# An Efficient Point Cloud Management Method Based on a 3D R-Tree

**ARTICLE** *in* PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING · APRIL 2012

Impact Factor: 2.07 · DOI: 10.14358/PERS.78.4.373

**5 AUTHORS**, INCLUDING:

Yeting Zhang
Wuhan University
**31** PUBLICATIONS **97** CITATIONS
SEE PROFILE

Xiao Xie
Wuhan University
**7** PUBLICATIONS **9** CITATIONS
SEE PROFILE

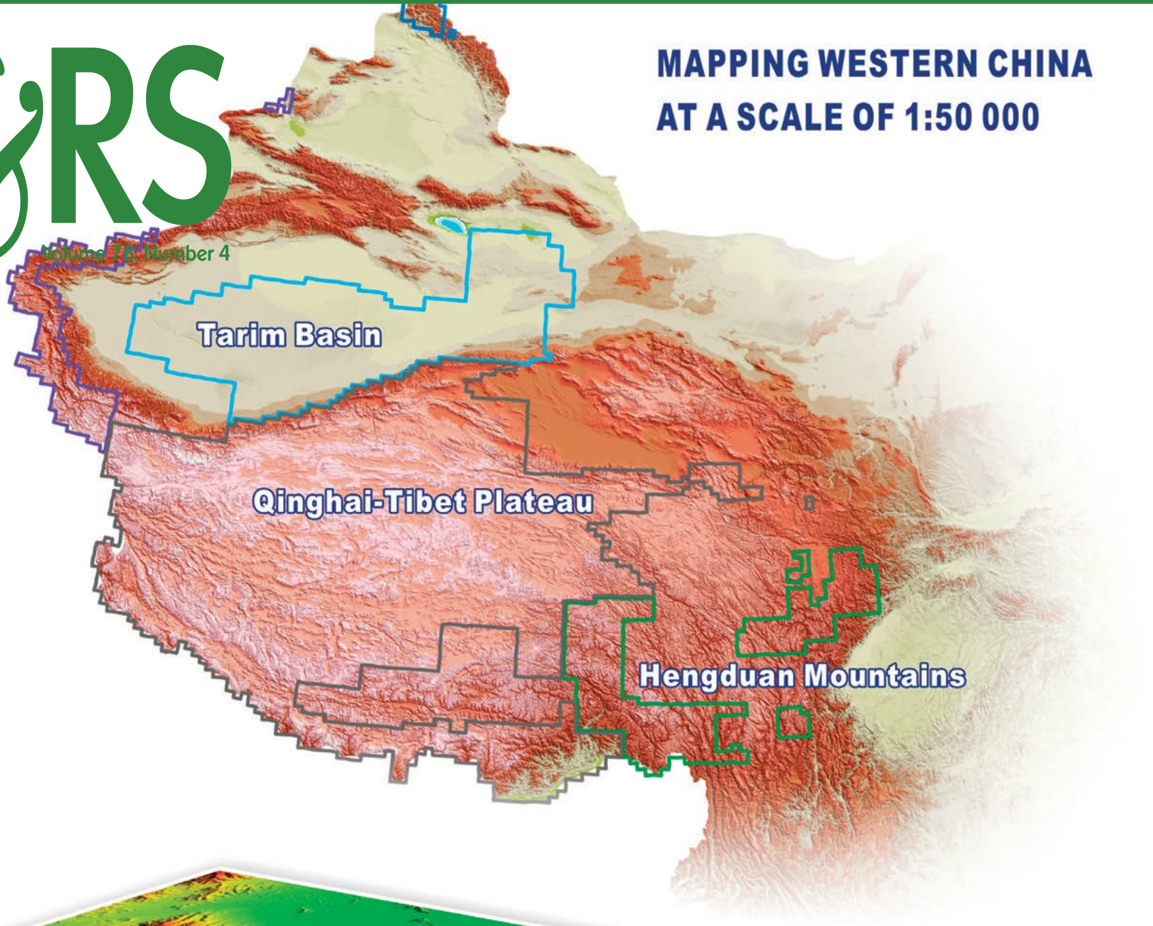
Special Issue: : Terrestrial Lidar/ASPRS Resource Directory

# PE&RS

April 2012

Volume 78, Number 4

MAPPING WESTERN CHINA AT A SCALE OF 1:50 000



PHOTOGRAMMETRIC ENGINEERING & REMOTE SENSING

*The official journal for imaging and geospatial information science and technology*

## Peer-Reviewed Articles

Is your contact information current?
Contact us at members@asprs.org
or log on to http://www.asprs.org/Member-Area/
to update your information.
We value your membership.

**PHOTOGRAMMETRIC ENGINEERING & REMOTE SENSING** is the official journal of the American Society for Photogrammetry and Remote Sensing. It is devoted to the exchange of ideas and information about the applications of photogrammetry, remote sensing, and geographic information systems.

The technical activities of the Society are conducted through the following Technical Divisions: Geographic Information Systems, Photogrammetric Applications, Primary Data Acquisition, Professional Practice, and Remote Sensing Applications. Additional information on the functioning of the Technical Divisions and the Society can be found in the Yearbook issue of *PE&RS.*

Correspondence relating to all business and editorial matters pertaining to this and other Society publications should be directed to the American Society for Photogrammetry and Remote Sensing, 5410 Grosvenor Lane, Suite 210, Bethesda, Maryland 20814-2144, including inquiries, memberships, subscriptions, changes in address, manuscripts for publication, advertising, back issues, and publications. The telephone number of the Society Headquarters is 301-493-0290; the fax number is 301-493-0208; email address is asprs@asprs.org.

**PE&RS**. *PE&RS* (ISSN0099-1112) is published monthly by the American Society for Photogrammetry and Remote Sensing, 5410 Grosvenor Lane, Suite 210, Bethesda, Maryland 20814-2144. Periodicals postage paid at Bethesda, Maryland and at additional mailing offices.

**SUBSCRIPTION.** Effective January 1, 2012, the Subscription Rate for non-members per calendar year (companies, libraries) is $420 (USA); $436 for **Canada Airmail** (includes **5% for Canada's Goods and Service Tax** (GST#135123065); $430 for all other foreign.

**POSTMASTER.** Send address changes to *PE&RS*, ASPRS Headquarters, 5410 Grosvenor Lane, Suite 210, Bethesda, Maryland 20814-2144. CDN CPM #(40020812)

**MEMBERSHIP.** Membership is open to any person actively engaged in the practice of photogrammetry, photointerpretation, remote sensing and geographic information systems; or who by means of education or profession is interested in the application or development of these arts and sciences. Membership is for one year, with renewal based on the anniversary date of the month joined. Membership Dues include a 12-month subscription to *PE&RS* valued at $68. Subscription is part of membership benefits and cannot be deducted from annual dues. Annual dues for Regular members (Active Member) is $135; for Student members it is $45 (E-Journal – No hard copy); for Associate Members it is $90 (see description on application in the back of this Journal). An additional postage surcharge is applied to all International memberships: Add $40 for **Canada Airmail,** and 5% for **Canada's Goods and Service Tax (GST #135123065)**; all other foreign add $60.00.

