Sensors and Materials, Vol. 35, No. 1 (2023) 87–102
MYU Tokyo

S & M 3156

Construction of Multi-resolution Spatial Data Organization for Ultralarge-scale 3D Laser Point Cloud

Haochen Huang[*]

School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China

(Received October 24, 2022; accepted January 6, 2023)

Keywords: 3D laser point cloud, high-precision vehicle-mounted 3D laser sensor, multi-resolution, double-shell Poisson disk sampling, octree, spatial data organization

The high-precision laser point cloud data obtained by 3D laser scanning technology is an important source of 3D spatial data for smart city construction. The efficient data organization of TB/PB ultralarge-scale point cloud data for urban applications is the key to point cloud data processing and visualization. Toward solving the problems of data redundancy and low storage efficiency in existing spatial data organization, we propose a spatial data organization model of a multi-resolution ultralarge-scale point cloud based on an octree and its multi-resolution point cloud construction method based on the divide-and-conquer algorithm. Firstly, a multi-resolution spatial data organization model based on an octree without redundancy is designed, which makes it easy to quickly judge the visibility of the point cloud. To obtain a high-quality rendering effect, a double-shell Poisson disk sampling method is used as a point cloud filling method to ensure constant spacing between sampling points and improve the visualization quality of point clouds. Finally, the original point cloud is partitioned by a quadtree and a process is started for each quadtree node. We propose a parallel construction algorithm for a multi-resolution point cloud to improve the construction efficiency of the algorithm when dealing with massive point clouds. In this paper, a large number of urban street point cloud data are obtained and tested using a high-precision vehicle-mounted 3D laser sensor. Experiments show that the multi-resolution spatial data organization model based on an ultralarge-scale 3D laser point cloud is reasonable and that its algorithm is efficient.

1. Introduction

Light detection and ranging (LiDAR) scanning technology, as a new 3D data acquisition technology, has become an important source of 3D spatial data for the current information construction of smart cities because of its advantages of rapidly obtaining massive point cloud data by using airborne, vehicle-mounted, and ground-based LiDAR scanners as data sources. The storage and processing of such a huge amount of point cloud data at the city level is a major

[*]Corresponding author: e-mail: huanghaochen66@126.com
https://doi.org/10.18494/SAM4190

challenge, and the reasonable organization of the data is the key to the application of point cloud engineering.

A large amount of research has been performed on the spatial data organization of massive point clouds. The B-tree, R-tree, R-variant tree, binary space partitioning (BSP) tree, grid index, quadtree, K-D tree, KDB tree, octree, and so forth have been used to establish a spatial data index, and some research achievements have been reported. Among them, the commonly used spatial data index structures include the K-D tree, quadtree, and octree. Different index structures have different advantages and disadvantages.

The spatial organization of point clouds using K-D trees can effectively improve the retrieval efficiency of point clouds. Goswami and coworkers proposed a multi-resolution point cloud organization structure based on multi-channel K-D trees, which simplifies memory management and can effectively control the depth of trees.(1–2) At the same time, they also proposed a rendering method based on point ceilings, which makes the library maintain stable and high performance in the rendering of the point cloud. This method has the function of asynchronous data and can provide continuous high-quality rendering through Levels of Detail (LOD) geographic deformation and delay mixing. Because the traditional K-D tree becomes seriously unbalanced when inserting or deleting points from the point cloud, Brown proposed a method of constructing a balanced K-D tree.(3)

Quadtrees and octrees are slightly inferior to K-D trees in terms of query efficiency, but because it is very easy to implement multi-resolution structures using quadtrees and octrees, they have wider applications in visualization. Zhi *et al.* organized and managed LiDAR point cloud data by using an improved quadtree to achieve the fast indexing of massive point clouds.(4) Wang improved the traditional quadtree structure to solve problems such as the easy redundancy of nodes and the overflow in the recursive process of tree construction. By combining a Hilbert-improved quadtree with a pyramid-PC model, massive point clouds can be organized more efficiently.(5) Wand *et al.* described an editing system for large-scale point clouds, which builds an octree and stores quantized grids at the internal nodes of the octree, with each grid cell storing an arbitrary point to represent all points at a lower level in the hierarchy.(6) The editing system is similar to that of internal octrees, where each octree node stores an octree of its own. The point data inserted into an octree node is stored at the lowest level of the internal octree. By assigning a weight to each point in the grid cell of the internal node, the point cloud can be updated when the point cloud is edited. This type of organization works well in the system of Wand *et al.*, but the disadvantage is that it is necessary to reserve at least three times the size of the original point cloud data on a disk. Yang *et al.* proposed combining a quadtree and a K-D tree to construct hybrid indexes for point clouds.(7) This method uses a quadtree pyramid model to achieve fast retrieval globally and builds a K-D tree locally to improve the query efficiency of a point cloud. This method improves the scheduling efficiency of the point cloud without reducing the precision of point cloud data. Xu studied the organization and spatial indexing of massive point cloud data and proposed an improved encoding method of the octree that adopts K-D trees for the leaf nodes of the octree to construct a hybrid index, improving the retrieval efficiency of massive point cloud data. However, the efficiency of the method decreases for point cloud data with distinctive characteristics.(8) Scheiblauer and Wimmer proposed a system based on out-ofcore selection and editing of massive point clouds, which uses a "selection octree" to spatially organize point cloud data, enabling the fast insertion and deletion of point clouds, laying the foundation for their subsequent research on the processing and visualization of massive point clouds.(9)

The following deficiencies can be seen in the organization of massive point cloud data: (1) Although the existing spatial data organization has a multi-resolution-based data storage structure, there are redundant point clouds at different levels of nodes. In addition, the spatial data organization model does not consider the problem of sampling point clouds to enable high-quality rendering. (2) The algorithms used in constructing multi-resolution data organization models are inefficient and incur high time costs when constructing the models for massive amounts of point cloud data. To this end, we propose an octree-based redundancy-free multi-resolution data organization model and a parallel construction method for multi-resolution point clouds based on a partitioning algorithm.

2. Multi-resolution Data Organization Structure and Its Parallel Construction Method

A massive amount of dense point cloud data, reaching hundreds of GB or even TB, is much larger than ordinary computer system memory, and it is impossible to visualize a whole point cloud by loading it into the memory. Therefore, it is necessary to limit the visualization of the point cloud to the visible range by using a multi-resolution structure, and only load the point cloud data judged to be visible by some rules. Visible point cloud data should contain at least the following contents:

(1) When each frame is rendered, it is located within the cone of view. In computer graphics applicaton programming interface (API) rendering, all the data transferred to the vertex buffer will be rendered. However, in 3D space, only the objects located in the cone of view can be observed, and the data outside the cone of view cannot be directly seen even if rendered. Therefore, the point cloud data outside the viewing cone should not be transferred to the vertex buffer.

