure 7 illustrates the R-Tree structure. The three minimum bounding rectangles of the root node Ra, Rb and Rc are represented by thick lines. Traditionally, R-Tree performance has been evaluated for range query such as NN (Nearest Neighbor) queries (Roussopoulos et al., 1995) and join query Brinkhoff et al (1993). From the report of Papadopoulos and Manolopoulos (1997)), the R-Tree structure shows the efficiency in answering Nearest Neighbor queries. In real world applications, NN queries are applied in various fields such as pattern recognition, clustering analysis, content based image retrieval and more. Unfortunately, just like other indexing structures, R-Trees have their own limitations due to their poor scaling characteristic. According to Manolopoulos et al (2006), the original R-Tree has two important disadvantages:

1. For a point query, the R-Tree structure may execute an investigation from root to leaf level which leads to performance degradation especially when MBRs overlap.
2. Large rectangles (MBR) may increase the overlap area which leads to performance deterioration during query execution.

These limitations or disadvantages were addressed in research works towards R-Tree variants, such as R^+-Tree by Sellis et al (1987), R*Tree by Beckmann et al

(1990), Hilbert-R Tree by Kamel and Faloutsos (1994) and R k-d Tree by Anandhakumar et al (2010). Every variant is attempting to tackle some limitation of the original R-Tree. For instance, the R^+-Tree variant by Sellis et al (1987) is dedicated to increase search performance for point query by reducing the number of disk access and attempts to have zero overlapping. This variation of R-Tree has been discussed by Hwang et al (2003), Manolopoulos et al (2006) and Balasubramanian and Sugumaran (2012). Several other R-Tree variants were introduced such as the R^+-Tree, R*Tree, Packed R-Tree, Hilbert R-Tree, R*Q Tree and Historical R-Tree. Lakshmi et *al.* 2012 classified the structure of R-Tree variance in three categories: process change, hybridization and extension. The classification is shown in Figure 8.

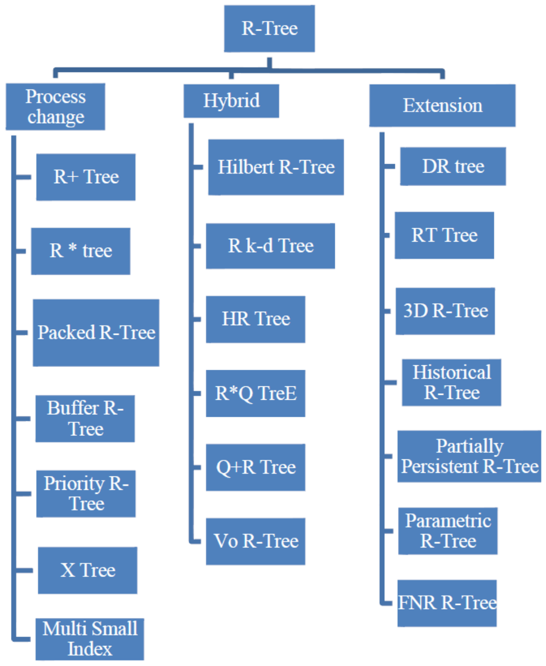


Fig. 8 R-Tree Variants by Balasubramanian and Sugumaran (2012)

The R-Tree structure has been widely used in spatial database management systems due to its efficiency in organizing spatial objects of arbitrary dimension. Besides that, using R-Tree index will save both disk space and time to build and maintain the index. Bottom-up build performance also will be improved. This is due to reducing memory and temporary database space usage. Commercial RDBMSs such as Oracle and Informix, also adopting an R-Tree structure in their spatial database according to Ravada and Sharma (1999); Kothuri et al (2002). Due to this reputation, the R-Tree structure and its variants attracted a lot of attention from researchers and commercial database companies. The simplicity of the structure and its ability to handle spatial objects efficiently are two tempting reasons for incorporating the structure of R-Tree in a spatial database. In modern computing-intensive applications, an efficient method for handling spatial data is of great importance Ravada and Sharma (1999).