# An Efficient Point Cloud Management Method Based on a 3D R-Tree

**Jun Gong, Qing Zhu, Ruofei Zhong, Yeting Zhang, and Xiao Xie**

## Abstract
*Vehicle-borne laser-scanned point clouds have become increasingly important 3D data sources in fields such as digital city modeling and emergency response management. Aiming at reducing the technical bottlenecks of management and visualization of very large point cloud data sets, this paper proposes a new spatial organization method called 3DOR-Tree, which integrates Octree and 3D R-Tree data structures. This method utilizes Octree's rapid convergence to generate R-Tree leaf nodes, which are inserted directly into the R-Tree, thus avoiding time-consuming point-by-point insertion operations. Furthermore, this paper extends the R-Tree structure to support LOD (level of detail) models. Based on the extended structure, a practical data management method is presented. Finally, an adaptive control method for LODS of point clouds is illustrated. Typical experimental results show that our method possesses quasi-real-time index construction speed, a good storage utilization rate, and efficient visualization performance.*

## Introduction
The main requirements for 3D city models have been identified as including high coverage, high fidelity, and being up to date, which are precisely the main shortcomings of conventional 3D modeling approaches (Nebiker *et al.*, 2010). Vehicle-borne laser scanning systems, such as Riegl's VMX-250, Topcon's IP-S2, and Optech's Lynx, provide a good solution for the conflict between the rigorous requirements of 3D modeling and finite human and financial resources. Such mobile mapping systems combine multiple 3D laser scanners, GNSS, and IMU technologies and cameras to kinematically acquire 3D point clouds at centimeter-level density and 5 cm accuracy (Barber *et al.*, 2008; Leberl *et al.*, 2010). With accurate calibration between the scanner and the camera, each laser point can be associated to a RGB value from the optical image (Pu *et al.*, 2009). Their raw data production rates are on the order of 360 GB/hr, which means that the data volume will approach the Terabyte level within three hours of operation. The unprecedented acquisition rate and high spatial resolution pose a considerable challenge for the management and visualization of the point clouds.

As high-density 3D laser scanning becomes more popular, there is an increasing demand for automatic postprocessing of the large volumes of point cloud data that are typically produced during a project. Most postprocessing, such as filtering, classification, and feature extraction, depends greatly on the performance of the organization and management of the point clouds, which notably limits further usages for Light Detection and Ranging (lidar) (Cheng *et al.*, 2011). The computer graphics community has developed several specific data organization methods to accelerate visualization efficiency and quality, but these methods mostly focus on a single object and cannot tackle complex environments, such as those found in even small-scale laser scanning projects.

Because point cloud data are of large volume and high-resolution, adaptive visualization with level of detail (LOD) is a suitable strategy. Surfels and Qsplat are two promising approaches for LOD point rendering, but both require time-consuming preprocessing (Pfister *et al.*, 2000; Rusinkiewicz and Levoy, 2000). Their unbalanced tree structures tend to induce high tree depths and increasingly worse query performance. Improvements in the implementation of these methods have been obtained by adopting some advanced techniques, such as parallel processing, GPU rendering, and data caches (Wand *et al.*, 2008).

Managing large-scale point clouds is a very active research area in the spatial information community. Huang (2006) presents an organizational approach based on sequential Quad-trees for airborne lidar data. A segment mapping technique is adopted to randomly extract LOD lidar data, which is linked with nodes at different levels. Although it implements adaptive rendering for lidar data, random extraction cannot ensure the best simplification result. Furthermore, the danger of producing an unbalanced index structure and its subsequent impact on point cloud management performance is also a factor. Lu *et al.* (2008) introduced a nested structure integrating Octree and binary tree to manage very large point cloud data sets. However, the space partitioning is one-dimensional and cannot take into consideration the characteristics of 3D space. Kovac and Zalik (2010) also adopt a hierarchical and out-of-core approach to manage point clouds. Ma and Wang (2011) distribute large point cloud data sets among multiple servers and boost up efficiency using parallel access.

3D R-Tree adaptively adjusts index structures according to the real data. Therefore, object distribution has little influence on R-Tree, making them a promising true 3D spatial access method (Zhu *et al.*, 2007). However, when extending into 3D space, node overlapping easily induces multipath

Jun Gong is with Department of Software, Jiangxi Normal University, 99 ZiYang Road, Nanchang, 330022, P. R. China (gongjunbox@gmail.com).

Qing Zhu, Yeting Zhang, and Xiao Xie are with the State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing, Wuhan University, 129 Luo Yu Road, Wuhan, 430079, P.R. China.

Ruofei Zhong is with College of Resources, Environment and Tourism, Capital Normal University, 105 West Third Ring Road, Beijing, 100048, P.R. China.

Photogrammetric Engineering & Remote Sensing
Vol. 78, No. 4, April 2012, pp. 373–381.

0099-1112/12/7804–373/$3.00/0
© 2012 American Society for Photogrammetry and Remote Sensing

queries, which becomes the main reason for low efficiency. Gong *et al.* (2011) introduced a global optimization mechanism and 3D cluster analysis to build the 3D R-Tree index dynamically, a process that can solve the problem of serious node overlapping. This approach extends the R-Tree index structure and integrates LOD information into intermediate nodes. Theoretically, its dynamic updating and adaptive adjusting capabilities are suitable for scattered and uneven 3D point clouds. However, because of the algorithm's complexity, so far, no related literature has been published.

This paper addresses that issue and is organized as follows: following the Introduction, followed by a proposed new 3D spatial index method: 3DOR-Tree. The next Section introduces how the 3DOR-Tree structure is extended to integrate the LOD model, and next, a data organization approach based on 3DOR-Tree is presented for huge point clouds. The next Section describes the adaptive control of LODS for point cloud scenes, followed by the experimental results in comparison with other methods. Some conclusions are given in the final Section.

## A New 3D Spatial Index Method: 3DOR-Tree

In vehicle-borne laser-scanning applications, the 3D point cloud density is very high and can have a very uneven spatial distribution with large variations of height over relatively short distances. These factors introduce major challenges to spatial index construction. In one cubic meter, there may exist any range between a few points and a few hundreds of thousands of points. Traditional methods, such as Cell and Quadtree, are based on 2D space and have been extended into 3D space through 3D Cell and Octree. However, with these structures, a great deal of empty nodes may be produced because of the uneven spatial distribution of the data, which makes for very low spatial utilization and rapidly deteriorating query performance.

R-Tree construction methods can be divided into dynamic and static types. In static methods, bottom-up construction is adopted, which can ensure high efficiency and spatial utilization, but it is difficult to generate the optimal structure, and such methods do not support updating of the tree. In dynamic methods, adaptive mechanisms are applied to ensure a reasonable tree shape, but the construction performance and spatial utilization are not as good as than static methods. Although dynamic methods can better satisfy the spatial data management requirement, every point is inserted into an index structure using a series of complex operations, including node choosing and node splitting. Obviously, this is unreasonable for the very large data sets commonly found in laser scanning projects. This paper adopts a new method, which integrates both static and dynamic methods in order to achieve both high construction efficiency and adaptive updating.

The result is a hybrid spatial index method that integrates R-Tree and Octree structures (3D OCTR-Tree; abbreviated as 3DOR-Tree). Before constructing a 3DOR-Tree structure, some parameters should be set by the user. The first are the fanout parameters of the 3D R-Tree (the maximum and minimum number of tuples in each R-Tree node, *m* and *M*). The second is the Octree's convergence condition, which is the maximum number of tuples in an Octree leaf node. In our method, the Octree's convergence condition equals the maximum fanout value of the 3D R-Tree.

