DE GRUYTER OPEN

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 4

Sofia • 2015

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0059

BRLO-Tree: A Data Structure Used for 3D GIS Dynamic Scene Rendering

Wenju Wang[1], *Zhang Xuan*[2], *Liujie Sun*[1], *Zhongmin Jiang*[1], *Jingjing Shang*[1]

[1]*University of Shanghai for Science and Technology, Shanghai 200093, China*
[2]*Shanghai Conservatory of Music, Shanghai 200031, China*
Email: wangwenju666@163.com

Abstract: *BRLO-Tree* (*Block-R-Tree-Loose-Octree*) *is presented in this paper based on the R-Tree and Loose-Octree. The aim of the structure is to visualize the large scale and complex dynamic scenes in a 3D* (*three-dimensional*) *GIS* (*Geographic Information System*). *A new method of clustering rectangles to construct R-tree based on an improved K-means algorithm is put forward. Landform in 3D GIS is organized by R-Tree. The block is used as the basic rendering unit. The 3D objects of each block are respectively organized by a Loose-Octree. A series of techniques, based on this data structure, such as LOD* (*Level of Detail*), *relief impostors are integrated. The results of the tests show that BRLO-Tree cannot only support the large scale 3D GIS scene exhibition with wandering and fighting, but it can also efficiently manage the models in a dynamic scene. At the same time, a set of integrated techniques based on BRLO-Tree can make the rendering pictures more fluence and the rendering time vastly improved.*

Keywords: *BRLO-Tree, data structure, geographic information system, dynamic scene.*

1. Introduction

3-D Geographic Information System (3D GIS) has already been extensively applied to various fields, such as real estate digital sand table, large-scale scenic spot display, urban planning, geological exploration and hydrographic survey for static display of scenes, which can bring the obvious economic and social benefits [1]. Now people have paid more attention to interactive simulation based on 3D GIS in the fields of environment and water resource monitoring and management, mineral resource exploitation, electro-communication, public rescue activities [2], and military activity simulation, etc. However, during this process, the scene model is always in changes or movement. So looking for an appropriate data structure to

support the real-time rendering of the dynamic scene for 3D GIS has become an important research problem now [3, 4].

Today most of the real-time rendering algorithms are based on various model index data structures constructed before rendering. These data structures can be roughly divided into hierarchical structure and bounding volume hierarchies. Hierarchical structure separates the data space along the predefined hyper-plane without considering data distribution. The divided data are not connected to each other, but they can completely reproduce the overall data space after combination. BVH [5], KD [6], Quad-Tree and Octree are all belong to this type. This type of data structure cannot organize and manage many data models, so they almost are not suitable for 3D GIS. The bounding volume hierarchies divide the spatial data by way of minimum bounding volume according to the situation of data distribution. However, such division will cause overlapping of the divided zones [7]. R-Tree, being one of the most extensively applied spatial index structures at present belongs to this type. R-Tree is an entirely dynamic spatial index data structure and it can carry out insertion, deletion and node query at the same time. However, when R-Tree is extended to 3D space, node overlapping will bring to the increase of multichannel query to lead to a decreasing of the search efficiency [8, 9]. Therefore, Chen Peng and Meng Lingkui [10] proposed an R-Tree index structure based on spatial topological constraints which can increase using ratio of nodes, but it will be a failure for objects with complex topological relations. Zhu Qing and Gong Jun [11] carried out 3D space cluster grouping via a K-means algorithm and improved node selection and node split, so as to solve partly serious overlapping problems of nodes in 3D R-Tree. Due to the simple structure and easy realization, Octree structure is also applied to index 3D spatial data [12]. However, when the index is established, a distribution range of the spatial object has to be predicted, so it could meet dynamic requirements of large-scale spatial data. In 2003, Xia and Prabhakar [13] proposed a mixed index structure of combining Quad-Tree with R-Tree to manage scenes with numerous moving objects, where Quad-Tree can index objects in quick movement and R-Tree can index quasi-static objects. This method pays attention to processing of moving objects on terrain surface and it does not make efficient management for an underground object model. In 2010, Block-Quadtree-R-Tree was proposed by Pina, Seron and Cerezo [7], where the block is used as an elementary unit of a city and the spatial data is divided via a Quad-Tree. This method can display large-scale 3D urban scenes in walking and flying ways and rendering time has been greatly reduced. However, it is only applicable to render the 2.5D urban scenes and manage the moving elements.

In fact, there are not still any appropriate data structures that can realize real-time rendering for real a 3D GIS dynamic scene, which can support model insertion, deletion and switch, etc. interaction demand. To solve this problem, a hybrid index data structure of Block-R-Tree-Loose-Octree (BRLO) is proposed in this paper. Meanwhile, based on this scene organization structure, LOD, Relief Impostors technologies are adapted to efficiently visualize in real-time the 3D GIS dynamic scenes.