3 3D Spatial Indexing

The state-of-the-art of 3D spatial indexing evolved when 3D data are widely used in various fields and applications. For example, spatial professionals can simulate the noise, heat and pollution spreading in virtual cities; architects can use a 3D dataset to design buildings and visualize the sceneries with real textures and utility management companies use 3D spatial databases for the integration of above ground, underground, indoor and outdoor objects. In order to retrieve those data efficiently, a 3D spatial index is required in managing, retrieving and optimizing records in spatial databases. The need of a 3D spatial index for 3D data management has been mentioned in Wang et al (2010); Zhu et al (2007); Deren et al (2004); Arens et al (2005); Zlatanova (2000); Kofler et al (2000).

As we discussed in the previous section, there are many different indexes in handling 2D spatial objects such as k-d Tree, quadtree and grid file. However, apart from the *k*-d tree, they are not suitable to 3D applications. Thus, most of the research on a 3D spatial index is done just by extending the 2D spatial index such as R-Tree to 3D R-Tree and Quadtree to Octree. The transition of 2D case to the 3D case is not straightforward. We could not assume objects in 2D are the same as in 3D space. For example, the geometry of objects in 2D is different when it transforms into 3D. Because of that, the relationships between objects in 2D sometimes are different in 3D. When it comes to spatial indexing, 3D data that was processed based on the expansion of 2D spatial index may affect the data retrieval during queries Gu et al (2011). As we know, a query is very significant to the efficient management of the database especially for large datasets of large scale areas.

For example, in a commercial database management system such as Oracle, 3D R-Tree is implemented when dealing with 3D data Murray (2009); Ravada and Sharma (1999). The concept of 3D R-Tree is not much different from the original R-Tree. What makes it capable of handling 3D objects is its extension to the third dimension of the MBR to the MBV (Minimum Bounding Volume). In the 3D R-Tree, the geometry of candidates that satisfy a query window will be identified through the volume box (MBV), or in other words, a 3D rectangular parallelepiped. Figure 9 shows a MBV in a 3D R-Tree. Just like the original R-Tree, the tree shape is autonomously optimized during data insertion and the leaf nodes are balanced to make sure the performance is stable. However, when the R-Tree is extended into 3D space, the MBVs of sibling nodes tend to frequently overlap, and MBVs among nodes can even contain each other. Overlapping of MBVs is the main reason for the low efficiency of query performance due to multi-path queries. Therefore, using 3D

R-Trees for 3D applications is not successfully promising. It needs to be optimized or modified according to features of 3D space. Several research works such as Zhu et al (2007); Jun and Shengnan (2011); Gong (2011) try to enhance the 3D R-Tree in order to minimize the overlapping among siblings.

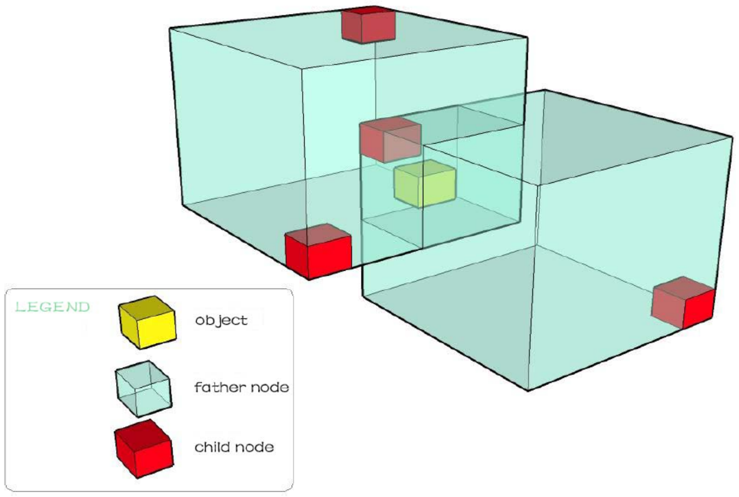


Fig. 9 MBV in R-Tree Structure in Gong et al (2009)

Due to critical overlap of sibling nodes and uneven size of nodes, Zhu et al (2007) attempts to minimize the overlap and optimize the clustering algorithm by introducing *k*-means clustering algorithm to put forward an improved 3D R-Tree. From his experiments, the performance of 3D R-Tree is increased by using an improved algorithm where the overlap is minimized drastically while balancing the volume of nodes. Figure 10 shows a comparison of the original algorithm of 3D R-Tree and this improved algorithm. Whilst, Jun Jun and Shengnan (2011) suggested splitting irregular 3D objects into several small parts in order to improve the overlapping node issue. In 2009, Gong et al (2009) reported two sub-algorithms of R-Tree that make the 3D R-Tree to be well-shaped. The two sub-algorithms are namely node-choosing and node-split procedures. Node-choosing is dedicated to finding a leaf node for new inserted objects and node-split is responsible for splitting overflown nodes into smaller ones. Using his algorithm, fewer R-Tree nodes are accessed in memory, which increases the query performance. Experimental results of Gongs work are shown in Figure 11. From the experiment, the improved algorithm is far better over the original R-Tree and R*-Tree.