The flow chart of the 3DOR-Tree construction algorithm is shown in Figure 1, and key issues of this algorithm are discussed in detail below. In the process of 3DOR-Tree construction, an Octree is utilized to subdivide the 3D space quickly, and subdivision continues until the point number in the leaf nodes is less than or equal to the maximum

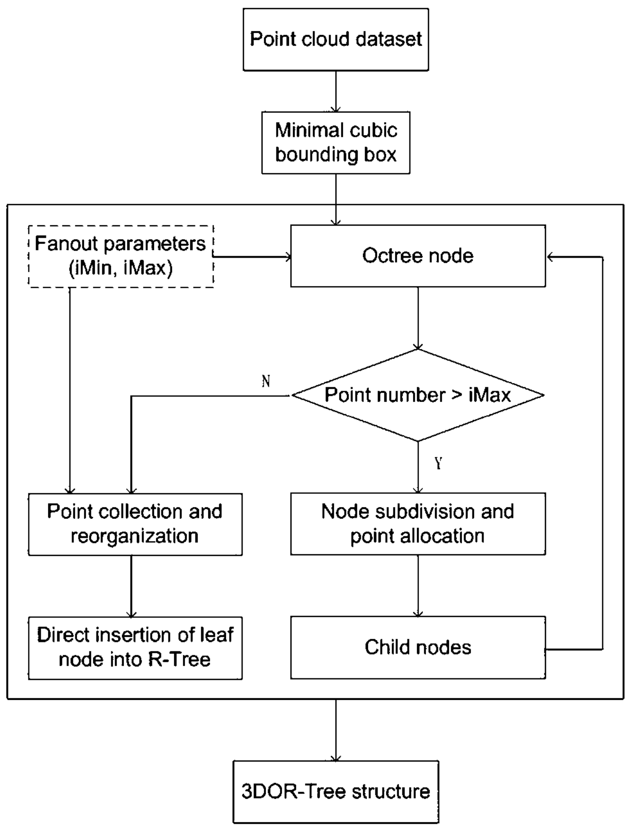

Point cloud dataset
Minimal cubic bounding box
Fanout parameters (iMin, iMax)
Octree node
N
Point number > iMax
Y
Point collection and reorganization
Node subdivision and point allocation
Direct insertion of leaf node into R-Tree
Child nodes
3DOR-Tree structure

Figure 1. Flow chart of the 3DOR-Tree construction procedure.

fanout value. If child nodes satisfy this fanout condition in the R-Tree, their bounding box will be recalculated and then inserted into the 3D R-Tree as leaf nodes. If there exist child nodes whose point number is less than the minimum fanout value, points in these child nodes are collected and sequentially reorganized into leaf nodes of the R-Tree and, finally, inserted into the 3DOR-Tree:

**Algorithm Description**: spatial index construction algorithm of one point data set;
**Algorithm Input**: R-Tree fanout parameters (*m*, *M*);
**Algorithm Output**: 3DOR-Tree structure;

- **Step 1**: Find the minimal bounding box of all points (MinX, MinY, MinZ, MaxX, MaxY, MaxZ). From the beginning of (MinX, MinY, MinZ), find the minimal cubic bounding box as Octree root node, Node. Create two new point arrays, named Array1 and Array2. Enter **Step 2**;
- **Step 2**: If point number in Node <= *M*, find its minimal bounding box and make it the root node of 3DOR-Tree, then enter Step 9; If point number in Node > *M*, enter **Step 3**;
- **Step 3**: Subdivide Node into 8 child nodes $\text{Child}_i$ ($I$ = 0,1,…,7) evenly and distribute points in Node into child nodes, then enter **Step 4**;
- **Step 4**: Clear Array1. Traverse child nodes $\text{Child}_i$ ($i$ = 0,1,…,7). If point number in $\text{Child}_i < m$, add its points into Array1. Enter **Step 5**;
- **Step 5**: Let *i*PtNum = the point number in Array1. If *i*PtNum > 2*M, enter **Step 6**; If *i*PtNum is in (M, 2*M], the points in Array1 are evenly divided into two leaf nodes and inserted into R-Tree, then enter **Step 7**; If *i*PtNum is in [*m*, *M*], the points in Array1 are reorganized into one leaf node and inserted into R-Tree, then enter **Step 7**; If *i*PtNum < *m*, insert the points in Array1 into Array2, then enter **Step 7**;

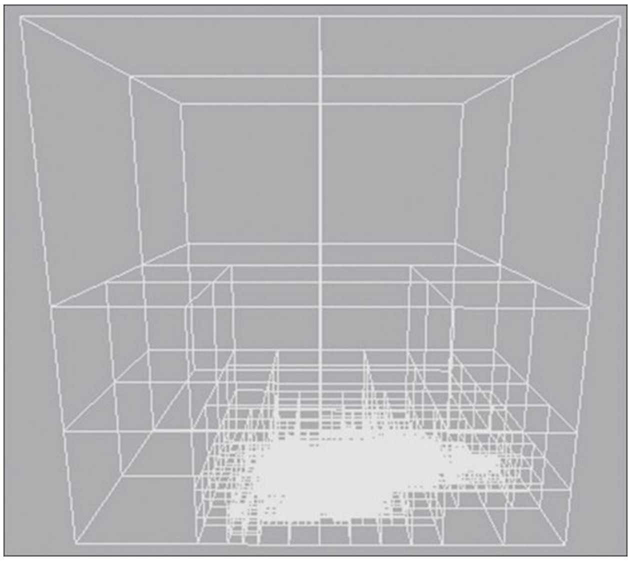


Figure 2. Leaf node layer in Octree.

- **Step 6**: Let $k$ = Rounding($i$PtNum/$i$Max). The preceding ($k$-1)*M points are divided into ($k$-1) leaf nodes and inserted into R-Tree. The number of the resting points, $i$RestNum, equals $i$PtNum-($k$-1)*$M$. If $i$RestNum = $M$, the resting points are reorganized as one leaf node and inserted into R-Tree. If $i$RestNum $> M$, divide the resting points evenly into two leaf nodes and insert them into R-Tree. Enter **Step 7**;
- **Step 7**: Traverse every child node $\text{Child}_i$ ($i$ =0,1,…,7). If the point number in $\text{Child}_i$ is in [$m$, $M$], find its minimal bounding box and insert it into R-Tree; If the point number in $\text{Child}_i > M$, let Node = $\text{Child}_i$, then enter **Step 3**;
- **Step 8**: If all Octree subdivisions end, insert the points from Array2 into R-Tree. Enter **Step 9**;
- **Step 9**: Exit.

The Octree is utilized to allocate neighboring points into the same or adjacent nodes. A node is used as an insertion unit to insert points in bulk to avoid time-consuming incremental-insertion operations, and hence, significantly boost index generation efficiency. The dynamic insertion mode allows the tree structure to have good spatial adaptation, which ensures a balanced tree structure and sound spatial utilization. Figure 2 shows the Octree structure of a point cloud data set (splitting parameter is 100). Figure 3 depicts the corresponding 3DOR-Tree structure (fanout parameters are 40 and 100).

