Available online at www.sciencedirect.com
ScienceDirect
ISPRS Journal of Photogrammetry & Remote Sensing 62 (2007) 217 – 224
PHOTOGRAMMETRY & REMOTE SENSING
www.elsevier.com/locate/isprsjprs

An efficient 3D R-tree spatial index method for virtual geographic environments

Qing Zhu [a,*], Jun Gong [b], Yeting Zhang [a]

[a] *State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan 430079, PR China*

[b] *The Key Lab of Poyang Lake Ecological Environment and Resource Development, Jiangxi Normal University, 99 Zhiyang Road, Nanchang 330022, PR China*

Received 1 March 2006; received in revised form 5 May 2007; accepted 10 May 2007
Available online 19 June 2007

Abstract

A three-dimensional (3D) spatial index is required for real time applications of integrated organization and management in virtual geographic environments of above ground, underground, indoor and outdoor objects. Being one of the most promising methods, the R-tree spatial index has been paid increasing attention in 3D geospatial database management. Since the existing R-tree methods are usually limited by their weakness of low efficiency, due to the critical overlap of sibling nodes and the uneven size of nodes, this paper introduces the *k*-means clustering method and employs the 3D overlap volume, 3D coverage volume and the minimum bounding box shape value of nodes as the integrative grouping criteria. A new spatial cluster grouping algorithm and R-tree insertion algorithm is then proposed. Experimental analysis on comparative performance of spatial indexing shows that by the new method the overlap of R-tree sibling nodes is minimized drastically and a balance in the volumes of the nodes is maintained.
© 2007 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS). Published by Elsevier B.V. All rights reserved.

Keywords: Virtual geographic environments; 3D spatial index; R-tree; Spatial cluster grouping

1. Introduction

As the most important geographic information communication tool and human-computer interface, the virtual geographic environments (VGEs) provide the augmentation of sensory reality, and open up new ways for us to comprehend the complicated real world (Lin and Zhu, 2005). For example, scientists can simulate the noise, heat and sunlight spreading in big virtual cities; telecommunication companies can use three-dimensional (3D) data to calculate the wave propagation in urban environments; and architects can design new buildings and visualize the resulting scenery based on photo-realistic models of existing buildings, etc. The emerging 3D spatial database real-time applications (interactive visualization, spatial query and analysis) in VGEs, which integrate the representation of the many above ground, underground, indoor and outdoor objects, require a true 3D spatial index in order to be able to retrieve the data efficiently. A spatial index is a structure supporting the spatial range queries by efficiently providing the addresses of the requested features. Since the commercial geospatial database management systems (Geo-DBMS) mainly support a 2D spatial index, the 3D spatial

* Corresponding author. Tel.: +86 27 68778322; fax: +86 27 68778969.
E-mail addresses: zhuq66@263.net (Q. Zhu), gongjunbox@163.com (J. Gong).

information of objects has therefore to be stored in the 2D Geo-DBMS with simple 3D extensions of the 2D spatial index (Arens et al., 2005). However, a 3D spatial database is not a simple extension from 2D to 3D space, and many new data types and spatial relations emerge. In a massive 3D spatial database, the 3D spatial index and spatial queries are very significant to the efficient management of the database, as well as to its real time applications.

Although there exist many different spatial indices such as *k–d*-tree and cell-tree, etc., they are not however suited to 3D applications. Most research on 3D spatial indices is involved in the 3D extension of Quad-Tree and R-tree, such as Octree and 3D R-tree. The design ideas of these 3D spatial indices are based on the strategy of secondary storage access. That is, the spatial index is stored out of the core memory. For data retrieving, the index node is extracted first, and then the real spatial objects related to the index node are accessed. One node of the index corresponds to a disc page in the secondary storage in order to access node information quickly. In the classical Octree, however, one object may be stored in multiple nodes, which results in high maintenance costs and low space utility. Moreover, Octree is a kind of space-driven index structure, and the scope of the database must be planned in advance. It is not able to dynamically adjust the tree structure according to the actual object layout. As a result, the tree depth is high where there are many objects, and this also results in unstable query performance. Many more recursive operations are involved, which makes the application difficult to optimize. Even the Octree is considered as a simpler data structure for a 3D spatial index since it is the 3D variant of Quad-Tree and its structure is regular (Gaede and Günther, 1998). However, the major argument in favour of 3D spatial index is the support for overlapping objects, which represent geographical data more closely.