Although improving static 3D R-Trees increases the efficiency in query performance, there are several factors that could not be ignored and must be considered in the implementation of spatial indexes. 3D R-Trees for different applications will result in different performances. For example, applications with large amounts of data still face low retrieval efficiency even when improved 3D R-Tree is applied.

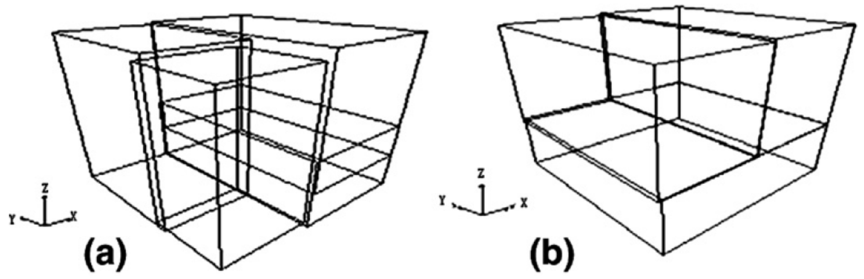
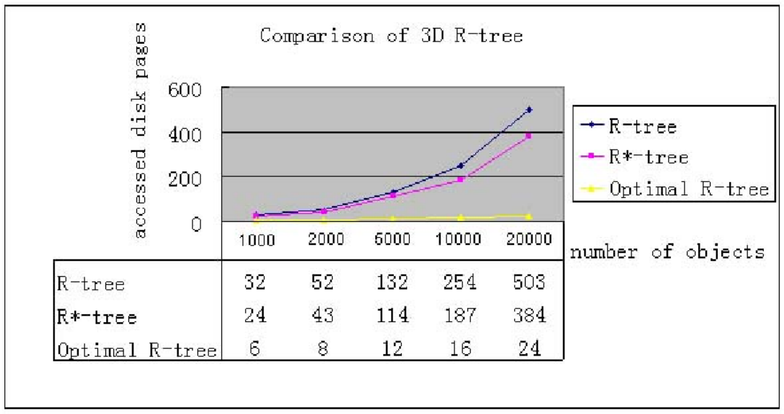


Fig. 10 Comparison of 3D R-Tree methods (a) Original algorithm of 3D R-Tree and (b) Improved algorithm in Zhu et al (2007)



	1000	2000	6000	10000	20000
R-tree	32	52	132	254	503
R*-tree	24	43	114	187	384
Optimal R-tree	6	8	12	16	24

Fig. 11 Improved 3D R-Tree algorithm over Original or Classic R-Tree and R*-Tree by Gong et al (2009)

According to Gu et al (2011), using a single index structure for 3D spatial data is not an ideal method. Thus, the combination of several index structures is proposed by many scholars (see Wang et al (2010); Gu et al (2011); Song et al (2006); He et al (2009)). A combination of two or more index structure is also known as hybrid indexing. By combining each other limitations and strengths, index structures in hybrid indexing would push the limits of data retrieval efficiency. Some examples of hybrid indexing of 3D objects are ORSI Gu et al (2011) and LOD_SKDR Tree He et al (2009). ORSI Gu et al (2011) is built based on an Octree and a R-Tree index structures. The principle of ORSI is:

1. The area of interest is divided by Octree. Then the first level index is established by setting up a threshold for subspace k, where k is the number of objects in sub-space S. When the number of objects in subspace S is greater than k, Octree division is repeated.

2. Secondary index is build based on R-tree. Each subspace is pointed by one pointer. If there is no R-Tree in the subspace, the pointer is empty. In an R-Tree node, adjacent or close spatial objects are gathered to improve query efficiency.