**3DOR-Tree Extended Structure Concerning LoDs**

In generic point cloud applications, the data volume is very large, with some projects containing a billion or more points. Such data sets usually far surpass the capabilities of the average computer system, especially if real-time interaction with the data is desired. The most influential factor on system performance and data interaction is LOD; the greater the LOD to be displayed, the greater the system performance must be. Because vehicle-borne laser scanning applications require high interactivity, an efficient LOD strategy becomes a prerequisite, which means that the proper level of detail should be selected to represent point cloud scenes in real time according to view length and software/hardware performance. Previous research about integrating R-Tree and LODS tried to utilize R-Tree's hierarchical structure to realize the dual functions of object query and LOD query (Kofler, 1998; Zlatanova, 2000). The minimal bounding box of an R-Tree node is regarded as a low LOD representation in those approaches, which obviously cannot satisfy high-quality visualization.

A traditional R-Tree only manages object models in a leaf-node layer. The structure developed in this paper extends the management of object models to intermediate nodes, as is illustrated in Figure 4. A leaf-node layer manages all objects within the layer. A single representative object (for example, the one nearest to the centroid of the node) is chosen and stored in the father node as the coarser object model. With this strategy, the number of objects in a father node is equal to the number of child nodes.

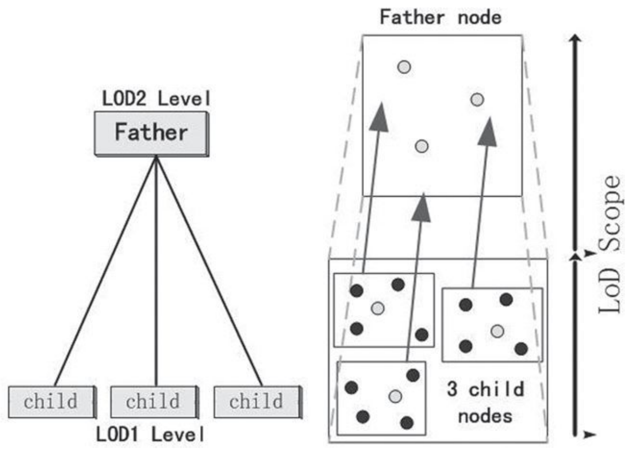


Figure 4. 3DOR-Tree structure.

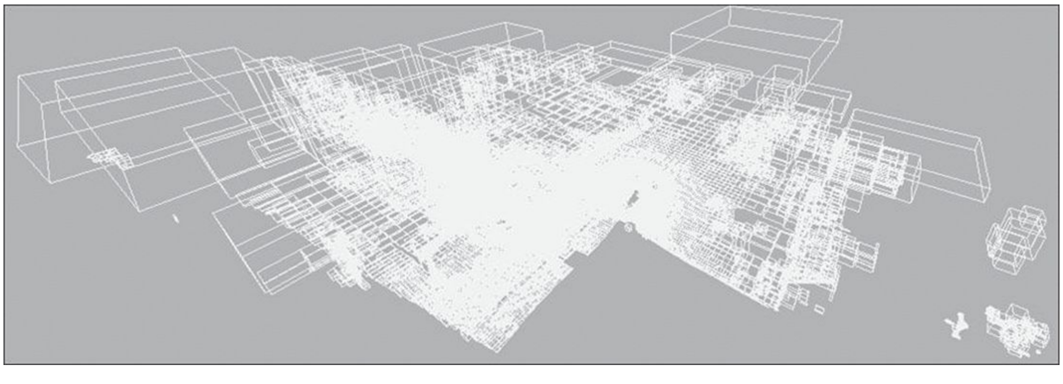


Figure 3. Leaf node layer in 3DOR-Tree.

Based on the hierarchical structure of an R-Tree, the leaf nodes denote the highest LOD, intermediate nodes denote a medium LOD, and the root node represents the lowest LOD. Every layer has one range, which includes the nearest distance and the farthest distance. The ranges of neighboring layers are seamless, and all nodes in the same layer have the same range. When view distance is in the range of one node, point models in the node will be accessed and rendered. In the mode of panoramic representation, only points in the root node are accessed. When the viewpoint is closer, the viewshed become smaller, and displayed details gradually increase. Figure 5 shows an example of the LOD representation of a point cloud.

**Point Cloud Organization Method Based on 3DOR-Tree**

Because of practical constraints related to managing very large file sizes, large-scale point cloud projects are organized in a project-cloud-point hierarchy. In the data-capturing process, vehicle-borne laser scanning systems typically partition the data into single point clouds of several million points. The size of a point cloud file may amount to several hundred MBytes. The collection of such point clouds in one project is called a "point cloud project." Our experience has shown that the point cloud project of one small village may reach dozens of GBytes. A feature of the 3DOR-Tree structure is that it can manage such large point cloud projects through implementation in commercial DBMS.

In our file organization mode, one point cloud project is a file catalog that includes many point cloud files. One point cloud file is a single binary-format file, whose format is defined in Figure 6. One single point cloud file comprises both header and entity parts. The metadata of the point cloud is stored in the header part, including version, data volume, fanout parameters, the total number of points and layers, centroid coordinates, compression flag, and the root node address. Point coordinates in the entity part are double-precision real values relative to the centroid coordinates, which allows them to be expressed as small values while maintaining the coordinates' precision. Moreover, if the range in each coordinate axis is less than 655.35 meters and centimeter-level precision is required, the coordinates can be expressed as a 2-byte short integer after being multiplied by 100, which allows data to be compressed to 25 percent of its original size.

A 3D R-Tree index structure is adopted to manage point data entities. To avoid repetitive storage, the representative points from child nodes are moved to father nodes so that the number of points in low-level nodes will be minus one.

To realize a cache mechanism, father nodes need to record the storage address of child nodes, i.e., the offset of child nodes relative to the starting address of the point cloud file. R-Tree storage can be classified as either breadth traversal storage or depth traversal storage. The breadth traversal storage sequence means that node data is stored in sequence at each R-Tree level, beginning at the root layer. Nodes are recorded to the point file layer by layer. The shortcoming of this method is that a father node cannot be directly stored with its child nodes. The depth traversal storage sequence means that the root node is first recorded, followed by its subtrees, so that every node is recorded in the same way as a root node. The problem with this structure is that sibling nodes in middle layers cannot be recorded together. Figures 7 and 8 illustrate the principles of breadth traversal and depth traversal storage, respectively.

Only point data in the top layers is required to represent the whole scene. When the viewpoint approaches local scene level, child nodes will be visited using their father node until the leaf nodes are reached. Therefore, neither of the two storage sequences can ensure that, on any occasion, points that are visited at the same time are stored together. To boost the efficiency, a hybrid scheme is adopted to record the 3DOR-Tree.

Let the leaf layer be the 1st layer. When a project is opened, the whole scene is presented. If a panoramic view requires point data in the 3rd layer and above, points in these layers should be recorded in breadth traversal sequence. Points in the lowest two layers are stored in depth traversal sequence. From a practical perspective, the total amount of data in the point cloud project determines the panoramic boundary layer. If the data volume is relatively small, then the layer can be lower, and vice versa. The principle of our approach is explained in Figure 9, in which the 3rd layer is the boundary. Nodes that will probably be visited at the same time are combined, thus benefiting from fast access time. Because father nodes record the addresses of all child nodes, a file mapping technique is easily adopted to access points in any node from the root layer to the leaf layer, which is theoretically simple and practically efficient.