Based on the idea of B-tree, Guttman put forward R-tree in 1984, which is height-balanced (Guttman, 1984). R-tree is the extension of a one-dimensional data index B-tree for multi-dimensional query and is a spatial index of depth balance, whose root node is the same distance from all leaf nodes. This feature means that R-tree can maintain a stable performance and high space utilization. For instance, as the space query accesses only part of the index data, it is not necessary to load all index data into the main memory. Therefore, R-tree is considered as one of the most promising 3D spatial indices.

2. From R-tree to 3D R-tree

Theoretically speaking, 3D R-tree would inherit the features of R-tree, and is a dynamic structure. The scope of the database can be adjusted, while the tree shape is autonomously optimized during the insertion of objects. The depth of all leaf nodes is then balanced and the query performance is stable. In other words, R-tree possesses the feature of spatial proximity. In an ideal case, the neighbouring objects should be in the same nodes or sibling nodes, the minimum bounding rectangle (MBR) of sibling nodes is different, and the overlap is then minimized. However, when the index expands from 2D to 3D, because of the great size and shape diversity of different objects in 3D space, the minimum bounding box (MBB) of sibling nodes will frequently overlap, and the MBBs of nodes can even contain each other. In order to adequately take into consideration the principle of 3D spatial proximity, the better space proximity of R-tree is therefore the key to 3D spatial indexing. 3D spatial clustering and the corresponding 3D R-tree indices are required in order to minimize the overlap among the sibling nodes and to balance the shape and size of nodes. Proximal objects in 3D space cluster together in the same nodes or proximal sibling nodes. Furthermore, its tree-shape hierarchical structure makes it easy to transit rapidly from the whole to the local, which can accelerate full 3D spatial queries.

Recently, more and more research has focused on the 3D spatial index for large-scale 3D city modelling. Kofler attempted to combine the R-tree index with LOD (level of detail), and put forward the LOD-R-tree method in which the level of R-tree is regarded as the level of LOD representation (Kofler, 1998). The basic idea of LOD-R-tree is that the depth of tree and the MBR of each node are predefined and each spatial object is then inserted into the corresponding node; however, it is obviously still a 2D R-tree. Zlatanova tried to find a similar way of uniting R-tree with LOD, and brought forward three kinds of grouping methods to group objects that are similar in the aspects of location and shape, which takes the altitude factor into consideration (Zlatanova, 2000). These methods maybe useful to the above ground and outdoor city models, but there are still difficulties with true 3D city models, such as the consideration of both underground and indoor objects, as well as the component models of complicated buildings.

For dynamic indexing as well as R-tree construction, both insertion and deletion are important basic operations. Of course, insertion is more critical to the R-tree construction procedure in complicated 3D space. The insertion of an object would result in the splitting of the R-tree node, and cluster grouping is usually used to support node splitting and node optimization. In classical R-tree, if two objects are in two different nodes, the two objects cannot be adjusted later into one node regardless of the layout of spatial objects. The classical R-tree follows the principle of minimum area, ignoring other factors such as overlap. In order to realize the cluster and reduce the overlap, R^+-tree allows one object to exist in multiple nodes, so the space cost is high (Sellis, 1987). R*-tree realizes rational clusters by synthetically introducing coverage, overlap and periphery (Beckmann, 1990). However, how to combine the three factors is unknown in 3D space. Hilbert R-tree manages to map the high-dimensional objects to one-dimensional space by Hilbert value, but the mapping from multi-dimensional to one-dimensional does not work well in all conditions, especially in 3D space (Kamel and Faloutsos, 1994). cR-tree applies a clustering algorithm such as *k*-means to realize node split, and obviously replacing a two-way split with a multi-way split is more reasonable (Brakatsoulas et al., 2002). With respect to point data, original *k*-means is good, but it cannot work well with spatially extended objects. Where the silhouette coefficient is considered as a measure to find the clusters (introduced from the field of data mining for point data), it does not adapt to the spatial data with various spatial extensions. Therefore, a new measure needs to be developed for various 3D spatial objects in VGEs.