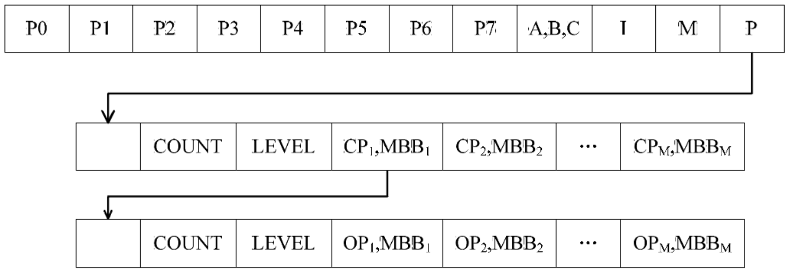


Fig. 12 ORSI structure by Gu et al (2011)

Figure 12 shows the ORSI index structure Gu et al (2011). Based on the operational performance of ORSI on R-Tree, average access on inserting, deleting and retrieving indicates that the performance of ORSI is overall better than classic R-Trees. Figure 13 shows the comparison of ORSI and classic R-Trees.

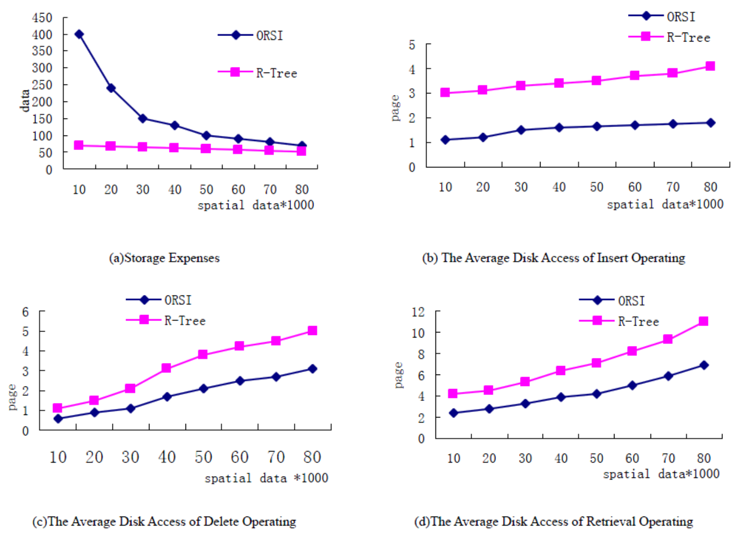


Fig. 13 Comparison of ORSI and R-Tree in Gu et al (2011)

As mentioned in previous paragraph, different structures may be used for different applications. For the applications such as underground utility management, geological analysis and other underground space entity object may need a different index structure. Any current index structure could not effectively deal with large scale complicated geological data. The index structure of multi-level spatial index or BCSR-Tree in He et al (2009) and 3D RR-Tree in Wang et al (2010) is dedicated to handling the retrieval of 3D underground objects. For large scale underground data, Liu et al (2010) proposed LOD_SKDR Tree structure for management of large scale database. The LOD_SKDR Tree is a combination of SKD-Tree, R-Tree and information of object's LOD (Level of Detail). According to Liu et al (2010) the LOD_SKDR Tree utilizes the object's number as a bifurcation bound target to improve the balance of SKD-Tree. While in the meantime, R-Tree and texture scheduling of SKD-Tree content are restricted to reduce time on insertion or deletion. A LOD object model is also used to improve real time rendering in the 3D scene. Figure 14 shows a complicated 3D underground scenario - utility pipeline in urban areas.