2. Relevant knowledge

2.1. R-Tree

Region-Tree was proposed by Guttman in 1984 and constructed according to the spatial relationship among Minimum Bounding Rectangles (MBR) of the spatial entity. All leaf nodes of this tree are at the same level, so it is a highly balanced tree. Actually, it is the natural extension of B-Tree at K dimensions. To 2D spatial data objects, the structure of R-Tree is constructed (as shown in Fig. 1) where the maximum capacity of the node is $k = 3$. This tree can directly and flexibly store and read spatial entity objects, so it is the most commonly used spatial data index structure in GIS at present.

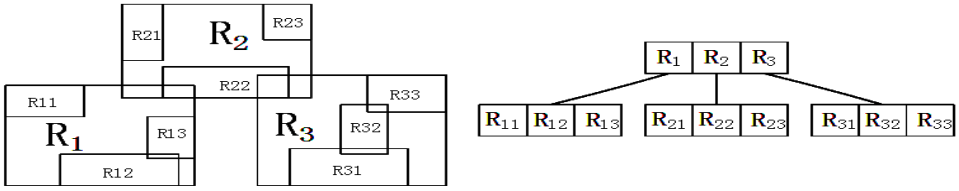


Fig. 1. Example of R-Tree

Typical characteristics possessed by R-Tree [9]:

i) A root node includes two child nodes at least, unless it is a leaf node.

ii) Suppose m is the minimum number of spatial entity contained in the node and $2 \le m \le (M/2)$, then the number n of spatial entity contained in each node mustbe between *m* and *M*.

iii) All leaf nodes are at the same level of the tree.

In addition, both data rectangles that can be stored in R-Tree and the index entry rectangle that can be retrieved can happen to overlap. With the rise of the data size, the overlapping amount of the space region will be rapidly increased, which will inevitably make the retrieval efficiency drop sharply. In large-scale scenes, the object model also has a large quantity, so the corresponding method must efficiently avoid overlapping of the rectangle data and index directory and thus to improve efficiency of the retrieval.

2.2. Loose-Octree

Octree is a frequently used spatial data organization structure. It divides 3D space into eight cubes with the same size. The principle of dividing one into eight will still be used for each cube, till division is not needed or the specified level is reached. However, as a method of space division, Octree has some defects as given below:

i) A small object might be stored in a very huge Octree node of bounding volume, so thus it may occupy a high level in the tree. As a result, the division efficiency will be decreased.

ii) When the object is in movement, a huge expense would be cost of real-time adjustment for the structure of Octree. Therefore it is difficult to manage the dynamic objects efficiently.

iii) Strict space division will destroy a topological structure of the object, and the object model cannot be managed in an object-oriented way.

Therefore, Thatcher Ulrich proposed the Loose-Octree based on the Octree [14]. The division for 3D space and encoding for 3D object are shown in Fig. 2.

In Loose-Octree, suppose that "w" is the side length of the bounding volume of a root node; then the depth is the "dep" level of the tree, the side length of bounding volume centering on root node is

$$L(\text{dep}) = l \times w \times (2 \wedge \text{dep}), \tag{1}$$

where l is the enlargement factor of the side length. When the situation of a level for most nodes and node centers are unchanged, a small object that runs across a boundary and sticks to a node at a high level can be laid at the node position with a deeper level. Thus, the division efficiency can be increased and more precise spatial database query will be generated. As shown in Fig. 3, compared with the root node where the maximum quadrangle is represented by an imaginary line, the hexagon is quite small. But it cannot be laid in a child node of a root node, because it crosses a division line. However in a Loose-Octree, the side length of the bounding volume is enlarged l times, so the hexagon can be contained in a child node B of the root node (the bounding quadrangle is presented by black lines, and slight excursion is carried out in order to distinguish these quadrangles with black lines). The parameter l in Equation (1) cannot be selected randomly. According to the practical experience, when the parameter $l < 2$, the Loose-Octree will encounter the puzzle of a stick plane. When $l > 2$, an excessively Loose-Octree structure will be formed. Thatcher Ulrich (see [14]) set $l = 2$ as a beneficial composite value. At this time, the edges of the two loose bounding volumes pass through each other's node center. When the object produces a movement, the time complexity of the update operation is only $O(1)$.

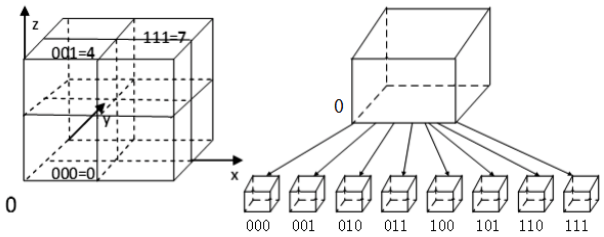


Fig. 2. Encoding for child nodes of Loose-Octree

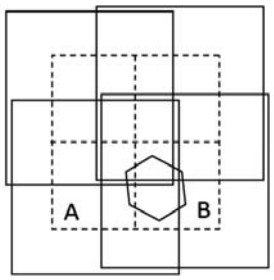


Fig. 3. Stick plane phenomenon

3. Scene division method and BRLO-Tree

3.1. A scene division method

As for a large-scale of 3D GIS urban scene, all models cannot be loaded at once for real-time rendering. So the scene needs to be divided into suitable sizes. The steps of the scene division method are given below:

Step 1. Division of the rendering area: The rendering area can be divided via Minimum Bounding Rectangle (MBR) on the obtained satellite image.