A new 3D R-tree algorithm based on 3D spatial cluster grouping is proposed in Section 3. Section 4 illustrates the results of experimental analysis. Finally, a few concluding remarks are presented in Section 5.

3. 3D R-tree algorithm based on 3D spatial cluster grouping

Aiming at the problems of serious overlap and unbalanced volume of nodes, this paper first proposes a new integrative grouping criterion concerned with the 3D overlap, 3D coverage and MBB shape value of nodes. Then the *k*-means algorithm is employed to improve the 3D spatial cluster grouping and inserting operation of 3D R-tree.

3.1. Integrative grouping criterion

For a 3D spatial object set $S=\{P_1, P_2,\ldots, Pn\}$, there are clustered group sets S_i, $i=1,\ldots, k$.

The integrative grouping criterion value W can be calculated using Eq. (1), in which the three factors have the same influencing weight. Of course, depending on different requirements, we can assign different weights to these three factors. For example, when the minimum of overlap volume is the most important factor, then its weight could be made the largest.

$$W = \sum_{i=}^{k-1}\sum_{j=i+1}^{k} \mathrm{Overlap}(S_i,S_j) + \sum_{i=1}^{k} \mathrm{Coverage}(S_i) + \sum_{i=1}^{k} \mathrm{Shape}(S_i). \qquad (1)$$

Where, the MBB range of S_i is from ($\min x_i$, $\min y_i$, $\min z_i$) to ($\max x_i$, $\max y_i$, $\max z_i$), the MBB centroid of S_i is $((\min x_i+\max x_i)/2.0, (\min y_i+\max y_i)/2.0, (\min z_i+\max z_i)/2.0)$, the MBB volume of S_i is $\mathrm{Coverage}(S_i)=(\max x_i-\min x_i)*(\max y_i-\min y_i)*(\max z_i-\min z_i)$, the MBB shape value of S_i is $\mathrm{Shape}(S_i)=((\max x_i-\min x_i+\max y_i-\min y_i+\max z_i-\min z_i)/3.0)^3$, the overlap volume between S_i and S_j is $\mathrm{Overlap}(S_i, S_j)$. Min($x$,$y$) is the minimum of x and y coordinates. Max(x,y) is the maximum of x and y coordinates.

When, $A=\mathrm{Max}(\min x_i, \min x_j)$, $B=\mathrm{Max}(\min y_i, \min y_j)$, $C=\mathrm{Max}(\min z_i, \min z_j)$, $D=\mathrm{Min}(\max x_i, \max x_j)$, $E=\mathrm{Min}(\max y_i, \max y_j)$, $F=\mathrm{Min}(\max z_i, \max z_j)$.

If $A \geq D$ or $B \geq E$ or $C \geq F$,

$$\mathrm{Overlap}(S_i,S_j) = 0.0.$$

Otherwise,

$$\mathrm{Overlap}(S_i,S_j) = (D-A)*(E-B)*(F-C).$$

According to the Arithmetic–Geometric Means inequality (Cauchy proof):

$$\text{when } x>0, y>0 \text{ and } z>0, \quad ((x+y+z)/3.0)^3 \geqq x \times y \times z.$$

When volumes are the same, the shape value is smaller and the lengths of MBB edges in the three axes are closer, which means more regular shapes and more balanced volumes. Therefore, the smaller the W value, the better the 3D spatial cluster grouping results.

3.2. 3D spatial cluster grouping

The 3D spatial cluster grouping operation includes two steps: the node splitting and the optimization among nodes. Fig. 1 illustrates a typical grouping result.