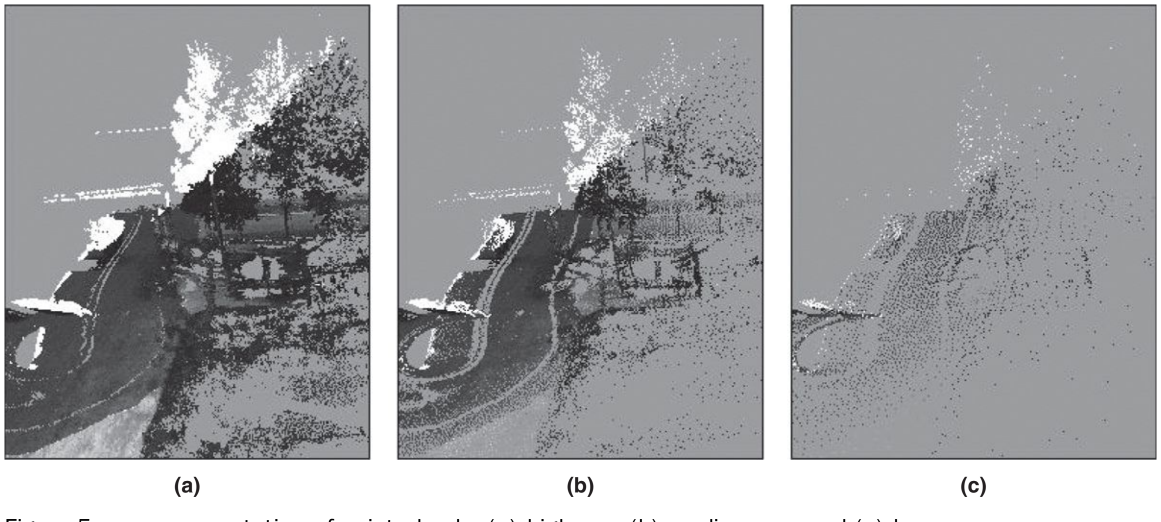


Figure 5. LOD representation of point clouds: (a) high LOD, (b) medium LOD, and (c) low LOD.

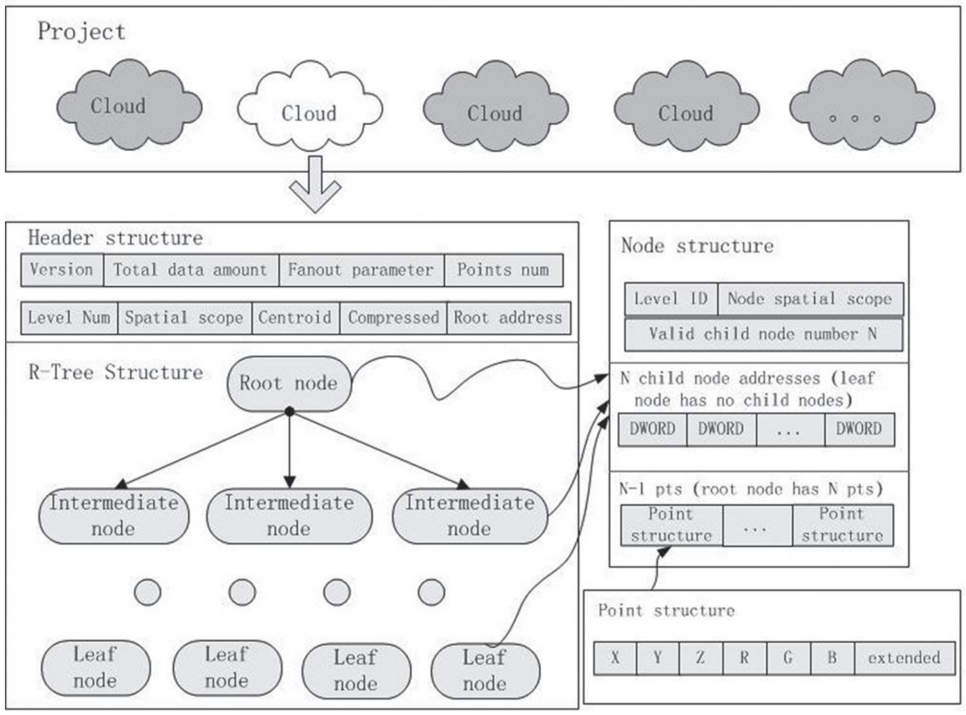


Figure 6. Data organization of massive point cloud data sets.

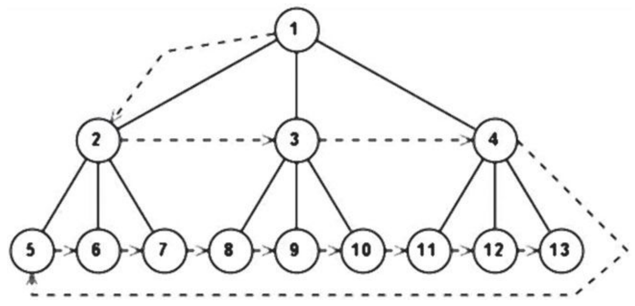


Figure 7. The principle of breadth traversal storage.

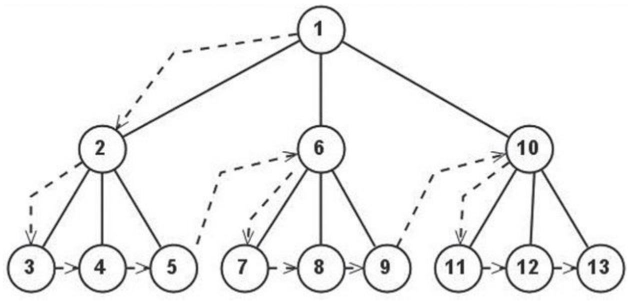


Figure 8. The principle of depth traversal storage.

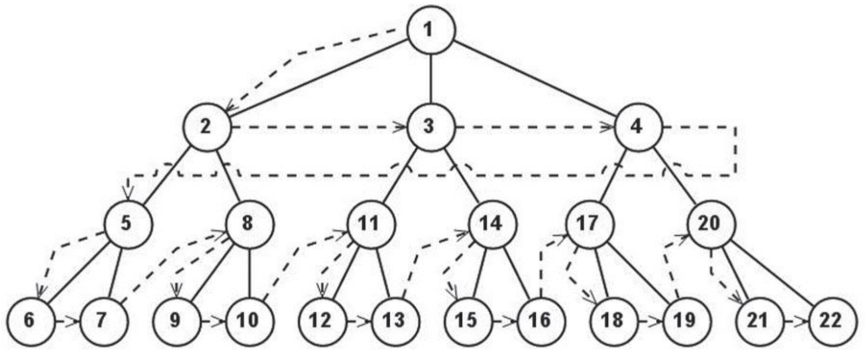


Figure 9. The principle of the hybrid storage approach.

## Adaptive Visualization Method Based on 3DOR-Tree

Software and hardware environments differ greatly, making adaptive control methods helpful when dealing with a broad range of computing systems. A critical aspect of data management is how to manage such adaptive approaches. Usually, the importance of objects can be quantitatively described by some rules, such as view length. Points are then rendered based on importance until the particular system's limit is reached.