Step 2. Determination of each large-area landform ranges: DEM (Digital Elevation Model) contour of various large-area landforms can be obtained via a contour extraction method in the rendering area.

Step 3. Division of a block in a large-area landform DEM: Many east-west and south-north main roads are extracted from the satellite image of every large landform on the basis of road extraction method. The block formed by the areas divided among various main roads or landform contours is known as rhe minimum block, denoted by the block. In this process a large-area landform DEM is divided by the right edge of every main road as the boundary. Thus landform block decomposition is conducted, and any underground utilities can easily be organized to one block.

Step 4. Determination of the block each 3D entity model is located in: 3D models of all entities are constructed on each large-area landform DEM. Each 3D entity model and every block DEM are surrounded via an Oriented Bounding Box (OBB). The bounding box is projected into the oxy plane with respect to the world coordinate system. If the projection rectangle center of entity model is located in the projection rectangular region of a certain landform block DEM divided according to Step3, then it can be determined that this entity model is within this block.

Step 5. Determination whether the entity model is above or under the ground of a block: Orthographic projection is conducted for the oriented bounding box of various over-ground and underground 3D entity models and block DEM on OYZ plane. If the minimum value $Z_{\min}$ of the orthographic projection rectangle side length of the oriented bounding box of 3D entity model at z direction is greater than the maximum value $Z_{\max}$ of the orthographic projection rectangle side length of the oriented bounding box of the block DEM, then the 3D entity model is above the ground of block DEM; otherwise, it is under the ground of block DEM.

Step 6. All over-the ground and underground models in any block are divided via a Loose-Octree.

3.2. BRLO-Tree

3.2.1. Improved K-means clustering method

Due to overlapping of the minimum bounding rectangles, query efficiency of R-Tree is low. Therefore, by setting the central point of MBR for the landform or block as a sample point, R-Tree is constructed via K-means clustering method [15] improved by us in this paper. In this improved method, the adjacent matrix A as an adoption and impact factor, r is adjusted to the computation result of the Euclidean distance. Compared with the “minimum area increment” criterion to construct the R-Tree, this method can obviously reduce the blank region and the overlapping region of the bounding rectangle, thus high query efficiency of R-Tree will be guaranteed. The improvement measure of K-means clustering for R-Tree is as follows:

Step 1. Selection of the initial clustering center has a great influence on the clustering result. At Step 1 of K-means algorithm, MBR of the total rendering region is calculated and is quartered into four small rectangles. A node with four pointers is set up as a parent node. The MBR center point of every spatial object on

this rendering region is marked as a sample s_i, i=1, 2, …, n. The distance from the sample $s_i \in \{s_1, s_2, \ldots, s_n\}$ to central points of the four small rectangles respectively is calculated, according to Euclidean distance formula. Four sample points (the MBR center of spatial object) that have the shortest distance to the four central points are selected as initial clustering centers $c_j(m),\ 1 \le j \le 4,$ where m is the order of the iterative operation.

Step 2. Adjacent matrix A is built. If s_i is in some overlap or adjacent to s_j, then $A_{ij} = 1$, otherwise $A_{ij} = 0$. Formula for Euclidean distance is determined by r:

$$d(s_i, c_j(m)) = \sqrt{r(s_i - c_j(m))^2}. \tag{2}$$

According to the adjacent matrix, if s_i is some overlapped (adjacent) to $c_j(m)$, then $r = 0.5$, otherwise $r = 1$.

Landforms (minimum blocks) being some overlap (adjacent) to $c_j(m)$ are guaranteed to be together at first through formula (2).

Step 3. As for s_i, $c_j(m)$ that minimizes $d(s_i, c_j(m))$ is determined, then s_i is put into the cluster center of $c_j(m)$. Finally, the clusters $\mathrm{Cl}_j(m),\ j = 1, 2, 3, 4,$ are formed; N_j is the sample number of $\mathrm{Cl}_j(m)$.

Step 4. The number of samples N_j is checked. If $N_j > \left\lceil \frac{n}{4} \right\rceil$, $N_j - \left\lceil \frac{n}{4} \right\rceil$ the samples which are far from the center of the cluster are deleted and reallocated to the cluster $\mathrm{Cl}_i(1),\ i = 1, 2, 3, \cdots, n,\ i \ne j,$ according to (2). Step 4 is repeated, till n samples are uniformly allocated to 4 clusters. For every cluster if $\left\lfloor \frac{n}{4} \right\rfloor \le N_j \le \left\lceil \frac{n}{4} \right\rceil$, then a child node with four pointer is allocated.