One of the reasons for serious overlaps of 3D R-tree nodes is that only the optimization of two nodes is considered in the existing algorithms. For example, node splitting means that an overflowing node is divided into two small nodes. As shown in Fig. 1, the wire frame box denotes the node that needs to be split, and solid boxes denote the child nodes, and in this example it is obvious

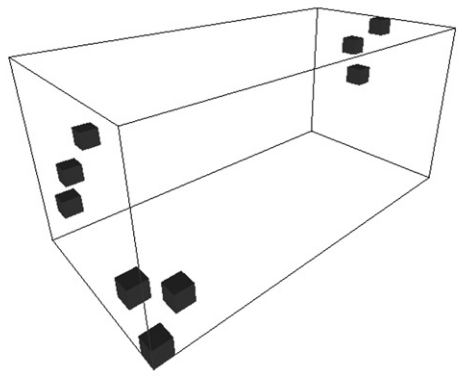

Fig. 1. Spatial cluster grouping.

that splitting the child nodes into three groups is more rational than into two groups. For this purpose, a new 3D spatial-cluster grouping algorithm is introduced, in which the k-means clustering method of data mining is employed to partition k clusters in a set concerning the 3D spatial layout of objects. Because both the spatial coverage and overlap of nodes should be minimized, as well as the shape of MBB nodes being considered, the above mentioned integrative grouping criterion value W is used as the grouping criterion. Fig. 2 illustrates the flow chart of the 3D spatial cluster grouping algorithm, which includes spatial clustering and spatial grouping.

3.2.1. Spatial clustering

Step 1: Calculate the maximal group numbers $k_{\max}$. Ensure that $n/k_{\max} \geq m$,
Where, n is the number of total spatial objects, m is the minimal number of children in a node.

Step 2: Choose different group numbers I ($I=2,\ldots,k_{\max}$) as parameters; adopt the spatial grouping algorithm given below to calculate the corresponding integrative grouping criterion value W using Eq. (1). Select the grouping strategy with the minimum value of W as the final grouping result.

The spatial grouping algorithm, a sub-algorithm of the spatial clustering algorithm, is described as follows:

3.2.2. Spatial grouping

Input: 3D spatial object set $S=\{P1, P2, \ldots, Pn\}$.

Output: k small group sets with inserted objects S_i, $i=1,\ldots,k$.

Step 1: According to the maximum sum distance principle, select k objects from S as the seeds of k group sets S_i, and assign the centroid of the object to be the centroid of the group set. First, find two objects whose distance is the maximum, and then choose one from the rest to make the sum of the distances between it and the first two maximal. Repeat the loop for all k objects. The sum of distances among k objects is calculated as

$$D = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathrm{Dist}(C_i, C_j) \tag{2}$$

where, C_i,C_j are the centroids of object i and object j respectively, $\mathrm{Dist}(C_i, C_j)$ is the three-dimensional distance between C_i and C_j.

$$\begin{aligned} C_i = (x_i, y_i, z_i) = ((\min x_i + \max x_i)/2.0, \\ (\min y_i + \max y_i)/2.0, \\ (\min z_i + \max z_i)/2.0), \end{aligned}$$

$$\begin{aligned} C_j = (x_j, y_j, z_j) = ((\min x_j + \max x_j)/2.0, \\ (\min y_j + \max y_j)/2.0, \\ (\min z_j + \max z_j)/2.0), \end{aligned}$$

$$\begin{aligned} &\mathrm{Dist}(C_i, C_j) \\ &\quad = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \end{aligned}$$

Step 2: Based on the k-means clustering method, the remaining objects are assigned to appropriate sets, and the centroids of these sets are updated. The k-means clustering method means that the object is added into the set the centroid of which is nearest to the centroid of the object.

Step 3: Repeat step 2, until all the objects are completely grouped into k sets.

3.3. 3D R-tree insertion

Insertion is the key step of R-tree generation, in which new objects are inserted into the R-tree and the rational structure of R-tree is formed. Insertion includes two important sub-operations: node-choice and node-split. To insert one object into the R-tree, one starts from the root to judge the sub-tree until the leaf node is reached. If the new insertion of an object into the leaf node leads to the overflow of a node, the overflowing node therefore has to be split and divided into several small nodes. If the split of the under-layer node leads to overflow of its father node, the split operation of the father node continues to the root node. During the insertion, 3D spatial cluster grouping guarantees that the adjacent objects in space are stored in the same or sibling nodes.