(2) Each frame of the view cone is within a certain range from the camera observation point. If the far plane of the viewing cone is set at a sufficient distance, even if it is located within the viewing cone, the distant point cloud data will be drawn on the computer screen and cannot be observed because the pixels are too small or overlap. Therefore, it is unnecessary to draw the data far from the cone of view.

2.1 Multi-resolution data organization model without redundancy based on octree

On the basis of the above two requirements, it is necessary to design a data structure that is convenient for quickly judging the visibility of point clouds. The spatial data organization adopted in this paper is based on the spatial division structure of an octree, and the point cloud data are sampled from top to bottom in the construction process to produce a multi-resolution form. The multi-resolution organization of the point cloud is shown in Fig. 1, and each node

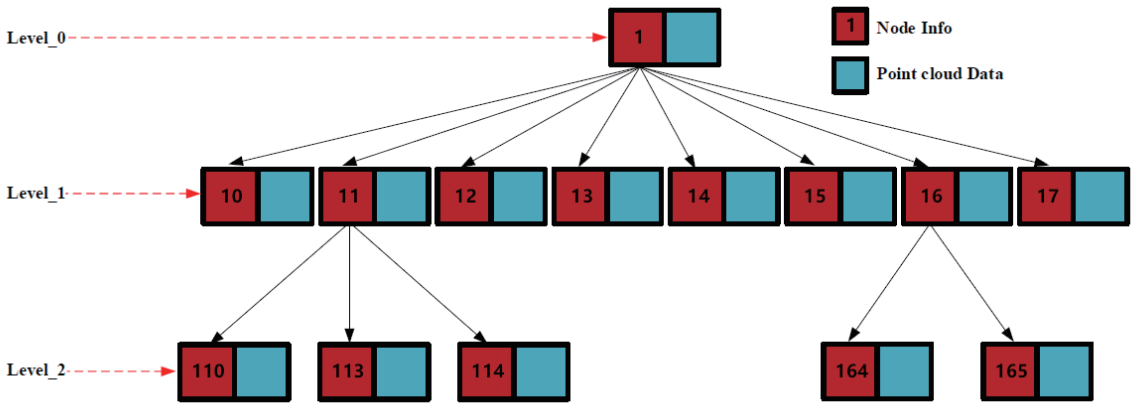


Fig. 1. (Color online) Multi-resolution organization of point clouds.

package contains two parts: point cloud data and node information. The point cloud data are the geometric data of this node, and the node information mainly includes the total number of point clouds, node bounding box, node name, node size, number of child nodes, and so forth, which provide the necessary traversal parameters for later point cloud visualization and scheduling.

The octree is built from the root node, and all nodes store point cloud data, but the points included in different levels and nodes are not repeated. That is, in the whole octree, any point only appears in one node. Therefore, in the multi-resolution structure of the point cloud, the point cloud data of the next level refine the point cloud data of all previous levels, and the nodes of all levels are merged to obtain the original point cloud data, as shown in Fig. 2(a). However, in the traditional multi-resolution structure of point clouds, the point clouds in each hierarchical node except the leaf nodes thin the original point clouds to varying degrees. The leaf nodes are spatial blocks of the original point clouds, and the union of leaf nodes can accurately express the original point clouds. The union of all nodes in the whole octree will greatly exceed the original point cloud data, resulting in a large amount of data redundancy [Fig. 2(b)].

2.2 Double-shell Poisson disk sampling

The quality of point cloud visualization depends on the uniformity of the point distribution. The distribution of the point cloud often shows density deviation, which reduces the quality of visualization. Therefore, Shu *et al.* adopted Poisson disk sampling to ensure the uniformity of point cloud sampling.(10) In addition, the spectrum sampled by the Poisson disk has the characteristics of blue-noise, so better results can be obtained when dealing with image aliasing.(11) The blue noise characteristic of the sampling results makes the point set keep a certain minimum distance between adjacent points without a large gap.(12,13) Point sets with blue-noise characteristics are considered to have high visual quality, and have achieved very good results in rendering, texture synthesis, and other applications. In this paper, a double-shell Poisson disk sampling method is used as a point cloud filling method for nodes. Double-shell Poisson disk sampling is used as a method to improve the traditional Poisson disk sampling,

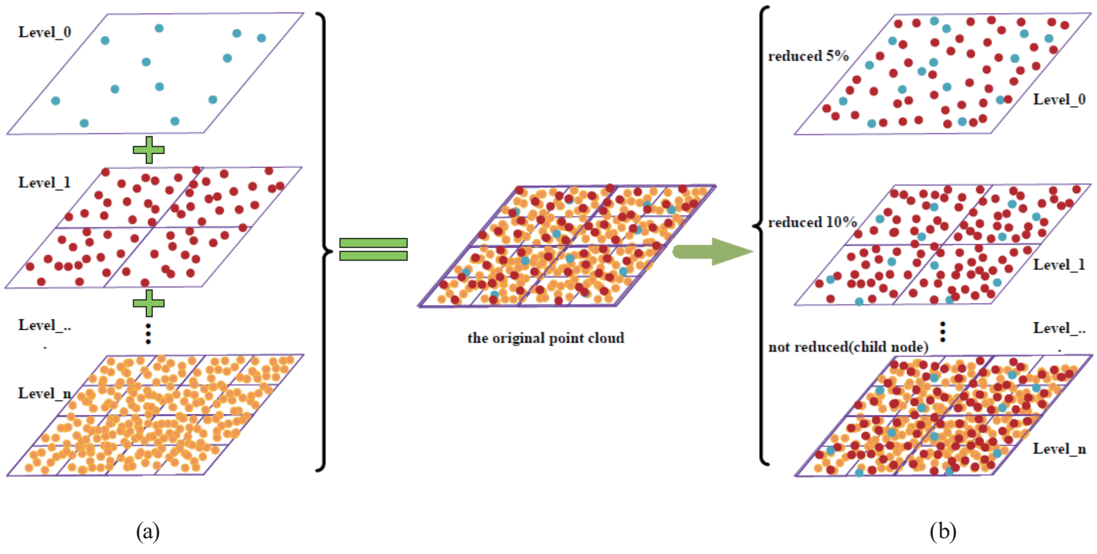


Fig. 2. (Color online) Multi-resolution structure of point clouds. (a) Point cloud multi-resolution structure in this article. (b) Traditional point cloud multi-resolution structure.

which can keep the point spacing constant and improve the visualization quality of point clouds.(14) As shown in Fig. 3, the double-shell Poisson disk adopts two concentric spheres, where the radius of the inner sphere is r and the radius of the outer sphere is R, and each point is given a selection weight priority. The point with greater weight is preferentially selected as the next center point. In each sampling process, the point located in the inner sphere is ignored, and the weighted priority of the point located between the inner sphere and the outer sphere is automatically increased. Compared with the traditional Poisson disk sampling method, the double-shell Poisson disk sampling method is advantageous and has been successfully applied to large-scale point cloud visualization.