Step 5. The parent node points to four child nodes.

Step 6. For every cluster $\mathrm{Cl}_j(m),\ j = 1, 2, 3, 4,$ if $0 \le N_j \le 4$ (N_j is the number of samples in the cluster $\mathrm{Cl}_j(m),\ j = 1, 2, 3, 4),$ then four pointers of a child node save MBR of N_j sample points. R-Tree level which the current parent node locate in, is returned to deal with others parent nodes according to Step 1 to Step 6. The algorithm will not be ended until the child nodes of the root node are not all treated. Every region of $\mathrm{Cl}_j(m),\ j = 1, 2, 3, 4,$ is regarded as a new rendering region, then $m{+}{+}$ and Step 1 up to Step 6 are repeated for every $\mathrm{Cl}_j(m),\ j = 1, 2, 3, 4.$

3.2.2. Building of a BRLO-Tree

Step 1. DEM contours of various landforms are surrounded by MBR in MBR of a rendering area, and clustered by improved k-means algorithm of Section 3.2.1.

Step 2. Various blocks are surrounded by MBR in MBR of every large landform, and clustered by improved K-means algorithm of Section 3.2.1.

Step 3. For model data under the same block node, the arrangement sequence is DEM, model_g, model_{ug}, ME. Such sequence is determined for both model_g and

$model_{ug}$ above or under DEM. In addition, their exhibition cannot run without rendering for DEM. Therefore, loading and rendering of DEM takes the front position, and $model_g$ and $model_{ug}$ are in the middle. ME is a moving element and plays an auxiliary and speckled role in the scene, so its loading and rendering takes the last position.

Step 4. As for the over-ground model $model_g$ or underground model $model_{ug}$, they are managed by the Loose-Octree constructed. The construction process of the Loose-Octree is: firstly, on a block $model_g$ a set of object models over-ground and $model_{ug}$ is a set of object models under-ground. They are encircled in two cubic bounding boxes. Then one of the bounding boxes is cut into eight equal sections in a recursive manner. In order to make the child nodes possess the same dimension, the Loose-Octree is different from Octree in the way of dividing the one into eight. If there is no object in a node for the Loose-Octree, this node still will continue to be divided into eight equal sections, till get to the specified level depth.

BRLO-Tree data structure can be built through the above 4 steps, as shown in Fig. 4. It can realize management and efficient retrieval for landform DEM and 3D entity model in 3D GIS.

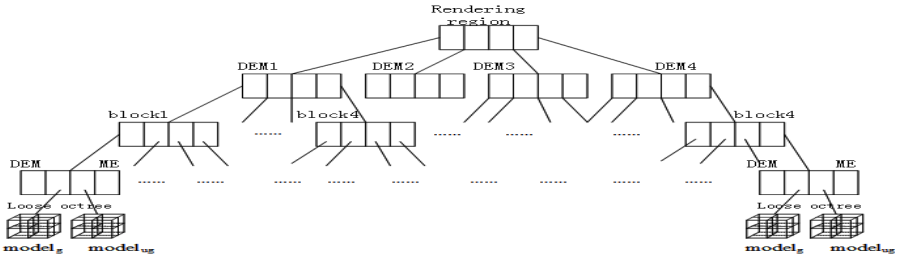


Fig. 4. BRLO-Tree hybrid-tree structure

4. Rapid rendering method based on BRLO-Tree

4.1. LOD based on BRLO-Tree

Each node above ME level of BRLO-Tree can add a LOD child node which stores four LOD models with different levels for this node model. According to the present viewpoint height and judgment for visible regions, the corresponding LOD model is called to realize the spontaneous transition among the models with different levels of details, which can display the multi-scale 3D map. Meanwhile, each node of the Loose-Octree of $model_g$ or $model_{ug}$ (except leaf node) can also add LOD child nodes to store different LOD models, according to the complexity of the model. However, at this time, LOD technology is simply applied to acceleration of scene rendering.

4.2. Model scheduling strategy based on BRLO-Tree

Step. 1. When the viewpoint position is at the short-distance height on the ground of the current block, LOD models of $model_{ug}$ and ME should be loaded, after DEM detailed model and over-ground $model_g$ the detailed model which belongs to the current block is loaded.

Step. 2. LOD models of $model_g$ and DEM of other blocks close to the current block should be loaded.

Step. 3. When the viewpoint height is lower than the setting ground height of the current block, a detailed model of $model_{ug}$ needs to be loaded.

Step. 4. When the distance from the viewpoint to the centroid of block is smaller than the setting distance, DEM, $model_g$ and the detailed models of $model_{ug}$ locating in this block must be loaded.