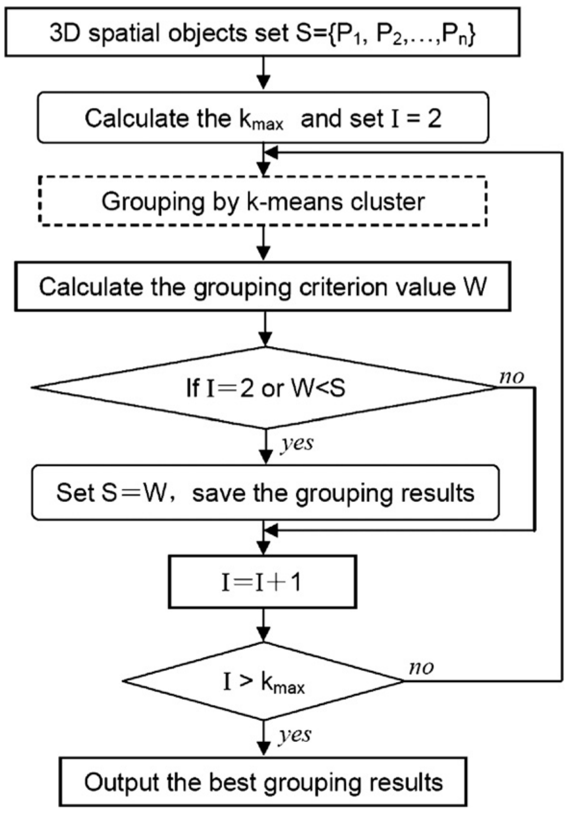
3D spatial objects set S={P₁, P₂,…,Pₙ}
Calculate the kmax and set I = 2
Grouping by k-means cluster
Calculate the grouping criterion value W
If I=2 or W<S
no
yes
Set S=W, save the grouping results
I=I+1
I > kmax
no
yes
Output the best grouping results

Fig. 2. Flowchart of the 3D spatial cluster grouping algorithm.

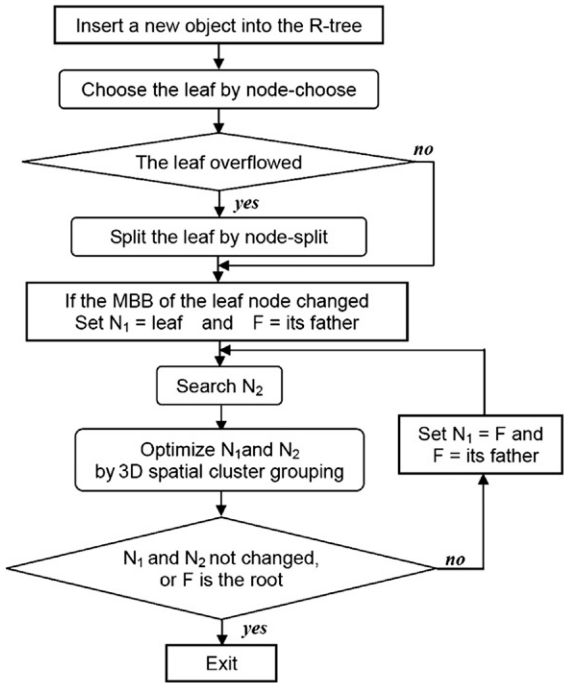
Insert a new object into the R-tree
Choose the leaf by node-choose
The leaf overflowed
no
yes
Split the leaf by node-split
If the MBB of the leaf node changed
Set N₁ = leaf and F = its father
Search N₂
Optimize N₁and N₂ by 3D spatial cluster grouping
Set N₁ = F and F = its father
N₁ and N₂ not changed, or F is the root
no
yes
Exit

Fig. 3. 3D R-tree insertion algorithm.