### LoD Definition Parameters

In the 3DOR-Tree model, LoDs are distributed into the nodes in corresponding layers of the R-Tree. Controlling how various LoDs are defined is performed using a set of definition parameters, and they are here described in detail.

First, the functional range of every LoD must be properly defined to ensure a stable quantity of visible points in different viewsheds. Take an ideal 3D scene, one in which the points are evenly distributed and the nodes in all R-Tree layers are also evenly distributed, as an example. Let the view line be straight down, and locate one critical view. At the view, LoD2 will be visible as rising up, so the visible scene is fully represented in the highest LoD, LoD1. The greatest view length for the visible scene equals the maximum value of LoD1's functional range, $d$. In this way, the second critical view can be found at which the visible scene is fully represented in LoD2. At this time, the farthest view length is approximately the farthest distance of LoD2, $D$. Figure 10 denotes the scope comparisons in the above two views. The fanout parameters in R-Tree, $m$ and $M$, determine that in a specific area, the ratio of node numbers in neighboring layers is 1: $m$ ~ 1:$M$. In our method, the point number in any node also fits the fanout parameters, which means that the point number in every node is similar. The area ratio of the above two scenes is $(D/d)^2$, and the ratio of node numbers in neighboring layers is 1:$m$ ~ 1:$M$. To ensure similar node numbers in the two scenes, $m < (D/d)^2 < M$. The farthest distance of adjacent LoDs should meet the geometric relationship. In the condition of $m = 40$ and $M = 100$, $6 < D/d < 10$, which can ensure almost the same point numbers in different viewing fields.

In our method, three parameters are viewed as LoD definition parameters, e.g., the number of R-Tree levels (LevelNum), the farthest distance of the highest LoD (FarDist), and the ratio of the farthest distances of adjacent LoDs (DistFactor). Suppose R-Tree has four levels, namely, LevelNum = 4, then let FarDist = 15 m and DistFactor = 8. The functional range of the 1st level is 0 ~ 15 m, the 2nd one is 15 ~ 120 m, the 3rd one is 120 ~ 960 m, and the 4th one is 960 ~ 8000 m (the farthest distance of the final level may be unlimited).

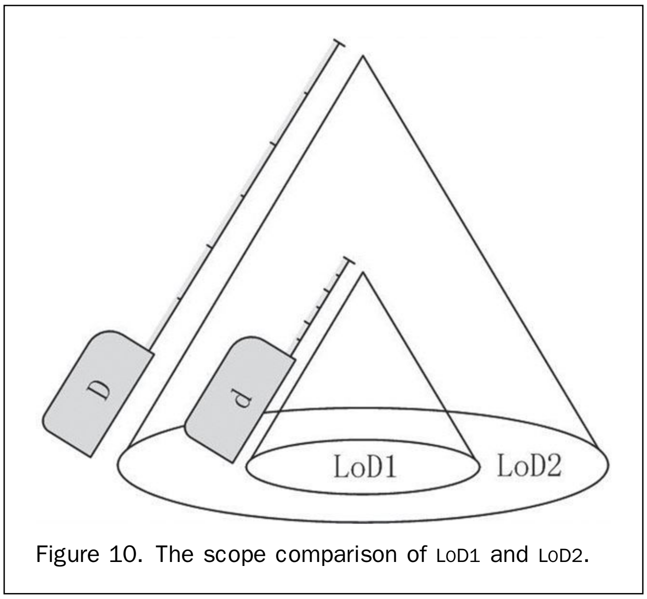


Figure 10. The scope comparison of LoD1 and LoD2.

### Adaptive Control of LoDs for a 3D Scene

According to the preceding section, the range of every level can be adjusted by changing FarDist. When FarDist is made larger, the range of every level also becomes larger. Hence, the displayed complexity of a 3D scene can be changed by adjusting FarDist. Adaptive control of the point cloud scene can be achieved by adjusting FarDist. If the scene is to be simplified, FarDist needs to be less, and vice versa.

Quantitative control of the 3D scene utilizing LoD parameters will be discussed below. When the viewpoint is close to the ground and the view line is horizontal, the viewing field contains the most objects. $S_{\text{lod1}}$ is the influential area of LoD1, and $S_{\text{lod2}}$ is that of LoD2, which are respectively calculated by Equations 1 and 2. Figure 11 illustrates the influential area of LoD1 and LoD2:

$$S_{\text{lod1}} = \pi D^2 * \alpha / 360; \quad (1)$$

$$S_{\text{lod2}} = (K^2 - 1)\pi D^2 * \alpha / 360, \quad (2)$$

where $D$ is the farthest distance of the highest LoD (FarDist), $K$ is the ratio of the farthest distances of neighboring LoDs (DistFactor), and α is the horizontal angle of the view frustum.

Both $S_{\text{lod1}}$ and $S_{\text{lod2}}$ are in proportional relationship with $D^2$. Suppose the 3D scene belongs to an ideal state in which the original point density is basically even and the point density in every level is also even. In this condition, the overlay area of every level decides the point number inside, so the processing cost is proportional to $D^2$. In real-time interaction, $D$ may be adjusted according to previous frames to deal with load variance. On the premise of a stable frame rate, the richest LoD scene can be loaded and represented in the limitation of the available main and video memory.

## Experimental Analysis

The performance of the 3DOR-Tree was tested on a sample data set of a small village captured by one vehicle-borne laser scanning system. The true color 3D data was acquired in 20 minutes and included all roads, building façades, trees, and electricity lines. The data comprise 92 point

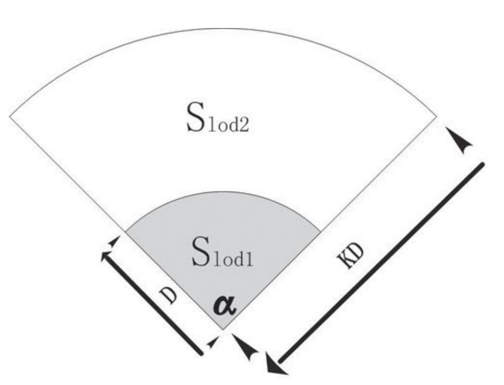


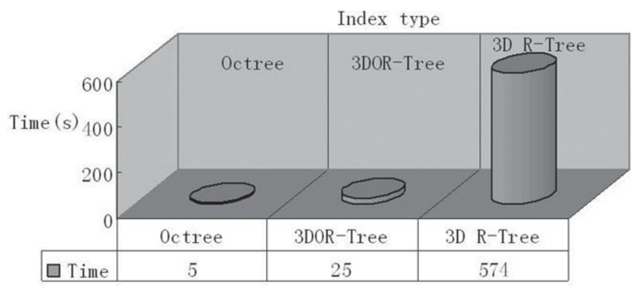
Figure 11. The comparison of the LoD1 and LoD2 influential areas.

clouds, with each cloud made up of approximately 2-million points. The total data volume was more than 16 GB and approximately 220 million points. The computing environment was very modest: a laptop computer powered by an Intel Duo T7500 CPU and 1 GB of main memory.