Step. 5. If the viewpoint changes in a large range and jumps out of the current block and near blocks, Impostors technology can be used to rapidly load Impostor photos in order to avoid the blank screen caused by the time interval required for model loading due to sudden movement of the viewpoint. Meanwhile, multithreading is invoked to parallel load corresponding impostor photos for the viewpoint to make a step forward, take a step back, turn left for 90 degrees, and turn right for 90 degrees to display card buffer. This group of photos is stored in nodes of the block of BRLO-Tree.They can be invoked and displayed through operations of $\uparrow,\downarrow,\leftarrow,\rightarrow$ on the keyboard. Then by utilizing the time difference of users adopting further operations in the above, multithreading technology will be used to repeat Step 1.

This model scheduling strategy has the characteristics of a short time of scene graph search and model loading, flexible scheduling strategy. It solves the frequent "fake system halted" problem of 3D game engine caused by long waiting time due to large-scale 3D model loading. Besides, it can also solve the problem that numerous CPU and GPU resources are occupied for out-of-core viewpoint visibility query.

4.3. Management of BRLO-Tree for moving elements

Every moving element, such as automobile running on the driveway and virtual human walking along the sidewalk, is always associated with a street and a movement direction. According to our BRLO-Tree construction method, every street belongs to a block. Therefore, every moving element in unidirectional movement on street just belongs to a block. For the sake of convenient management, the geometry of moving element is stored as the child node of the end of the block node. In this way, it will be easy for us to know the position and speed of each moving element at any time. When the moving element goes to a different block, the moving element must break away from the old block and connect to the node of the new block. Namely, the pointer of moving element from a node of the original block is transferred to a node of the new block. However, not all moving elements need a new ownership connection. The connection processing is only required for those moving elements that will not encounter obstacles during unidirectional movement in the next block or those moving elements that have already been removed. As for other moving elements, they will automatically recover to the initial position and restart the movement after reaching the boundary of block, or they will be controlled by setting the movement time or pause time when the elements meet obstacles. In order to relieve the operation burden of CPU and GPU, any collision detection is not carried out.

4.4. 3D object insertion and deletion for BRLO-Tree

If there is a new landform to render in rendering region, Step 1 of Section 3.2.2 is executed to determine the storage position of the new landform in BRLO-Tree. If this landform is complex and there are roads, it needs to divide into many blocks. A series of operations for Step 2, 3 and 4 of Section 3.2.2 are done to build up the branch node for this new landform DEM.

In landform DEM, there is a closed relationship between the block numbers and the road construction or abandonment. The construction or abandonment of a road leads to the change of the block numbers n_{block}, $n_{block} \geq 2$, this is because a road may cross over other roads. Steps 2, 3 and 4 of Section 3.2.2 are done to set up branch node for new block.

In dynamic 3D GIS, the insertion of 3D objects always appears in $model_g$ or $model_{ug}$ which is built up by Loose-Octree. Now the specific operation process will be illustrated by taking insertion of a node:

Step. 1. The node of landform DEM where 3D object locates is determined.

Step 2. The node of the block where 3D object is located is found out.

Step 3. The node of $model_g$ or $model_{ug}$ where 3D object located is found out, namely, the root node of a Loose-Octree is determined.

Step 4. Determine the depth of the node according to the object size, $\text{depth}(R) \geq \log_2(lw/(R) - 1$, namely that:

$$\text{depth}(R) == \text{floor}(\log_2(lw/(R)) - 1. \tag{3}$$

In (3), R represents the maximum radius of the object and the function floor() denotes the integral part after rounding down.

Step 5. According to the calculated depth and central position of the object (under local coordinate system), the index number of node in Loose-Octree is determined by

$$\text{index}\{x, y, z\} = \text{floor}[(\text{object.center}\{x, y, z\} + w/2)/(w/2^{\text{depth}})] - 1. \tag{4}$$

The numbering method is shown in Fig. 2.

Step 6. The storage location of the model is added to the position of the index number, which is easy to query and call the 3D model.

If a landform node or a 3D object node is to be deleted in a BRLO-Tree, the pointer of its parent node will be set as NULL.

5. Experimental results and discussion

This method proposed in this paper is used for real-time rendering of 3D Yangshan Port which has an area of 8.14 km^2. The computer is equipped with Intel Core2 Quad Q9550 2.8GHZ CPU, 3 GB main memory, and Nvidia GeForce GTX480 display card. The dynamic scene contains 25,437 object 3D models and 46,157,026 triangular faces in total.