According to the minimum distance principle of common node-choice, the leaf node with its centroid closest to the object's centroid is selected for the insertion of the object. The insertion operation probably leads to the change of MBB of nodes from the root to the leaf, which is the reason why current R-tree forms an irrational layout. In order to search for the irrational nodes for further optimization of the tree, a new dynamic insertion algorithm is proposed as shown in Fig. 3:

Step 1: Node-choice operation, to find the leaf node into which the object is inserted.
Step 2: Insert the object; if the node overflows, carry out the node-split operation in order to realize the optimum grouping.
Step 3: If the insertion leads to a change of MBB of the leaf node, set the changed leaf node to be N_1 and its father node to be F.
Step 4: Search for the sibling node N_2 that has the largest overlap with N_1 using the 3D spatial cluster grouping algorithm, to optimize and group the child nodes of N_1 and N_2 again.
Step 5: If the original group of N_1 and N_2 are optimized, end the operation; if there is any change, set F to be N_1 and the father of F to be F, repeat Step 4, up to the root node.

As shown in Fig. 4, suppose ⑤,⑥,⑦,⑧,⑨ and ⑩ represent different objects, and the object ⑩ is the new one to be inserted into node ④. In Fig. 4, the children of ③ are not described. Because the operation leads to the change of MBB of node ④, all the child nodes of node ④, and node ②, which has the maximum overlap with node ④, have to be optimized according to the grouping algorithm. After the optimization, the overlap between nodes ② and ④ is minimized. If this operation leads to the MBB of father node ① changing, the optimization process from the child node to its father node is repeated, i.e. until the node MBB does not change again or up to the root node.

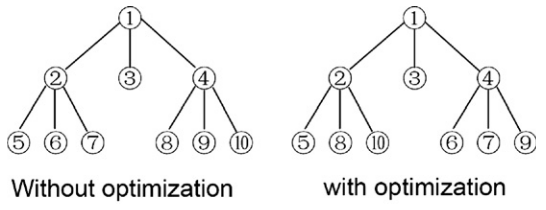

Fig. 4. Dynamic insertion process of R-tree.

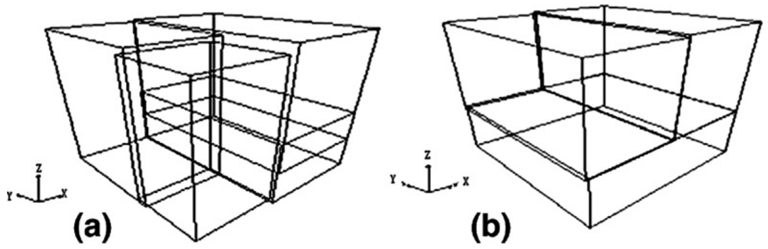

Fig. 5. Comparison of 3D R-tree index methods. (a) Classical algorithm. (b) Improved algorithm.

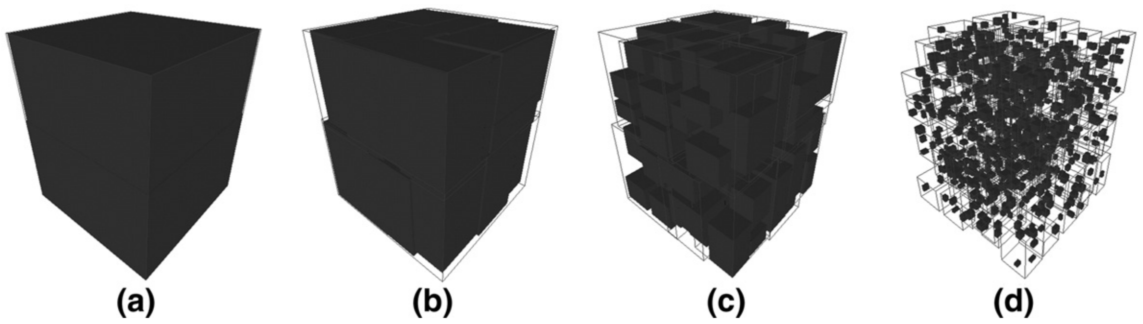

Fig. 6. 3D R-tree generation procedure. (a) Root layer. (b) 2nd middle layer. (c) 3rd middle layer. (d) Leaf layer.

The advantage of this insertion algorithm is that it is possible to optimize all the influenced nodes during the insertion procedure because the whole tree may be traversed if necessary. Irrational nodes are optimized through dynamic adjustment.