Figure 4 shows the overall sampling process of the double-shell Poisson disk. First, the input points are sorted from inside to outside according to their distance from the center point. Then, the distance between each input point and the center point is measured, and the points between the inner and outer spheres are taken as the candidate point set for the next center point selection. Finally, the sampling result is obtained. The algorithm steps are as follows:

(1) Give the weight of each point in the point cloud an initial value of 0, that is, $priority = 0$.

(2) Define the radius r of the inner sphere and the radius R of the outer sphere ($r < R$), and randomly select a point as the first center point of the concentric sphere.

(3) Place the center point in the result container *v_accept*, ignoring the points in the inner sphere, add 1 to the weight of the points between the inner sphere and the outer sphere, and place these points in the container *v_accept* to be selected.

(4) Sort the points in the container *v_candidate* to be selected in descending order of weight, and select the point with the highest weight as the center point for the next time. If there are multiple points with the same highest weight, one of these points is randomly selected.

(5) Continue to execute step (3) until the next center point cannot be determined, and end the algorithm.

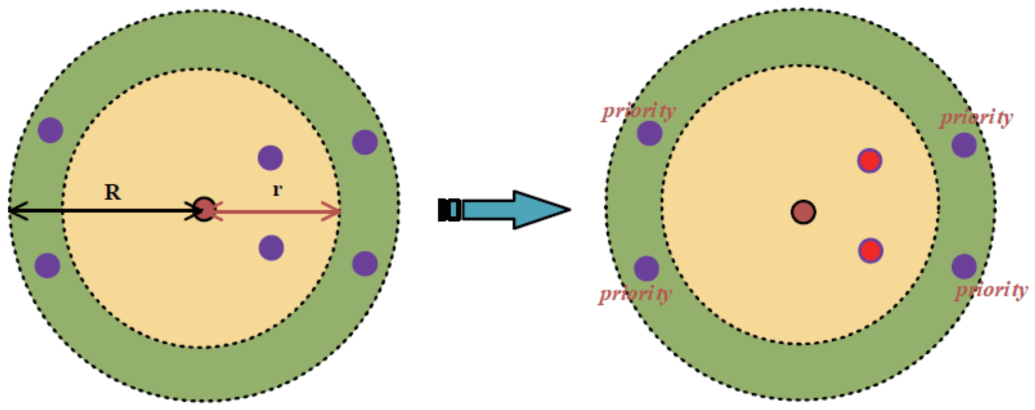


Fig. 3. (Color online) Schematic diagram of double-shell Poisson disk sampling.

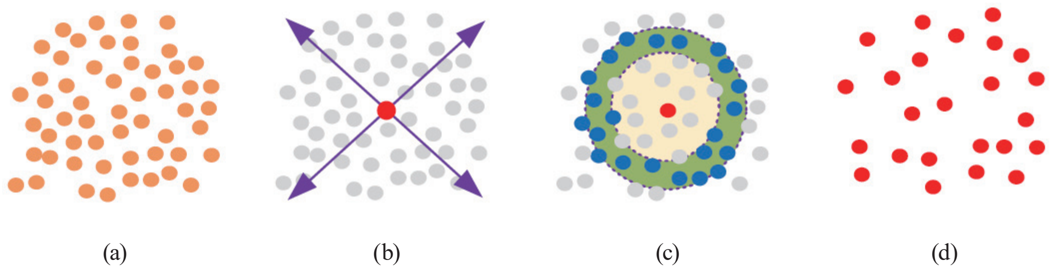


Fig. 4. (Color online) Sampling process of double-shell Poisson disk. (a) Input data. (b) Sort. (c) Distance measurement. (d) Sampling result.

In the sampling process, each point has the smallest distance from all other points. Whenever a new point of cloud data is filled into a tree node, it is necessary to calculate the distance between this point and the other points that have already been filled to judge whether this distance satisfies the minimum distance of double-shell Poisson sampling. This method can be used to process a small amount of data, but when processing large-scale point cloud data, the frequent distance calculations will greatly reduce the overall efficiency of the algorithm. To improve this shortcoming, a spatial grid is introduced in the sampling process.

As shown in Fig. 5, each node is divided into grids, and for each cell, the points falling into the cell and its neighboring cells can be conveniently recorded. When judging the distance of a newly filled point, only the distances between the point and the points in its cell and adjacent cells are calculated (that is, the distances between points in up to 27 cells are calculated). To strictly implement double-shell Poisson disk sampling between adjacent nodes, the neighborhood cell record of each cell contains cells inside and outside the node. When a cloud is filled into a tree node, it is necessary to judge not only the minimum distance between the neighboring cells in the node but also that between the cells in the neighboring nodes. Creating a spatial grid based on nodes can not only improve the efficiency of the algorithm, but also effectively avoid the problem of dense sampling results at the boundary of adjacent nodes.

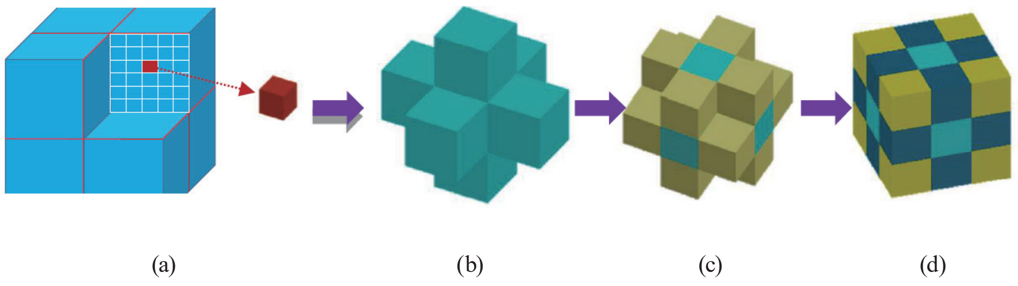


Fig. 5. (Color online) Creation of a spatial grid, (a) Current grid, (b) 6-neighborhood, (c) 18-neighborhood, and (d) 26-neighborhood.