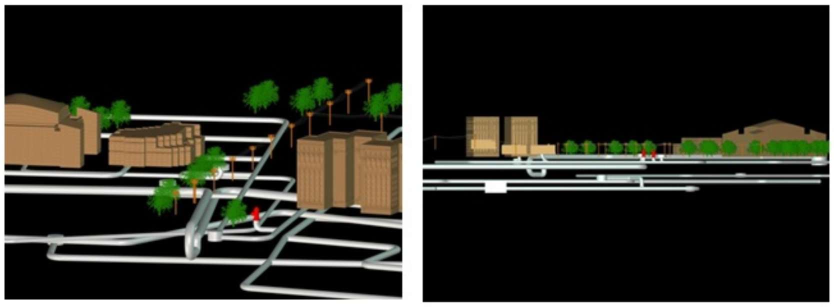


Fig. 14 Complicated Underground Scenes: Underground Pipeline in Urban Area

In managing large raw datasets acquired using LiDAR, spatial indexing plays an important role by managing points in spatial databases. With the rapid development of LiDAR acquisition, a variety of spatial data will continue to be accumulated. The aim of database management in manipulating those data is by having efficient high-quality rendering of point clouds. Many scholars and researchers attempt to investigate suitable indexing data structures for handling these huge massive point clouds. The hybrid indexing approach on LiDAR data proposed in Liu et al (2008) is specialized for point cloud visualization. In their work, Liu et *al.* 2008 integrate a k-d Tree and an Octree for a 3,200,079 point cloud on display time. Wang and Guo (2012) has presented the QMBB spatial index which has both a 2D and 3D index structures. QMBB Tree represents Quadtree segmentation as 2D index based on a 2D range image. Each one of the corresponding point image of MBR is then calculated as a MBV in 3D space. Based on the result, the rendering speed is up to 10 million points per second. There are more research developments of hybrid indexes for point clouds such as Gong (2011); Koo and Shin (2005); Schön et al

(2009, 2013). These researches proved the capability of hybrid indexing in efficient data management for point clouds.

4 Conclusion

In this paper, we presented several index structure based on space driven structure and data driven structure. We briefly described their structure and their implementation along with large volume data application such as, large urban data management and point clouds data management. Limitation and advantages of each structure are also revealed. Based on the several methods of 2D and 3D spatial indexing, it is clearly shown that spatial indexing is capable in handling a large amount of dataset especially for urban data management. However, the structure of each spatial indexing is unique and only suits for certain situation or case. For an instance, continuous area of surface is not suitable for tessellated square based index structure such as Quadtree. Among all the data structure R-Tree is the most widely used spatial index structure in commercial database which serving many applications and multidimensional data. In this paper, several extension of R-Tree index structure based on 3D has been discussed in the last section of this paper. It could be seen that most of the structure are developed and tested for point dataset and lack of the example and effort for volume based data. Based on the discussions and some example given, it indicates that there is more space for 3D spatial index to be improved especially in the aspect of complex and large data management. A survey on spatial index structure in this paper may give some view and idea on future development of spatial indexing for large data management.

Acknowledgements Major funding for this research was provided by the Ministry of Higher Education Malaysia and partially funded by the Land Surveyors Board of Malaysia. This research was done while the third author was a Visiting Full Professor at the Faculty of Geoinformation and Real Estate at Universiti Teknologi Malaysia, for which the third author is very thankful.

References

Anandhakumar P, Priyadarshini J, Monisha C, Sugirtha K, Raghavan S (2010) Location based hybrid indexing structure - r k-d tree. In: 2010 First International Conference on Integrated Intelligent Computing (ICIIC), pp 140–145, DOI 10.1109/iciic.2010.56

Anders S (2009) Visualization of genomic data with the hilbert curve. Bioinformatics 25(10):1231–1235, DOI 10.1093/bioinformatics/btp152, URL `http://bioinformatics.oxfordjournals.org/content/25/10/1231.abstract`, `http://bioinformatics.oxfordjournals.org/content/25/10/1231.full.pdf+html`

Arens C, Stoter J, van Oosterom P (2005) Modelling 3d spatial objects in a geo-dbms using a 3D primitive. Computers & Geosciences 31(2):165–177, DOI http://dx.doi.org/10.1016/j.cageo.2004.05.013, URL `http://www.sciencedirect.com/science/article/pii/S009830040400192X`

Balasubramanian L, Sugumaran M (2012) Article: A state-of-art in R-Tree variants for spatial indexing. International Journal of Computer Applications 42(20):35–41, published by Foundation of Computer Science, New York, USA

Bærentzen J, Gravesen J, Anton F, Aanæs H (2012) Spatial Data Indexing and Point Location. In: Guide to Computational Geometry Processing, Springer, London, pp 213–225, DOI 10.1007/978-1-4471-4075-7_12, URL `http://dx.doi.org/10.1007/978-1-4471-4075-7_12`

Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The R*-Tree: an efficient and robust access method for points and rectangles. SIGMOD Rec 19(2):322–331

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Bentley JL, Friedman JH (1979) Data structures for range searching. ACM Comput Surv 11(4):397–409, DOI 10.1145/356789.356797, URL `http://doi.acm.org/10.1145/356789.356797`