As illustrated in Fig. 5, the wire frame denotes the MBB of nodes at the same layer. There is little overlap in the result of the new algorithm ((b) compared to (a)), and the lengths of 3D MBB along the three axes are almost equal. This facilitates the reduction of the multi-

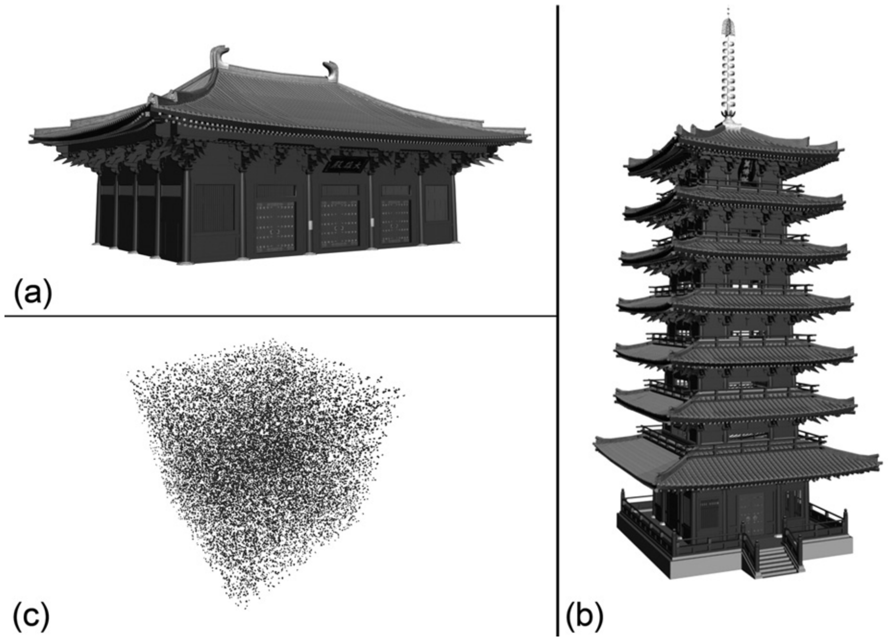

Fig. 7. 3D representation of the test data. a) Building model. b) Tower model. c) Simulated model.

Table 1
Description of the test data

Model type	Reference image	Number of components	Number of triangles	Size of scene ($x*y*z$, m^3)
Building	Fig. 7-a	9282	389,668	33.6 * 24.6 * 15.8
Tower	Fig. 7-b	3518	336,450	13.7 * 13.6 * 32.7
Simulated model	Fig. 7-c	20,000	240,000	1000 * 1000 * 1000

Table 2
Query time for building model (s)

Algorithm	Query type			
	Regional query			Point query
	0.01%	0.1%	1%	
Classical R-tree	0.1228	0.2197	0.6884	0.0836
R*-tree	0.1229	0.2252	0.6816	0.0859
Improved algorithm	0.0447	0.1029	0.4262	0.0212

path query and improves the spatial query performance. Fig. 6 illustrates an experimental result of 3D R-tree generation.

4. Experimental analysis

In order to verify the performance of our proposed algorithm, a test environment was established using a desktop computer with the configuration of an Intel Pentium 2.0GHz CPU, 512M RAM. In order to keep the performance comparison manageable, the default R-tree parameters were set as: $M=10$, $m=4$. In this condition, the page size for data and directory pages is 1024 bytes. For comparison purpose, the classical R-tree and R*-tree are employed for experimental analysis. To compare the performance of the three algorithms, three different types of true 3D models were selected as shown in Fig. 7. The first model is a complicated building, the second model is a tower, and the third model is simulated data; the detailed descriptions of these three models are listed in Table 1. The first and second models are very complicated and include many components of various sizes and shapes. The simulated model consists of randomly generated box models (each rectangle surface of the box model consists of two triangles implicitly) with different sizes from 1 to 10.

Two typical queries, i.e. the regional query and point query, are tested. The query algorithm of the improved R-tree is the same as that of the classical R-tree. The area of regional query varies from 0.01%, 0.1% to 1% relative to the volume of the whole model space. The experimental results of regional and point queries are listed in Tables 2 3 and 4 for the three models, respectively.