The cell size is between the spacing of the current level and the node size. If the cell size is too small, more memory will be required and performance will be reduced. If it is too large, the cost of judging the distance between points cannot be effectively solved.(15) The grid spacing diagram in Fig. 6 shows the effect of the grid size. Whenever a point is added, the distances between the point and all the points in its cell and adjacent cells are calculated, and the points that exceed the minimum distance are added to the green cell. When the current cell receives the first point, it creates a cell instance and determines whether a neighboring cell already exists. If a neighboring cell exists, it is added to the list of neighboring cells. In Fig. 6(a), because the grid size is small and the number of points in each cell is small, a small number of distance judgments need be performed. However, because of the large number of grids, more memory will be taken up. In contrast, in Fig. 6(b), the cell size is larger; thus, more distance judgments are required every time a new point is inserted. However, during our experiment, a smaller grid size setting is required at the beginning, and the grid size is gradually increased during the experiment.

2.3 Multi-resolution point cloud construction of divide-and-conquer algorithm

Figure 7 shows the algorithm flow of multi-resolution point cloud construction. The specific steps of the algorithm are as follows:

(1) Firstly, read the point cloud data to calculate the space bounding box. If the current node is the root node, it is necessary to determine the distances between the points in the root node.

(2) Judge whether the number of read points (R_Count in Fig. 7 represents the total number of point clouds currently cumulatively read) reaches the set node processing threshold (δn). If δn has been reached, obtain all the currently read point cloud data; otherwise, continue to read the point cloud.

(3) Add points to the current node one by one and carry out Poisson distribution sampling. Considering C_d as the minimum distance from the point to its cell and points in neighboring cells, if (C_d) is not less than the resolution of the node (δd), store the point in the current node and execute step (5); otherwise, execute step (4).

(4) Judge whether a child node exists. If it exists, execute step (2); otherwise, create a child node, reduce the distance between points in the node to half that in the parent node, and continue to execute step (1).

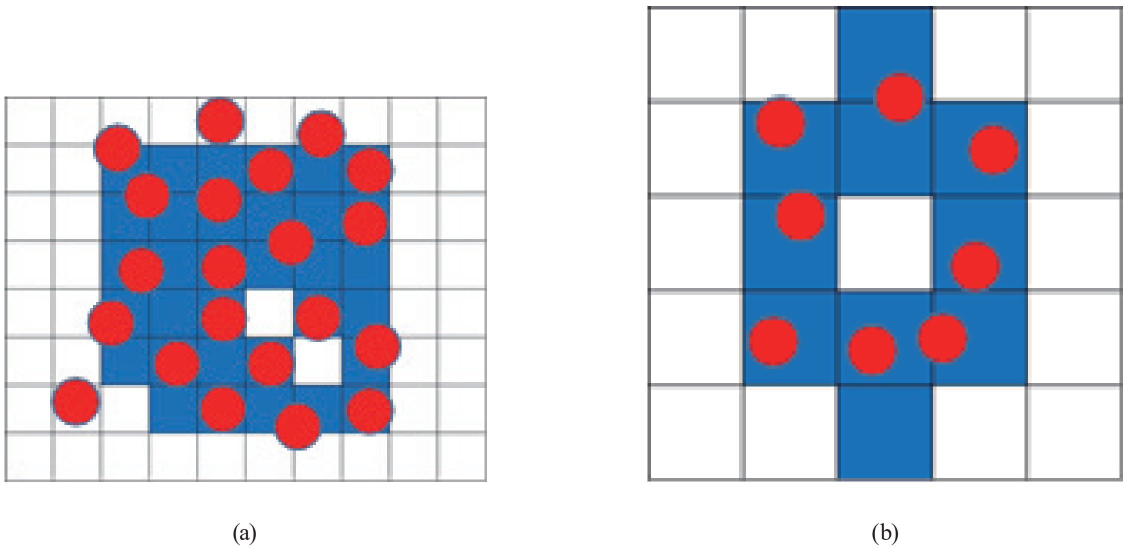


Fig. 6. (Color online) Grids of different sizes. (a) Small grid. (b) Large grid.

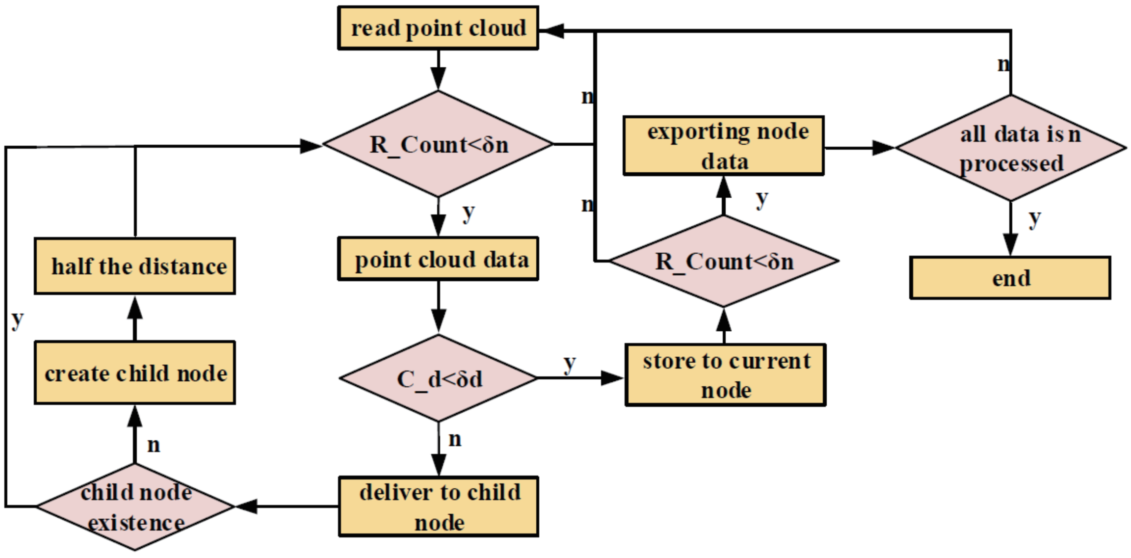


Fig. 7. (Color online) Algorithm flow of multi-resolution point cloud construction.

(5) Judge whether the data in the current node reach the storage threshold set by the node. If it has been reached, save the node and execute step (6); otherwise, execute step (1).
(6) Judge whether all data have been processed. If yes, end the algorithm; otherwise, execute step (1).

The above algorithm is described as a single construction process for a multi-resolution point cloud. It can process about 300 million points per hour, and the processing efficiency drops sharply with increasing data volume. When processing large-scale vehicle-mounted laser point cloud data or a massive city-level airborne scanning point cloud, the time cost will be extremely high. Therefore, in this paper, we propose a more efficient multi-resolution point cloud construction method based on the above-mentioned single construction algorithm.