Bocarra N (2007) Lindenmayer Systems. In: Essentials of Mathematica, vol 1, Springer New York, pp 407–416, DOI 10.1007/978-0-387-49514-9_23

Brinkhoff T, Kriegel HP, Seeger B (1993) Efficient processing of spatial joins using r-trees. SIGMOD Rec 22(2):237–246

Castro J, Burns S (2007) Online Data Visualization of Multidimensional Databases Using the Hilbert SpaceFilling Curve, Lecture Notes in Computer Science, vol 4370, Springer Berlin Heidelberg, chap 9, pp 92–109

Chung KL, Huang YL, Liu YW (2007) Efficient algorithms for coding hilbert curve of arbitrary-sized image and application to window query. Information Sciences 177(10):2130 – 2151, DOI http://dx.doi.org/10.1016/j.ins.2006.12.003, URL `http://www.sciencedirect.com/science/article/pii/S0020025506003732`

Deren L, Qing Z, Qiang L, Peng X (2004) From 2d and 3d gis for cybercity. Geospatial Information Science 7(1):1–5

Friedman JH, Bentley JL, Finkel RA (1977) An algorithm for finding best matches in logarithmic expected time. ACM Trans Math Softw 3(3):209–226, DOI 10.1145/355744.355745, URL `http://doi.acm.org/10.1145/355744.355745`

Gaede V, G Oliver (1998) Multidimensional access methods. ACM Comput Surv 30(2):170–231

Gomes A.J.P., Voiculescu, Irina, Jorge, Joaquim, Wyvill, Brian, Galbraith, Callum (2009) 2 - Spatial Data Structures. In: Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms, Springer, London, pp 41 – 62, DOI 10.1007/978-1-84882-406-5_2, URL `http://dx.doi.org/10.1007/978-1-84882-406-5_2`

Gong J, Ke S, Li X, Qi S (2009) A hybrid 3D spatial access method based on quadtrees and R-trees for globe data. In: International Symposium on Spatial Analysis, Spatial-Temporal Data Modeling, and Data Mining, DOI 10.1117/12.837594, URL http://dx.doi.org/10.1117/12.837594

Gong J, Zhu Q, Zhang H, Li X, Zhou D (2011) An adaptive control method of lods for 3D scene based on r-tree index. Acta Geodaetica et Cartographica Sinica (Issue 4):Page 531–534

Gu W, Wang J, Shi H, Liu Y (2011) Research on a hybrid spatial index structure. Journal of Computational Information Systems 7(11):3972–3978

Guttman A (1984) R-trees: a dynamic index structure for spatial searching. SIGMOD Rec 14(2):47–57

He Z, Liu G, Weng Z, Wu C (2009) Three-dimensional spatial indexing method of complicated geological scene. In: 17th International Conference on Geoinformatics, 2009, IEEE, 1, pp 1–4

Hinrichs K (1985) Implementation of the grid file: Design concepts and experience. BIT Numerical Mathematics 25(4):569–592

Hutflesz A, Six HW, Widmayer P (1988) Twin grid files: space optimizing access schemes. SIGMOD Rec 17(3):183–190, DOI 10.1145/971701.50222, URL http://doi.acm.org/10.1145/971701.50222

Hwang S, Kwon K, Cha S, Lee B (2003) Performance Evaluation of Main-Memory R-tree Variants, Lecture Notes in Computer Science, vol 2750, Springer Berlin Heidelberg, chap 2, pp 10–27

Jun G, Shengnan K (2011) 3D spatial query implementation method based on r-tree. In: International Conference on Remote Sensing, Environment and Transportation Engineering (RSETE), 2011, IEEE, 1, pp 2828–2831

Kamata S, Niimi M, Kawaguchi E (1996) A gray image compression using a hilbert scan. In: Proceedings of the 13th International Conference on Pattern Recognition, 1996., 3, pp 905–909, DOI 10.1109/icpr.1996.547299

Kamel I, Faloutsos C (1994) Hilbert R-Tree: An improved R-Tree using fractals

Kevin B (2008) Organizing point sets. PhD thesis, Freie Universitat Berlin

Klinger A, Dyer CR (1976) Experiments on picture representation using regular decomposition. Computer Graphics and Image Processing 5(1):68–105, DOI http://dx.doi.org/10.1016/S0146-664X(76)80006-8, URL http://www.sciencedirect.com/science/article/pii/S0146664X76800068