Fig. 5 shows the result pictures at every stage obtained in the construction process for 3D GIS scene structure of Yangshan Port, according to the scene division and BRLO-Tree construction methods are proposed in this paper. Fig. 5a is the obtained satellite remote sensing image of Yangshan Port. Fig. 5b is the result map gained on the basis of the road and terrain contour extraction algorithm [16]. In

Fig. 5c there are (1)-(6) MBRS of landform DEM marked by red imaginary lines. Based on Fig. 5b, three main roads at northwest/southeast direction and twenty one main roads at northeast/southwest direction are found out. In order to divide the scene more conveniently and swiftly, we have also properly extended some main roads (along their original directions) (they are marked by red lines in Fig. 5c. Road and terrain contour make the various landform DEM (1) divide into 31 blocks. These blocks all are bounded by MBR. The 3D GIS BRLO tree scene structure of Yangshan Port can be finally formulated according to Section 3.2.2, as shown in Fig. 5d.

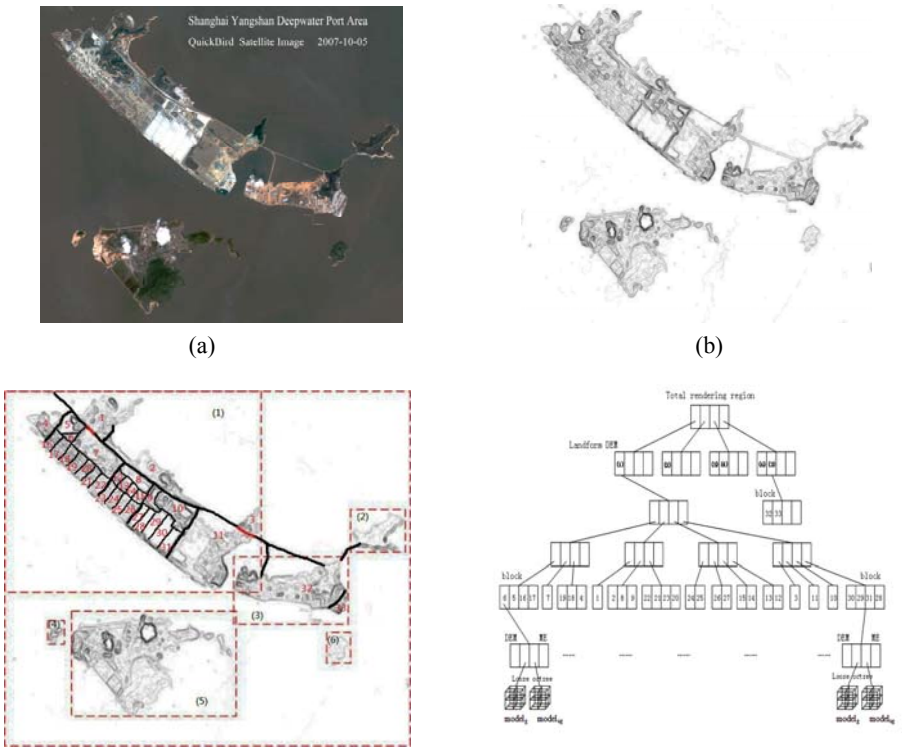


Fig. 5. Building 3D GIS BRLO-Tree scene structure of Yangshan Port: Satellite image of Yangshan Port (a); road extraction result map (b); blocks of main roads division (c); BRLO-Tree scene structure (d)

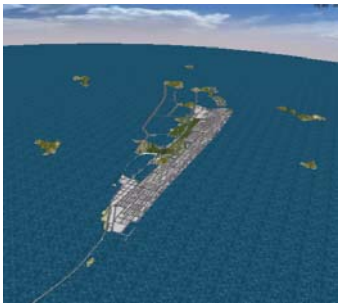


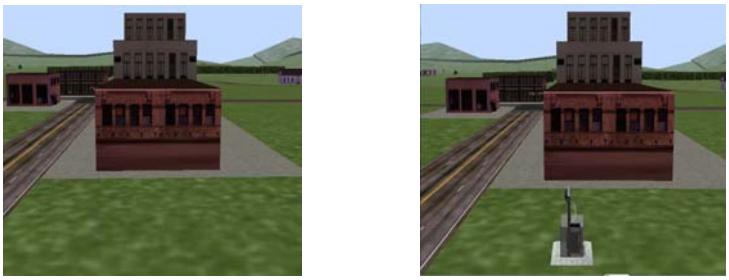
Fig. 6. 3D display under the scale of 1:2500

Fig. 7. Management of moving element car

Fig. 6 is the 3D display for geographic information of Yangshan Port under a scales of 1:2500. It shows that this data index structure based on BRLO-Tree can realize 3D geographic information display under different scales.

In Fig. 7, the scene in which three cars stop at the stalls is shown. However, in the program, the right car in the figure stops at this position will a dynamic move with a constant speed for 20 s (the stopping position is determined by the driving time set in the program and no collision detection is conducted). This car does not leave its block, so no extra management is required in BRLO.