In this paper, a multi-resolution point cloud construction method based on the divide-and-conquer algorithm is proposed. The octree is divided into several completely independent tasks, and a process is started for each task. These tasks can run in parallel on the same host. Each task generates a sub-octree, and the complete multi-resolution octree of the original point cloud can be obtained by correctly merging the multiple sub-octrees. The efficient construction of a multi-resolution point cloud based on the divide-and-conquer algorithm mainly includes three steps:

Step 1: Partition the point cloud: divide the complete point cloud into multiple small blocks in space ("blocking").

Step 2: Build an octree: build a multi-resolution point cloud for each block and create an octree for each block.

Step 3: Merge sub-octrees: Merge the sub-octrees generated by each block to obtain a complete multi-resolution point cloud structure.

2.3.1 Blocking

The point cloud is divided into several small blocks in space. The number of point clouds in each block is the same or similar, allowing parallel processing to be carried out quickly. However, note that the partition of each block should not be too small; too many small blocks will take up a large amount of memory, preventing the efficiency from being effectively improved. Blocking the point cloud results in the formation of a quadtree, which is more efficient than an octree in terms of construction efficiency. The expected number of point clouds in each block is set to N (N in this experiment is 10 million), and the quadtree is constructed recursively according to the value of N. As shown in Fig. 8, the point cloud is constructed as a quadtree, and after its construction, multiple spatial blocks of the point cloud are obtained.

2.3.2 Constructing sub-octrees

Firstly, the bounding box of the original point cloud is calculated, and the distances between points in the root node are determined according to the bounding box of the original point cloud. Then, each block is regarded as a subtask, and a multi-resolution point cloud is constructed in this process. The distances between the points in the subtask root node should be consistent with the distances between the original points in the cloud root node, and finally, multiple sub-octrees are obtained.

2.3.3 Merging octrees

As shown in Fig. 9, each block is taken as a subtask to generate a sub-octree. Because the spatial range and position of each block do not change, a same point spacing determination

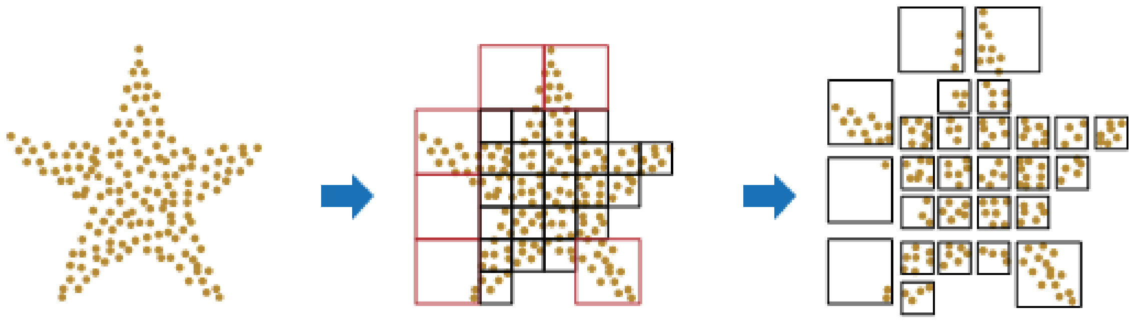


Fig. 8. (Color online) Point cloud block.

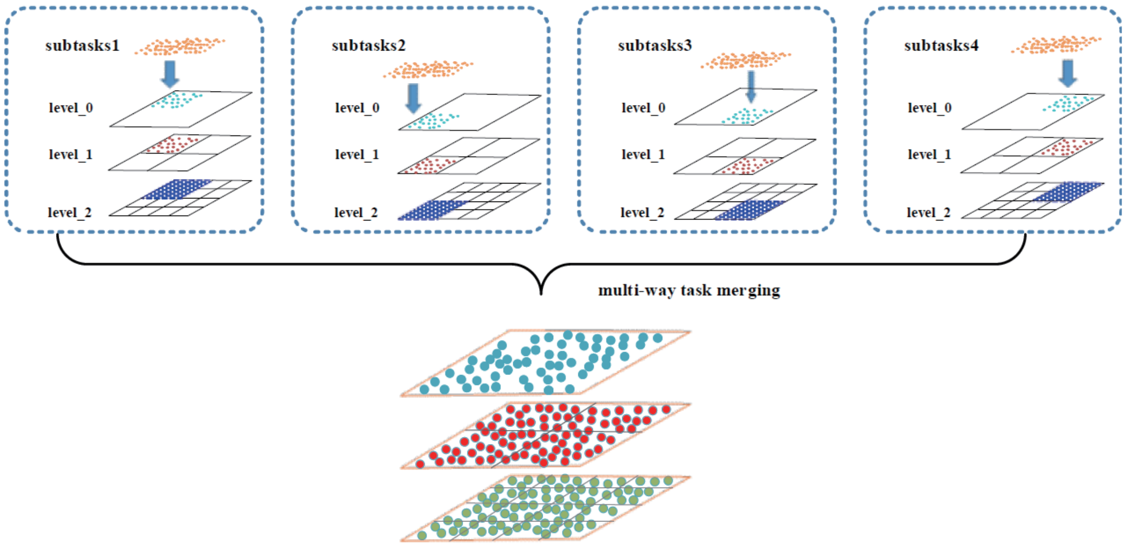


Fig. 9. (Color online) Construction of multi-resolution point clouds based on divide-and-conquer algorithm.

strategy is adopted in the construction of each sub-octree. Thus, the sub-octrees can be merged. The nodes with the same resolution in different sub-octrees must belong to the same level after merging, thus obtaining a complete multi-resolution point cloud structure. The merging of octrees does not involve complicated calculations; thus, the point clouds only need to be merged in units of nodes. This method can improve the efficiency of the multi-resolution building of massive point clouds compared with a single-resolution building.

3. Experiment and Analysis

3.1 Experimental data

In this paper, in the multi-resolution construction of a point cloud, the main data collection area consists of some sections of the Beijing–Kaifeng city road in the Daxing district of Beijing.

The data are obtained using a high-precision vehicle 3D laser sensor along a total road length of about 12.8 km, as shown in Fig. 10(a). The scanned point cloud data obtained by the vehicle-mounted laser and the GPS tracking data are used as experimental data after calibration and coordinate conversion, and the data volume is about 35 GB, or nearly 3 billion points, as shown in Fig. 10(b).

3.2 Building multi-resolution point cloud

Because the laser point cloud for the road is locally dense and globally sparse, direct multi-resolution construction will cause problems such as excessive depth of the octree and an extremely unbalanced distribution of nodes, resulting in too slow construction of the multi-resolution point cloud index. To solve this problem, we segment the point cloud obtained with the vehicle-mounted laser before multi-resolution construction. As shown in Fig. 11(a), the space is first divided along the trajectory, and the length of each segment is determined according to the selected interval number N of trajectory points. The green and purple dots in the figure represent the starting point and the end point of each segment, respectively, and certain overlapping areas are maintained between adjacent segments. The space width of the road section is set manually. Along the direction perpendicular to the trajectory line, the width of the city road is extended by D meters on both sides. The result of segmentation is shown in Fig. 11(b). On the basis of the segmentation result, the multi-resolution point cloud obtained from the vehicle-mounted laser is constructed.