## Index Construction Performance

The first test was to assess the efficiency of indexing a single point cloud of 2,426,454 points from the data set. The performance of the 3DOR-Tree structure was tested against that of a standard Octree and of a dynamic 3D R-Tree. The split parameter in the Octree was set to 100, and fanout parameters in the 3D R-Tree were set to 40 and 100. These same parameters were set for the 3DOR-Tree. Two items relating to construction efficiency were compared: the time for construction and the depth of the resulting tree. Figures 12 and 13 present the results.

In terms of speed, the Octree was constructed in just five seconds, whereas the 3DOR-Tree took five times longer at 25 seconds. Both these results could be considered to constitute near-real-time performance. At 574 seconds, the dynamic 3D R-Tree took 23 times longer than the 3DOR-Tree and 115 times longer than the Octree. Inspection of the Octree's structure showed an imbalance, with the depth of leaf nodes ranging from 1 to 17. The depth of all leaf nodes in the 3DOR-Tree was 4, and the corresponding value for the dynamic 3D R-Tree was 5. It is easily understood from the test results that the construction of the 3DOR-Tree satisfies the quasi-real-time requirement. Its tree depth is balanced and is less than the other two methods as, by its nature, almost all leaf nodes are full. Hence, the number of nodes is less so that the tree depth is smaller, which is of benefit to the algorithm's efficiency. It took approximately 40 minutes to construct the 3DOR-Tree index for the whole data set with the Octree and dynamic 3D R-Tree being faster and slower proportional to the single point cloud results.



|  | Octree | 3DOR-Tree | 3D R-Tree |
|---|---|---|---|
| ▪ Time | 5 | 25 | 574 |

Figure 12. Comparison results of the spatial index construction efficiency.

## Storage Space Utilization Rate

Compared with the CPU and the main memory, the speed of external storage access is typically two orders of magnitude slower, with the result that data volume affects data scheduling performance and the user's experience. As shown in Figure 14, when compared with a standard ASCII text file, a saving of over 90 percent can be achieved. When compared with a leading commercial point cloud software, Pointools™, a saving of 20 percent was observed.

## Data Access and Visualization Performance

When comparing the time it takes to open the file, Pointools™ took four minutes before the 220 million points were ready for user interaction.

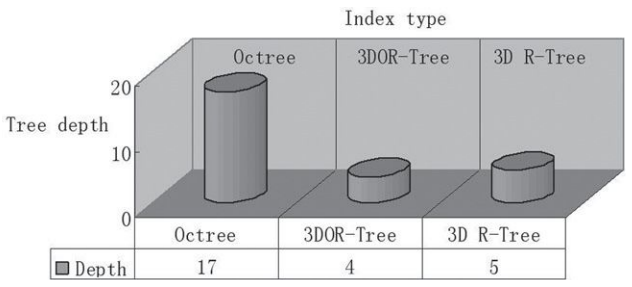
In our method, with the fanout parameters of 40 and 100, point models in the 3rd level and higher do not exceed $10^5$ points: $(2*10^8/(40*40))$. In the beginning of a project, only points in the 3rd and higher levels are required. As these levels are stored in breadth traversal sequence nodes, these layers can be sequentially accessed, resulting in only a short delay before the project can be viewed. Experimental results show that opening the project only took four seconds. Figure 15 shows the 3rd level panoramic view of the sample project.

When stored in depth-first format, the project took approximately three minutes before a user was able to interact with the point cloud.

The relatively low specification of the computing environment meant that the data cache should not exceed 200 MB and that less than 5 million points be rendered in a single frame. The data cache will be automatically cleared once the 200 MB limit is exceeded. The farthest distance of the highest LOD was kept to approximately 15 meters, which meant the amount of data scheduled in a single frame would not exceed 10 MB and would take less than two seconds to refresh. Because the rendering is part of a multithread mechanism, the visualization task in the main thread will not be influenced by such refresh times. Figure 16 is the LOD description of the point cloud scene in which high details are represented in the near distance and few details are represented in the far distance.

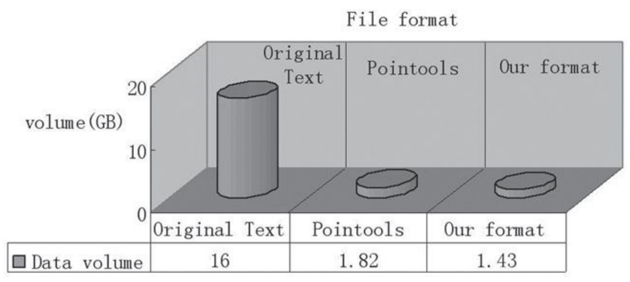
## Spatial Query Performance

3D mapping and modeling is one of the key tasks to be formed on vehicle-borne laser scanning point clouds, and 3D snapping operations are crucial to an efficient workflow. Without an efficient spatial index, searching appropriate target points from potentially billions of candidates is a time-consuming process. When tested on a sample data set, it was found that the powerful indexing capacity of the 3D R-Tree makes snapping an instantaneous operation, which satisfies user requirements for interactive mapping. Other operations, such as moving through the point clouds and connecting lines, were also performed in real time. Figure 17 shows the



|  | Octree | 3DOR-Tree | 3D R-Tree |
|---|---|---|---|
| ▪ Depth | 17 | 4 | 5 |

Figure 13. Comparison results of the tree depth.



|  | Original Text | Pointools | Our format |
|---|---|---|---|
| ▪ Data volume | 16 | 1.82 | 1.43 |

Figure 14. Comparison results of the spatial utilization.

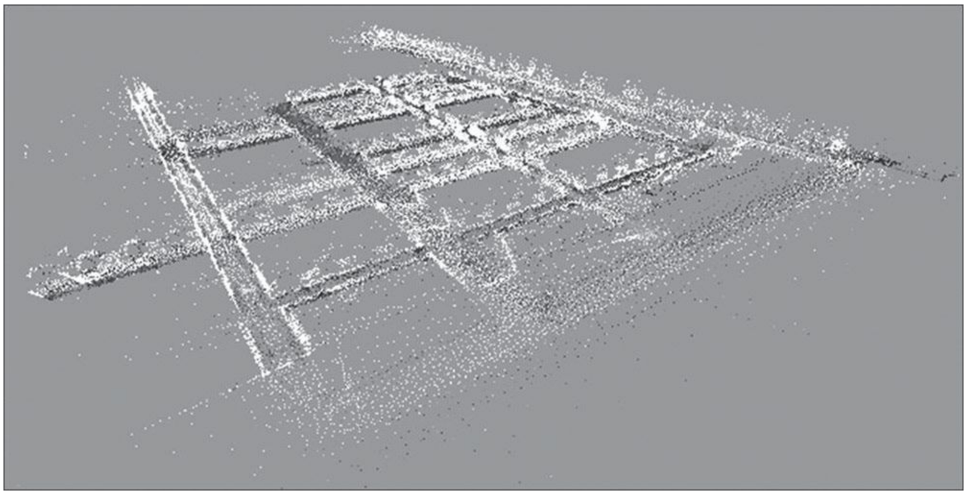


Figure 15. Panoramic view of a point project.