As well as query performance, the performance of the insertion operation is also very critical to the construction of R-tree. The test results of insertion for the three models are presented in Table 5, which records the time cost of the R-tree generation.

From the analysis of experimental results, our proposed algorithm is superior to both classical R-tree and R*-tree in query performance, especially with respect to the point query and small region query. Generally, query performance is improved by more than a factor 2, and the point query on uniformly distributed spatial data is improved by 50 times. Furthermore, our proposed algorithm also improves the insertion operation performance of tree construction. This is particularly due to the structure improving properties: the tree shape is kept rational and the object is inserted into the proper node all the times, the node split operations therefore being greatly reduced.

Table 3
Query time for tower model (s)

Algorithm	Query type			
	Regional query			Point query
	0.01%	0.1%	1%	
Classical R-tree	0.0312	0.0312	0.1826	0.0259
R*-tree	0.0342	0.0353	0.1847	0.0274
Improved algorithm	0.0102	0.0113	0.1205	0.0088

Table 4
Query time for simulated model (s)

Algorithm	Query type			
	Regional query			Point query
	0.01%	0.1%	1%	
Classical R-tree	1.2374	1.7471	2.8322	0.7531
R*-tree	0.8984	1.2846	2.2860	0.5387
Improved Algorithm	0.0349	0.0946	0.4477	0.0125

Table 5
Time cost of R-tree generation (s)

Algorithm	Data name		
	Building model	Tower model	Simulated model
Classical R-tree	0.8730	0.3017	1.9618
R*-tree	1.4749	0.5009	3.1303
Improved algorithm	0.8272	0.2739	1.6322

5. Conclusion

Because of the comprehensive consideration of MBBs' coverage, overlap and shape of nodes, the 3D spatial clustering algorithm carries out the grouping of nodes in the most reasonable way with minimum overlap and balanced volume of coverage, which therefore distinctively improves the classical 3D R-tree structures. This paper provides a potentially true 3D spatial index for increasingly massive 3D spatial database's fast data retrieval in real-time applications of virtual geographic environments. Further research will include the integration of the LOD models and the 3D R-tree, as well as its practical applications in spatial queries within a pyramid or cone of vision.

Acknowledgments

The work described in this paper was supported by National Natural Science Foundation of China (40671158 and 40620130438) and National High Technology Research and Development Program of China (2006AA12Z224).

References

Arens, C., Stoter, J., van Oosterom, P., 2005. Modeling 3D spatial objects in a geo-DBMS using a 3D primitive. Journal of Computers and Geosciences 31 (2), 165–177.

Beckmann, N., 1990. The R*-tree: an efficient and robust access method for points and rectangles. Proceedings of ACM SIGMOD, Atlantic, pp. 322–331.

Brakatsoulas, S., Pfoser, D., Theodoridis, Y., 2002. Revisiting R-tree construction principles. Proceedings of 6th Advances in Databases and Information Systems, Bratislava, Slovakia, pp. 149–162.

Gaede, V., Günther, O., 1998. Multidimensional access methods. ACM Computing Surveys 30 (2), 170–231.

Guttman, A., 1984. R-trees: a dynamic index structure for spatial searching. Proceedings of ACM SIGMOD international conference on Management of data, Boston, Massachusetts, pp. 47–57.

Kamel, I., Faloutsos, C., 1994. Hilbert R-tree: an improved R-tree using fractals. Proceedings of 20th International Conference on Very Large Databases, Santiago, pp. 500–509.

Kofler, M., 1998. R-trees for Visualizing and Organizing Large 3D GIS Databases. PhD. Dissertation, Graz University of Technology, Austria.

Lin, H., Zhu, Q., 2005. Virtual geographic environments, in: large-scale 3D data integration: challenges and opportunities. In: Zlatanova, Sisi, Prosperi, David (Eds.), CRC Press, pp. 211–231.

Sellis, T., 1987. The R+-tree: a dynamic index for multi-dimensional objects. Proceedings of 13th International Conference on Very Large Databases, Brighton, pp. 507–518.

Zlatanova, S., 2000. 3D GIS for Urban Development. PhD. Dissertation, International Institute for Geo-Information Science and Earth Observation, Netherlands.