To evaluate the multi-resolution point cloud construction algorithm proposed in this paper, the PotreeConverter and CesiumLab algorithms are selected for comparison, and the efficiency of each algorithm in the multi-resolution construction of the city road point cloud is evaluated. At the same time, to evaluate the performance of the three algorithms for different data volumes,

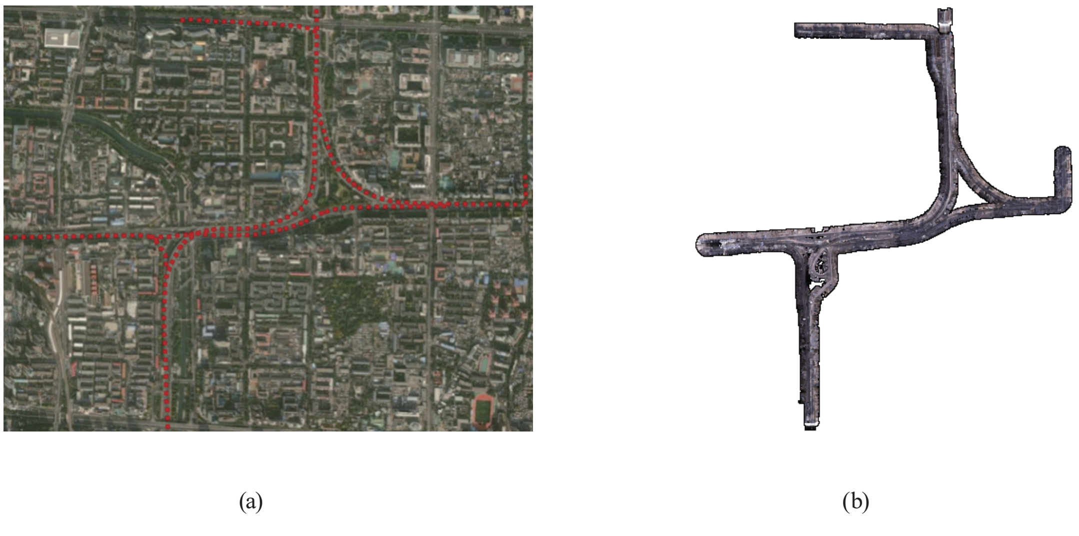
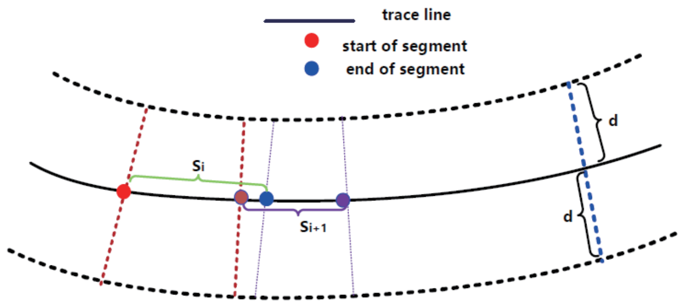
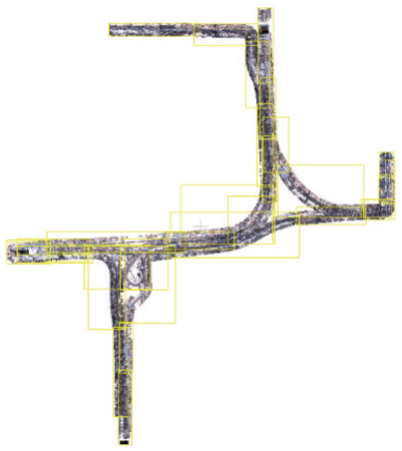

(a) (b)

Fig. 10. (Color online) Point cloud experimental data of urban road. (a) Road section used to collect point cloud. (b) Point cloud data.



(a)



(b)

Fig. 11. (Color online) MLS point cloud segmentation. (a) Segmented schematic diagram. (b) Segmentation result.

the point cloud is divided into 1 million to 100 million road segments. The statistical results are shown in Table 1.

From Table 1, it can be seen that PotreeConverter has the highest efficiency for road section 1, requiring only 3.514 s, compared with 15.504 s for CesiumLab and 5.533 s for the proposed algorithm. With increasing amount of data, the construction efficiency of PotreeConverter and CesiumLab drops sharply. To process road section 5, PotreeConverter requires 704.661 s and CesiumLab requires 1501.507 s, while the proposed algorithm requires only 224.648 s. Thus, its construction efficiency is three times that of PotreeConverter and six times that of CesiumLab. The final result of construction using the proposed algorithm is shown in Fig. 12.

Figure 13 shows the processing efficiency of each algorithm for different data. After analysis, PotreeConverter uses a single-process processing mode, which can maintain high efficiency when processing point clouds with a small amount of data. With increasing amount of data in point clouds, its efficiency drops significantly. CesiumLab, as an excellent slice-processing algorithm, has good support for vector data and oblique photography models, but its efficiency is relatively low when processing large-scale point clouds. In this paper, the concept of a distributed construction is used to process the point cloud. When processing the point cloud with a small amount of data, the efficiency is not improved compared with that of PotreeConverter because of the time required to establish the multi-processing and process communication, and sometimes the construction efficiency is slightly reduced. However, with increasing data volume, multi-process processing starts to show clear advantages, greatly improving the efficiency of constructing the multi-resolution point cloud. All the algorithms in this paper show high efficiency. In conclusion, this algorithm has high practical significance in the multi-resolution construction of large-scale point clouds.

Table 1
Time required for multi-resolution construction of highway point clouds.