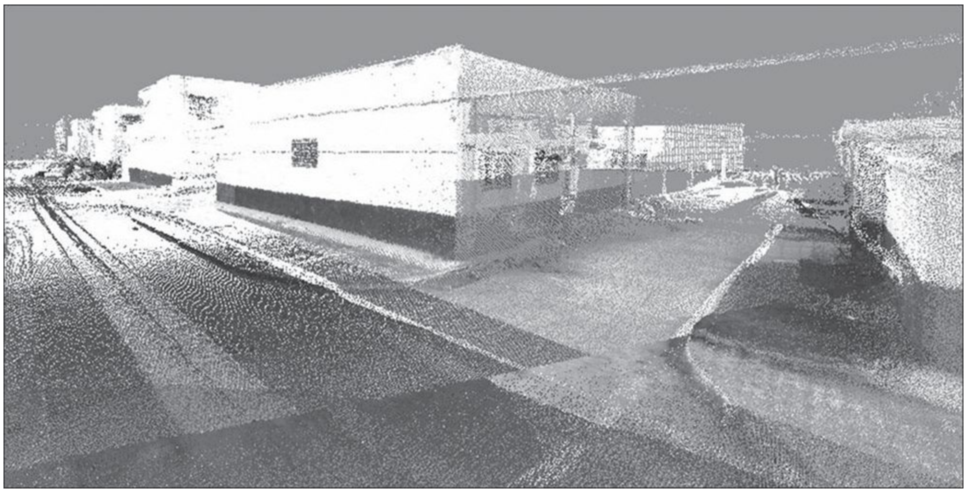


Figure 16. LoD representation in the near distance.

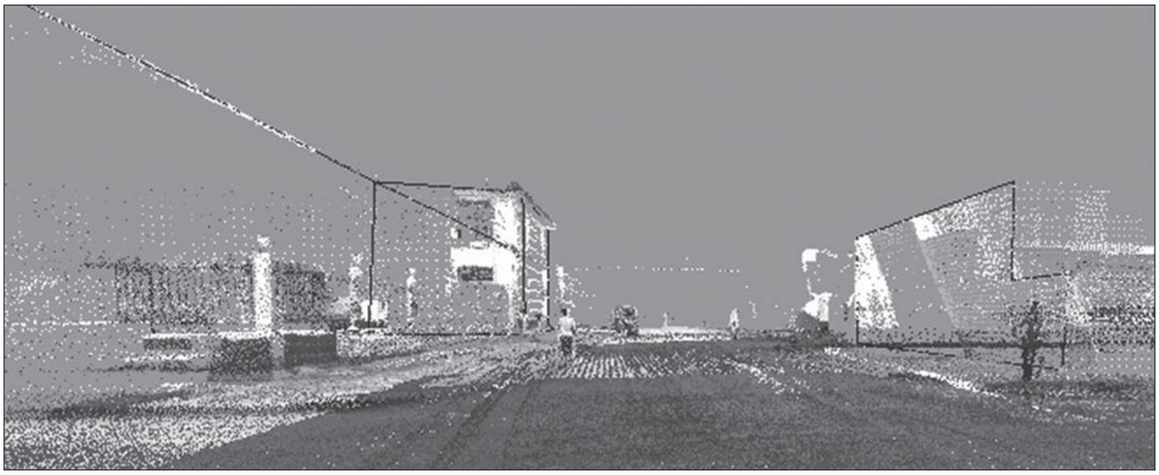


Figure 17. 3D mapping operation based on the management platform.

effects of some simple mapping operations using the developed platform.

## Conclusions

This paper introduces a new, fully 3D spatial index called 3DOR-Tree and proposes an efficient management method based on LOD for very large point cloud data sets, such as those acquired by vehicle-borne laser scanning technology. Based on a natural hierarchical structure, the LOD model management approach is useful for the adaptive visualization of massive point clouds.

With the rapid development of sensor technologies, a single point created by a laser scanning system may possess four or more properties; thus, the creation of semantic information by automatic classification and object extraction brings new challenges to point cloud management. Semantics can be viewed as another type of LOD, and further research will aim at data fusion and the semantic management of point clouds, which will realize advanced integration of panoramic images, vectors and semantics.

## Acknowledgments

This study is supported by the National Basic Research Program of China (973 Program, No. 2010CB731801 and 2011CB302306) and the National Natural Science Foundation of China (41001222 and 41021061).

## References

Barber, D., J. Mills, and S. Smith-Voysey, 2008. Geometric validation of a ground based mobile laser scanning system, *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):128–141.

Cheng, L., J. Gong, M. Li, and Y. Liu, 2011. 3D building model reconstruction from multi-view aerial imagery and lidar Data, *Photogrammetric Engineering & Remote Sensing*, 77(2):125–139.

Gong, J., Q. Zhu, Y.T. Zhang, and X.M. Li, 2011. An efficient 3D R-tree extension method concerned with levels of detail, *Acta Geodaetica et Cartographica Sinica*, 40(2):249–255.

Huang, X.F., 2006. *Research on 3D Building Model Extraction from Airborne LIDAR Data*, Ph.D. dissertation, Wuhan University, Wuhan, China, 156 p.

Kofler, M., 1998. *R-trees for Visualizing and Organizing Large 3D GIS Databases*, Ph.D. dissertation, Graz University of Technology, Austria, 156 p.

Kovac, B., and B. Zalik, 2010. Visualization of LIDAR datasets using point-based rendering technique, *Computers & Geosciences*, 36(11):1443–1450.

Leberl, F., A. Irschara, T. Pock, P. Meixner, M. Gruber, S. Scholz, and A. Wiechert, 2010. Point clouds: Lidar versus 3D vision, *Photogrammetric Engineering & Remote Sensing*, 76(10): 1123–1134.

Lu, M.Y., and Y.J. He, 2008. Organization and indexing method for 3D points cloud data, *Geo-information Science*, 10(2):190–194.

Ma, H., and Z. Wang, 2011. Distributed data organization and parallel data retrieval methods for huge scanner point clouds, *Computer & Geoscience*, 37(1):193–201.

Manolopoulos, Y., A. Nanopoulos, and A.N. Papadopoulos, 2005. *R-Trees: Theory and Applications*, Springer Press, Germany, 223 p.

Nebiker, S., S. Bleisch, and M. Christen, 2010. Rich point clouds in virtual globes - A new paradigm in city modeling?, *Computer, Environment & Urban System*, 34(6):508–517.

Pfister, H., M. Zwicker, J. van Baar, and M. Gross, 2000. Surfels: Surface elements as rendering primitives, *Proceedings of ACM SIGGRAPH 2000*, 23–28 July, New Orleans, Louisiana, pp. 335–342.

Pu, S., and G. Vosselman, 2009. Knowledge based reconstruction of building models from terrestrial laser scanning data, *ISPRS Journal of Photogrammetry & Remote Sensing*, 64(6):575–584.

Rusinkiewicz, S., and M. Levoy, 2000. QSplat: A multiresolution point rendering system for large meshes, *Proceedings of ACM SIGGRAPH 2000*, 23-28 July, New Orleans, Louisiana, pp. 343–352.

Wand, M., A. Berner, M. Bokeloh, and P. Jenke, 2008. Processing and interactive editing of huge point clouds from 3D scanners, *Computer & Graphics*, 32(2):204–220.

Zhu, Q., J. Gong, and Y.T. Zhang, 2007. An efficient 3D R-tree spatial index method for virtual geographic environments, *ISPRS Journal of Photogrammetry & Remote Sensing*, 62(3):217–224.

Zlatanova, S., 2000. *3D GIS for Urban Development*, Ph.D. dissertation, ITC, The Netherlands, 187 p.