During the process of dynamic interaction in 3D GIS, 3D model often needs to be added to the scene. For example, an electrical equipment-transformer is erected for line reconstruction in block (3) of landform (1). Then 3D mode transformer needs to be inserted to the scene. The scene before insertion of a 3D mode transformer can be seen from Fig. 8 (a). The concrete insertion process is described in Section 4.4. In a Loose-Octree, the model$_g$ the parameter is set to $k = 2, d = 4$ in block (3) of landform (1). Fig. 8(b) shows the scene after the transformer model insertion. If a 3D object model is deleted in 3D GIS, the pointer of its parent node in BRLO-Tree will be set as NULL. According to display (see Figs 7 and 8), it can efficiently manage 3D model with dynamic change during interactive operation in the corresponding outdoor scenes.



(a) (b)

Fig. 8. Insertion for 3D model transformer: scene before model insertion (a); scene after model insertion (b)

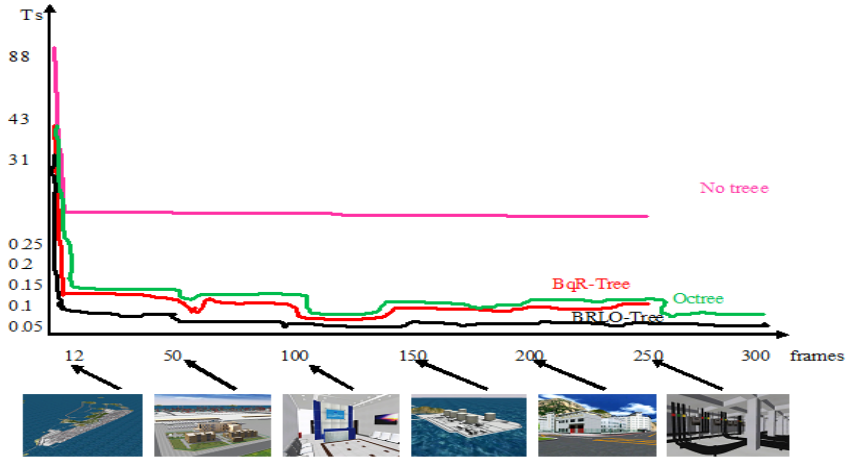


Fig. 9. Comparison results of four tree structures with change of the rendering time

In the following part, excellent performances of rendering scenes by model scheduling via BRLO-Tree will be illustrated and compared with the three typical scene organization structures of No-Tree, Octree and BqR-Tree from the aspect of time response. The camera goes through the viewpoint transformation process of overlooking Yangshan Port, envisaging the work service area, entering the entrance of Yangshan substation, and glancing over natural gas wharf and underground cables. In this process, 250 or 300 frames images are rendering. In Fig. 9, the *X* axis records the frame order and the *Y* axis records the corresponding time for rendering each frame image. The comparison rendering figure of the four scene organization structures about No-Tree, Octree, BqR-Tree, BRLO-Tree is shown in Fig.9. (1) No-Tree structure means to load the model one time. So the rendering time for the initial frame in Fig. 9 is long, reaching about 88 s. Later the rendering time for each frame is almost the same. Octree and BqR-Tree actually quarter the plane area, so the rendering times for the initial frame are almost the same. (2) For Octree structure, the level depth of the scene is higher than the other tree structures. So it the rendering time is longer. (3) Similarly to No-Tree structure, BqR-Tree structure does not support 3D GIS dynamic scene rendering, while Octree and BRLO can support such rendering. Therefore, Octree and BRLO has a corresponding rendering frame-time curve when it is rendering underground cable from the 250th frame to the 300th frame, while BqR-Tree and No-Tree structures cannot provide such a curve in Fig. 9. (4) For BRLO structure, the level depth is lower than that of Octree, so the time of querying 3D model is shorter than Octree and BqR-Tree. In actual rendering, its rendering time for the initial frame is 31 s, less than 43 s taken by Octree and BqR-Tree. The 100th frame image presents the rapidly rendering scene via Relief Impostors technology. When the viewpoint directly jumps to the entrance of Yangshan Station, no “white screen” or “fake system halted” phenomenon happens to the screen. It shows that the model scheduling strategy based on BRLO-Tree is more rapid and flexible than the other data structures. According to the above analysis, a reasonable explanation can be provided for Fig. 9. Meanwhile frame-time curve of this figure also shows that BRLO is better than other typical scene organizational structures in real-time performance.

Table 1. Comparison of the average performance among different tree structures

Tree structure	Seconds per frame	Triangles per frame	Support 3D GIS	Support dynamic scene
No-Tree	0.25473	46157026	No	No
Octree	0.07143	12944627	Yes	Yes
BqR-Tree	0.05759	10513124	No	Yes
BRLO-Tree	0.03147	5698315	Yes	Yes

Table 1 shows the comparison of average performance indices among different tree structures when applied to rendering 3D GIS dynamic scene of Yangshan Port. It is easy to see that BRLO-Tree structure proposed in this paper and Octree structure fully support 3D GIS dynamic scene management. However, as for the major index of the average frame rates, BRLO-Tree is better than Octree.