Koo YM, Shin BS (2005) An Efficient Point Rendering Using Octree and Texture Lookup, Lecture Notes in Computer Science, vol 3482, Springer Berlin Heidelberg, chap 127, pp 1187–1196

Kothuri RKV, Ravada S, Abugov D (2002) Quadtree and r-tree indexes in oracle spatial: a comparison using gis data

Kun Z, Xiu-guo L, Hui Y (2005) Research on Spatial Index Structure of LOD-OR Tree in 3D GIS. Bulletin of Surveying and Mapping 5(1):27–29

Li MH, Hong BR, Cai ZS, Piao SH, Huang QC (2008) Novel indoor mobile robot navigation using monocular vision. Engineering Applications of Artificial Intelligence 21(3):485 – 497, DOI http://dx.doi.org/10.1016/j.engappai.2007.05.

003, URL http://www.sciencedirect.com/science/article/pii/S0952197607000607

Liu H, Huang Z, Zhan Q, Lin P (2008) A database approach to very large LiDAR data management. In: International Society of Photogrammetry and Remote Sensing (ISPRS), ISPRS Beijing XXXVII:463–468

Liu Y, Liu G, He Z (2010) Spatial index technology for multi-scale and large scale spatial data. In: 18th International Conference on Geoinformatics, 2010, IEEE, 1, pp 1–4

Kofler M., Gervautz M. (2000) R-trees for organizing and visualizing 3d gis databases. The Journal of Visualization and Computer Animation 11(3):129–143, DOI doi:10.1002/1099-1778(200007)11:3\&\#060;129::AID-VIS227\&\#062;3.0.CO;2-T, URL http://www.ingentaconnect.com/content/jws/vis/2000/00000011/00000003/art00227

Manolopoulos Y, Nanopoulos A, Papadopoulos AN, Theodoridis Y (2006) R-Trees: theory and applications. Springer

Mark DM (1986) The use of quadtrees in geographic information systems and spatial data handling. Hardware, Data Capture and Management Techniques, Auto-Carto London 1:517–526

Mark DM, Lauzon JP (1985) Approaches for quadtree-based geographic information systems at continental or global scales. In: Proceedings of Auto-Carto, 7, pp 355–365

Mohamad Yusoff I, Uznir Ujang M, Abdul Rahman A (2010) 3D Volumetric Soft Geo-objects for Dynamic Urban Runoff Modeling, Developments in 3D Geo-Information Sciences, vol 1, Springer Berlin Heidelberg, chap 18, pp 200–219

Murray C (2009) Oracle spatial developer's guide, 11g release 1 (11.1)

Nene SA, Nayar SK (1997) A simple algorithm for nearest neighbor search in high dimensions. Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(9):989–1003

Nievergelt J, Hinterberger H, Sevcik K.C. (1984) The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Trans. Database Syst 9(1):38–71, DOI 10.1145/348.318586, URL http://doi.acm.org/10.1145/348.318586

Oosterom PV (1999) Spatial access methods edited by longley, goodchild, maguire en rhind, john wileypages385-400 (vol.1), 1999. Geographical information systems 1:385–400

Papadopoulos A, Manolopoulos Y (1997) Performance of nearest neighbor queries in r-trees. In: Database Theory ICDT '97, Lecture Notes in Computer Science, vol 1186, Springer Berlin Heidelberg, pp 394–408, DOI 10.1007/3-540-62222-5_59, URL http://dx.doi.org/10.1007/3-540-62222-5_59

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

Peuquet DJ (1984) Data structures for a knowledge-based geographic information system. In: Proc. Spatial Data Handling

Ranade S (1981) Use of quadtrees for edge enhancement. IEEE Transactions on Systems Man and Cybernetics 11:370–373

Ranade S, Shneier M (1981) Using quadtrees to smooth images. IEEE Transactions on Systems Man and Cybernetics 11:373–376

Ravada S, Sharma J (1999) Oracle8i Spatial: Experiences with Extensible Databases, Lecture Notes in Computer Science, vol 1651, Springer Berlin Heidelberg, chap 21, pp 355–359