Data	Number of point clouds (ten thousand)	PotreeConverter elapsed time (s)	CesiumLab elapsed time (s)	Our method elapsed time (s)
Segment 1	113	3.514	15.504	5.533
Segment 2	1401	25.073	149.643	18.152
Segment 3	6567	163.863	739.193	77.221
Segment 4	25505	730.737	1689.478	241.482
Segment 5	22708	704.661	1501.507	224.648

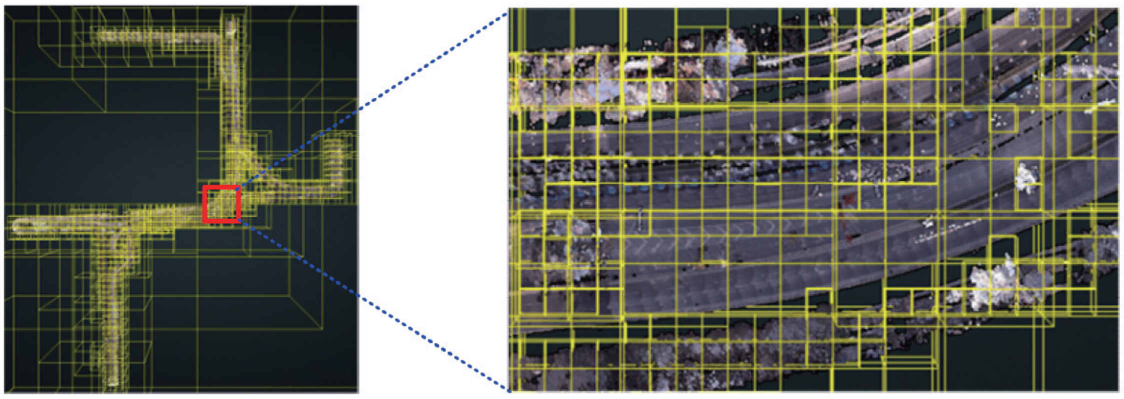


Fig. 12. (Color online) Result of point cloud multi-resolution construction.

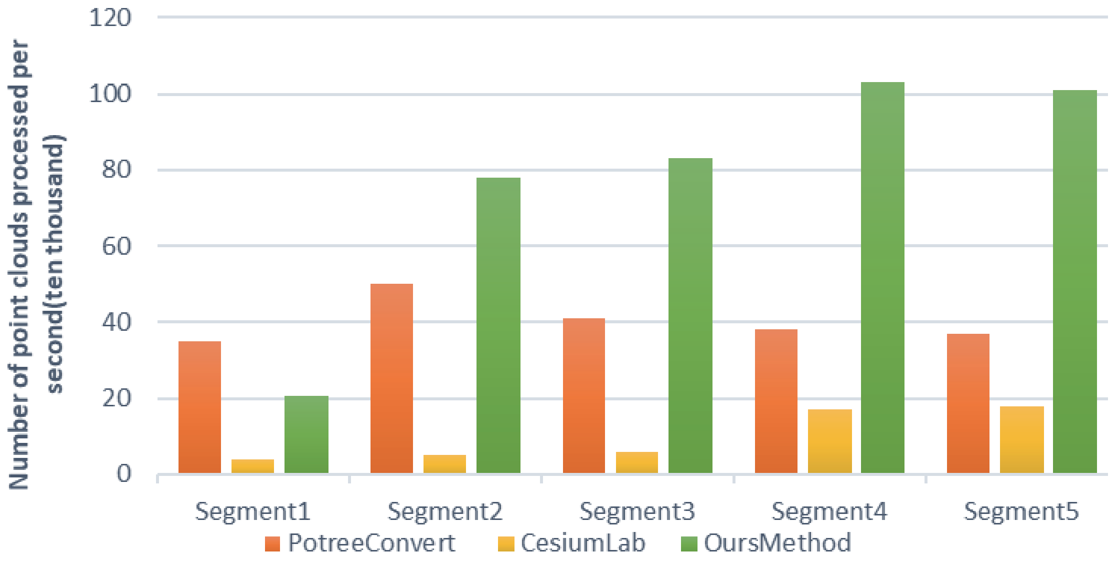


Fig. 13. (Color online) Processing efficiency of various algorithms.

3.3 Visualization performance of multi-resolution point cloud

Figure 14 shows the visualization of a point cloud of nearly 3 billion points. From the overall overview to the local details, the detailed features such as lane lines, street lamps, and guardrails of the road are clearly and smoothly displayed.

The visualization performance is usually evaluated on the basis of the number of frames rendered per second, that is, the frame rate (FPS). The higher the FPS, the smoother the picture and the better the rendering performance of the system. Figure 15 shows FPS in interactive operations such as rotation, translation, zooming, and roaming when multi-resolution point cloud visualization of the road is performed.

As can be seen from the figure, the minimum frame rate for visualization of the road point cloud is about 24 FPS, and the frame rate will be increased for the zoom-out operation, that is, the viewpoint moves away from the point cloud. The maximum frame rate is about 60 FPS.