6. Conclusion

In order to increase the rendering speed of 3D GIS dynamic scenes, a hybrid index data structure of Block-R-Tree-Loose-Octree (BRLO) is proposed in this paper. Meanwhile, based on this scene organization structure, LOD, Relief Impostors are adopted for efficient real-time rendering of 3D GIS dynamic scenes. The test results show that BRLO-Tree structure can be applied to exhibition of large-scale 3D GIS dynamic scenes through wandering and flying patterns. Besides, it can efficiently manage a 3D model with dynamic change during interactive operation. A series of technologies based on this data structure makes the rendering picture smooth and fluent. At the same time, both the model loading time when initializing the program and the average rendering time during the operating period are greatly reduced. However, there is still an improvement space for reduction of the rendering time. For instance, real-time rendering can be accelerated through further combining with a visibility calculation method of precise occlusion judgment.

Acknowledgement: This work was supported in part by a Grant from the Starting Project of Doctor of University of Shanghai for Science and Technology (No 1D-13-309-005), Funding Scheme for Training Young Teachers in Shanghai Colleges in 2013(No slg14039), Innovation Program of Shanghai Municipal Education Commission (No 13ZZ111), the bidding project of Shanghai research institute of publishing and media (No SAYB1408) funded by the National Higher Vocational and Technical Colleges Construction Project of the Shanghai Publishing and Printing College, Incentive Program for Young Teachers in Shanghai Conservatory of Music.

References

1. Yu, Kanhua. Urban Planning Support System Based on Three Dimensional GIS. – IAES TELKOMNIKA Indonesian Journal of Electrical Engineering, Vol. **12**, 2014, No 5, pp. 3928-3935.
2. Li, Chang, W. Shi, F. Li. The Scheme and the Preliminary Test of Object-Oriented Simultaneous 3D Geometric and Physical Change Detection Using GIS-Guided Knowledge. – Telkomnika – Indonesian Journal of Electrical Engineering, Vol. **11**, 2013, No 12, pp. 7462-7469.
3. Toscano, E. S. A., V. Sorengo. 2D and 3D GIS-Based Geological and Geomechanical Survey During Tunnel Excavation. – Engineering Geology, Vol. **192**, 2015, pp. 19-25.
4. Petrov, V. A. Spatial-Temporal Three-Dimensional GIS Modeling. – Automatic Documentation and Mathematical Linguistics, Vol. **49**, 2015, No 1, pp. 21-26.
5. Mattausch, O., J. Bittner, M. Wimmer. CHC++: Coherent Hierarchical Culling Revisited. – Computer Graphics Forum, Vol. **27**, 2008, No 2, pp. 221-230.
6. Bittner, J., M. Wimmer, H. Piringer, et al. Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful. – Computer Graphics Forum, Vol. **23**, 2004, No 3, pp. 615-624.
7. Pina, J. L., F. Seron, E. Cerezo. BqR-Tree: A Data Structure for Flights and Walkthroughs in Urban Scenes with Mobile Elements. – Computer Graphics Forum, Vol. **29**, 2010, No 6, pp. 1745-1755.
8. Ahn, H. K., N. Mamoulis, H. M. Wong. A Survey on Multidimensional Access Methods. 2001, pp. 1-19.
9. Gong Jun, Zhu Qing. An Adaptice Control Method of LODs for 3D Scene Based on R-Tree Index. – Acta Geodaetica et Cartographica Sinica, Vol. **40**, 2011, No 4, pp. 531-534.

10. Chen Peng, Meng Lingkui. R-Tree Structure Appended with Spatial Topology Restrictions in 3D GIS. – Geomatics and Information Science of Wuhan University, Vol. **32**, 2007, No 4, pp. 347-349.
11. Zhu Qing, Gong Jun. An Improved Full 3D R-Tree Spatial Index Method. – Geomatics and Information Science of Wuhan University, Vol. **31**, 2006, No 4, pp. 340-343.
12. Hui Wenhua, Guo Xincheng. Research on Octree Spatial Index in 3D-GIS. – Bulletin of Surveying and Mapping, Vol. **1**, 2003, No 1, pp. 25-27.
13. Xia, Y., S. Prabhakar. Q+ rtree: Efficient Indexing for Moving Object Databases. – In: Proc. of 11th International Conference on Database Systems for Advanced Applications. United States: Institute of Electrical and Electronics Engineers Computer Society, 2003, pp. 175-182.
14. Deloura, M. Game Programming Gems 1. Charles River Media, 2001.
15. Likas, A., M. Vlassis, J. Verbeek. The Global K-means Clustering Algorithm. – Pattern Recognition, Vol. **31**, 2003, No 2, pp. 451-461.
16. Laptev, I., H. Mayer, T. Lindeberg et al. Automatic Extraction of Roads from Aerial Images Based on Scale Space and Snakes. – Machine Vision and Applications, Vol. **12**, 2000, No 1, pp. 23-31.