Rigaux P, Scholl M, Voisard A (2002) 6 - spatial access methods. In: Spatial Databases, Morgan Kaufmann, San Francisco, pp 201–266, DOI http://dx.doi.org/10.1016/B978-155860588-6/50008-7, URL `http://www.sciencedirect.com/science/article/pii/B9781558605886500087`

Robinson G, Tagare H, Duncan J, Jaffe C (1996) Medical image collection indexing: Shape-based retrieval using kd-trees. Computerized Medical Imaging and Graphics 20(4):209–217, DOI http://dx.doi.org/10.1016/S0895-6111(96)00014-6, URL `http://www.sciencedirect.com/science/article/pii/S0895611196000146`

Roussopoulos N, Kelley S, Vincent E (1995) Nearest neighbor queries. SIGMOD Rec 24(2):71–79

Samet H (1982) Distance transform for images represented by quadtrees. Pattern Analysis and Machine Intelligence, IEEE Transactions on (3):298–303

Samet H, Rosenfeld A, Shaffer CA, Webber RE (1984) A geographic information system using quadtrees. Pattern Recognition 17(6):647–656, DOI http://dx.doi.org/10.1016/0031-3203(84)90018-9, URL `http://www.sciencedirect.com/science/article/pii/0031320384900189`

Sample N, Haines M, Arnold M, Purcell T (2001) Optimizing search strategies in kd trees. In: Fifth WSES/IEEE World Multiconference on Circuits, Systems, Communications & Computers (CSCC 2001)

Schön B, Bertolotto M, Laefer DF, Morrish S (2009) Storage, manipulation, and visualization of lidar data. In: Proceedings of the 3rd ISPRS International Workshop 3D-ARCH 2009:" 3D Virtual Reconstruction and Visualization of Complex Architectures", International Society of Photogrammetry and Remote Sensing, XXXVIII-5/W1

Schön B, Mosa ASM, Laefer DF, Bertolotto M (2013) Octree-based indexing for 3d pointclouds within an oracle spatial dbms. Computers & Geosciences 51(0):430–438

Sellis TK, Roussopoulos N, Faloutsos C (1987) The R+-Tree: A dynamic index for multi-dimensional objects

Sharkawi KH, Ujang MU, Abdul-Rahman A (2008) 3D navigation system for virtual reality based on 3D game engine. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2008, ISPRS, XXXVII, pp 513–518

Song Xy, Zhou Xw, Wang Yh (2006) Research on spatial index structure of hybrid tree in 3D GIS. Journal of Shenyang Jianzhu University (Natural Science) 3:030

Tamminen M (1982) The extendible cell method for closest point problems. BIT Numerical Mathematics 22(1):27–41

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

Tobler W, Chen Zt (1986) A quadtree for global information storage. Geographical Analysis 18(4):360–371

Ujang U, Abdul-Rahman A (2013) Temporal Three-Dimensional Ontology for Geographical Information Science (GIS) - A Review. Journal of Geographic Information System 5(3):314–323, DOI 10.4236/jgis.2013.53030, URL `http://www.scirp.org/journal/PaperDownload.aspx?paperID=33290`

Wang Y, Guo M (2012) An integrated spatial indexing of huge point image model. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIX-B4:397–401, DOI 10.5194/isprsarchives-XXXIX-B4-397-2012, URL `http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B4/397/2012/`

Wang Y, Sheng Y, Zhou L, Guo F, Zhao L (2010) An underground space object-oriented three-dimensional hybrid spatial indexing method. In: 18th International Conference on Geoinformatics, 2010, IEEE, 1, pp 1–5

Zhang J, You S, Gruenwald L (2011) Parallel quadtree coding of large-scale raster geospatial data on gpgpus. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, New York, NY, USA, GIS '11, pp 457–460, DOI 10.1145/2093973.2094047, URL `http://doi.acm.org/10.1145/2093973.2094047`

Zhu Q, Gong J, Zhang Y (2007) An efficient 3D r-tree spatial index method for virtual geographic environments. ISPRS Journal of Photogrammetry and Remote Sensing 62(3):217–224, DOI http://dx.doi.org/10.1016/j.isprsjprs.2007.05.007, URL `http://www.sciencedirect.com/science/article/pii/S0924271607000524`

Zlatanova S (2000) 3D GIS for urban development. International Inst. for Aerospace Survey and Earth Sciences (ITC)