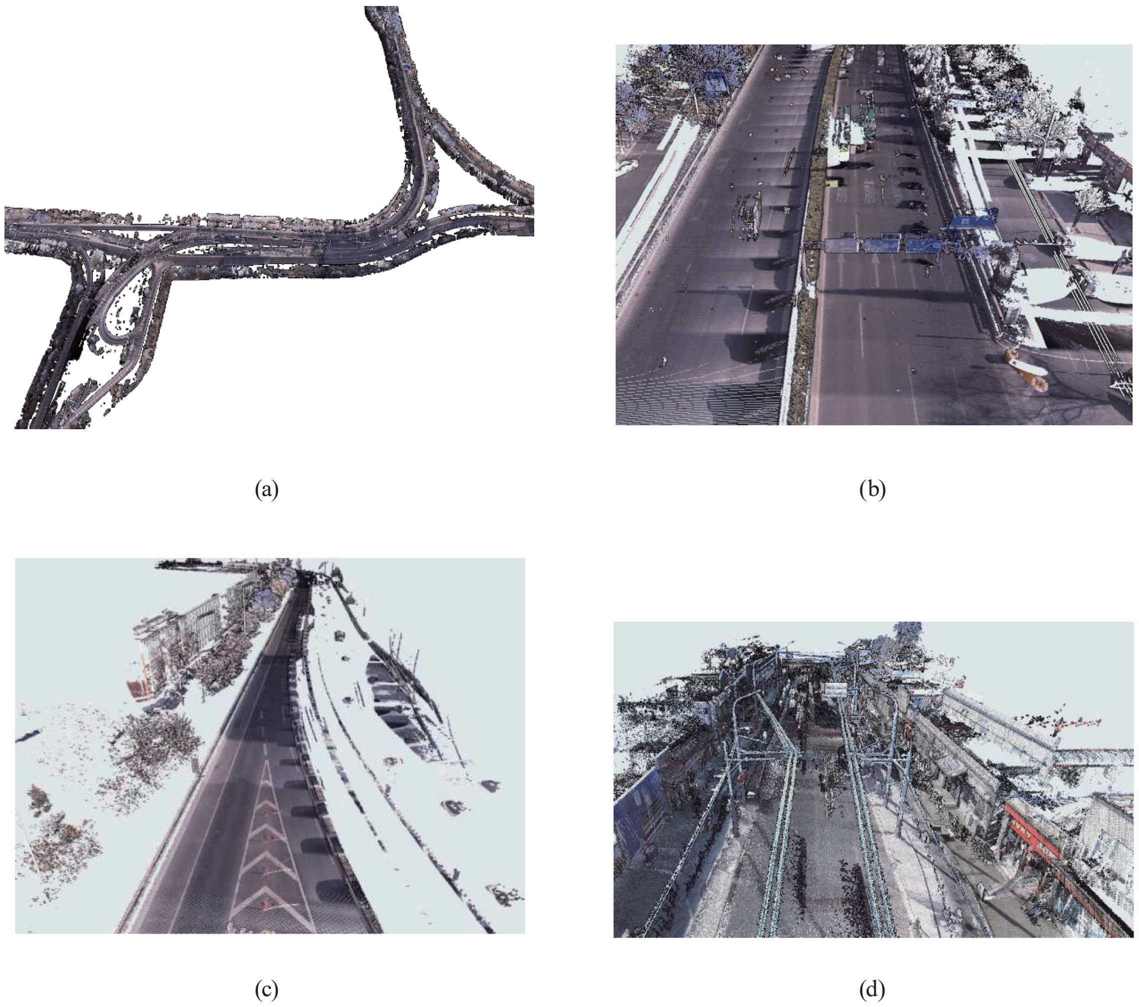


Fig. 14. (Color online) Point cloud visualization of road (about 3 billion points). (a) Global view. (b) Partial view 1. (c) Partial view 2. (d) Partial view 3.

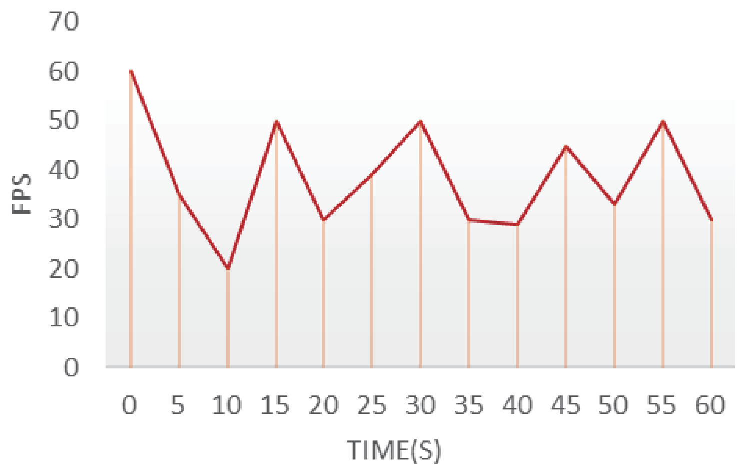


Fig. 15. (Color online) FPS statistics of point cloud visualization.

When the frame rate drops sharply, it usually means that the data of visible nodes are being requested from the server for drawing, and when the drawing is finished, the frame rate starts to rise. The average frame rate is above 30 FPS, which fully meets the requirements of smooth rendering.

4. Conclusion

Focusing on very large scale 3D laser point cloud data, which are important 3D spatial data, we proposed solutions to increase the rationality of the existing spatial data organization model and the efficiency of its construction method. Starting from the rapid scheduling of massive data visualization, a multi-resolution spatial data organization model based on an octree without redundancy is proposed, which effectively enables the rapid visualization of different levels of details based on the viewpoint. To meet the demand for high-quality visualization, double-shell Poisson disk sampling based on the constant distance of point clouds is adopted to ensure the rendering quality of multi-resolution point cloud visualization. Finally, a parallel algorithm for the multi-resolution point cloud data organization model based on quadtree partition is proposed, which significantly improves the construction speed. Experiments clearly show that the proposed data organization model and algorithm are advantageous and have superior performance to traditional methods.

References

1 P. Goswami, R. Mukhi, and E. Gobbetti: Visual Comput. **29** (2013) 69. https://doi.org/10.1007/s00371-012-0675-2
2 P. Goswami, Y. Zhang, R. Pajarola, and E. Gobbetti: 2010 18th Pacific Conf. Computer Graphics and Applications. 93-100
3 W. Bi, J. Ma, X. Zhu, W. Wang, and A. Zhang: Appl. Soft Comput. **131** (2022) 1568. https://doi.org/10.1016/j.asoc.2022.109780
4 X. Zhi, Z. Lin, G. Su, and L. Zhong: Comput. Eng. and Appl. **46** (2010) 71. https://doi.org/10.3778/j.issn.1002-8331.2010.09.021

5 L. Wang: Master's Thesis, Beijing University of Technology. 2016. https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201701&filename=1016785801.nh
6 M. Wand, A. Berner, M. Bokeloh, P. Jenke, A. Fleck, M. Hoffmann, B. Maier, D. Staneker, A. Schilling, and H. Seidel: Comput. Graphics **32** (2008) 204. https://doi.org/10.1016/j.cag.2008.01.010
7 J. Yang, H. Liu, and P. Lin: Bull. Survey Map. **7** (2014) 18. https://doi.org/10.13474/j.cnki.11-2246.2014.0216.
8 P. Xu: Master's Thesis, Nanjing Normal University. 2013. https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201401&filename=1013338533.nh
9 C. Scheiblauer and M. Wimmer: Comput. Graphics **35** (2011) 342. https://doi.org/10.1016/j.cag.2011.01.004
10 S. Yanai, R. Umegaki, K. Hasegawa, L. Li, H. Yamagushi, and S. Tanaka: 2017 Int. Conf. Culture and Computing (Culture and Computing) 13–19. https://doi.org/10.1109/Culture.and.Computing.2017.19
11 F. Zhang: Master's Thesis, Dalian University of Technology. 2017. https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201801&filename=1017821926.nh
12 D. Yan, J. Guo, B. Wang, X. Zhang, and P. Wonka: J. Comput. Sci. Technol. **30** (2015) 439. https://doi.org/10.1007/s11390-015-1535-0
13 A. Parada-Mayorga, D. L. Lau, J. H. Giraldo, and G. R. Arce: IEEE Trans. Signal and Information Progressing over Networks **5** (2019) 554. https://doi.org/10.1109/TSIPN.2019.2922852
14 S. Yuto, N. Yukihiro, L. Liang, H. Kyoko, N. Satoshi, and T. Satoshi: Commun. Comput. Inf. Sci. **1094** (2019) 161. https://doi.org/10.1007/978-981-15-1078-6_14.
15 P. van Oosterom , S. van Oosterom, H. Liu, R. Thompson, M. Meijers, and E. Verbree : ISPRS J. Photogramm. Remote Sens. **194** (2022) 119. https://doi.org/10.1016/J.ISPRSJPRS.2022.10.004

About the Authors

Haochen Huang is studying for a bachelor's degree in computer science at University of Science and Technology Beijing. His research interests are in 3D laser point clouds and high-performance data structure design. (huanghaochen66@126.com)