Efficient Randomized Hierarchy Construction for Interactive Visualization of Large Scale Point Clouds

Lai Kang
College of Systems Engineering
National University of Defense Technology
Changsha, China
kanglai123@yeah.net

Jie Jiang
College of Systems Engineering
National University of Defense Technology
Changsha, China
jiejiang@126.com

Yingmei Wei
College of Systems Engineering
National University of Defense Technology
Changsha, China
weiyingmei@nudt.edu.cn

Yuxiang Xie
College of Systems Engineering
National University of Defense Technology
Changsha, China
yxxie@nudt.edu.cn

***Abstract*—Point cloud is widely used in various applications like 3D geographical information system (GIS), cultural heritage preservation, urban planning, etc. Most of these applications require interactive visualization of massive point cloud, which is challenging since their sizes are usually very large. This paper presents a method to construct a hierarchical data structure for point cloud data organization and real-time rendering, with an emphasis on speeding up the construction processing. The overall pipeline consists of the following three steps: first, the spatial extent of the whole dataset is divided into nested blocks; second, data in each block is reorganized using a octree based on random subsampling in a parallel fashion; finally, octree of all blocks are merged into a consistent hierarchy. The effectiveness and efficiency of the above approach was demonstrated by applying it to a set of point clouds of varying sizes reconstructed using photogrammetry pipeline.**

Keywords—point cloud, large scale, hierarchy, visualization, photogrammetry

I. Introduction

Modern remote sensing techniques such as laser scanning and photogrammetry facilitate the acquisition of huge amounts of highly accurate 3D point cloud data. Such data has been widely used in a variety of applications, like 3D geographical information system (GIS), cultural heritage preservation, urban planning, etc. Usually, efficient visualization of such data is desirable since users need to explore and inspect the scene interactively, which poses challenge in practice due to the large sizes of point cloud dataset. Compared with mesh representation, more data is needed for point cloud to represent the same geometry. Therefore, another straightforward option is to convert point clouds into meshes and organize them using level of detail (LOD) algorithms for visualization. However, mesh reconstruction can be time consuming and costly for large point cloud data. More importantly, such operation may cause loss of information due to the substantially lower resolution of triangulated meshes.

Generally, the visualization of massive dataset is difficult since the size of data can easily exceed the main memory of computer. Even in the case the whole dataset can be loaded into the main memory, display the data with multiple attributes (e.g., position, normal, color, etc.) requires huge computational resources, thus visualization of such data at interactive frame rates on a personal computer is not possible for large dataset. In both cases, data organization and scheduling need to be handled carefully. As for point cloud data, in order to avoid decreasing point density (i.e., subsampling the data) or showing only a small portion of the whole dataset at once, it is essential to reorganize the original dataset with a multi-resolution data structure.

II. Related work

The first pipeline for rendering large scale point clouds do not fit in main memory is called Qsplat [1], which builds a multi-level point-per-node data structure based on point-sampling meshes. This solution involves several data preprocessing such as normal vectors computation. Layered Point Clouds (LPC) [2] makes the assumption that point data is uniformly sampled and stores a set of points in each node. The visualization performance of LPC is accelerated using graphics processing unit (GPU). Another point cloud rendering method named Instant Points [3] does not depend on any sampling distribution or involve any pre-processing like normal computations. This approach uses an octree data structure in which the hierarchical nodes also contain multiple points. Besides octree data structure, some other data structures are also investigated in the literature, e.g., Goswani et al. [4] propose a different data structure using multi-way (balanced) kd-tree.

From the point view of implementation, even though some of the previous point cloud rendering solutions are very efficient, most of them are desktop-based systems. However, the rising popularity of WebGL and the availability of cloud computing or storage resources are changing the way in which point cloud data are consumed [5]. In fact, a few years ago point cloud visualization was limited to desktop-based solutions, but after the introduction of WebGL web renderers have become more and more popular. One of them is Potree [6] which uses a multi-resolution octree data structure to deal with large datasets. The data structure used in Potree is constructed in a preprocessing stage. The computation required to create the multi-resolution octree data structure is not optimized, which limits the usability for massive datasets.

In this paper we present a solution for speeding up the creation of point cloud hierarchy by using a divide-and-conquer pipeline, which splits the spatial extent of the whole dataset into smaller nested blocks that can run in parallel and can be easily combined into a consistent hierarchy with little additional file operation overhead. Moreover, a random subsampling strategy is employed to accelerate the creation of octree for each block.

978-1-7281-4528-0/19/$31.00 ©2019 IEEE
DOI 10.1109/DSC.2019.00096

III. Method

In this section, we present details on the construction of point cloud hierarchy and how it is used in the rendering stage.

A. Randomized point cloud hierarchy construction

The basic data structure used in this paper is a multi-resolution octree, which is similar to the data structure used in Potree. Within this data structure, the original point cloud is subsampled without adding any new points to the dataset, thus it requires no additional disk space and enables operations like point selection and measurements on the original data at any level. Specifically, a variation of the modifiable nested octree (MNO) [7] structure combined with random subsampling is used in this paper, resulting in an efficient and flexible point cloud hierarchy. Since each point of the original dataset is assigned to exactly one octree node, combining all the points in all nodes returns the original dataset. An illustration of the hierarchy structure is shown in Fig. 1.

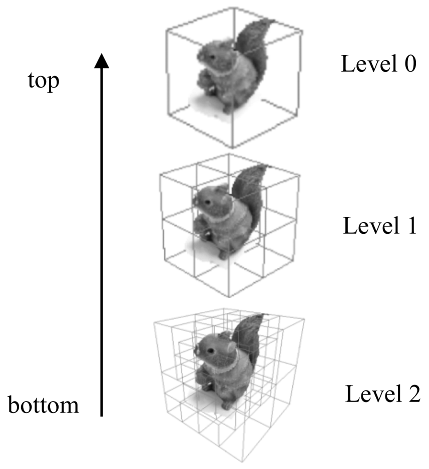


Figure 1. A three-level dense point cloud hierarchy.

The resolution of a node is defined by the number of points in a node, which in practice is specified by a minimal distance threshold D_s between points, iteratively decreases by half for a lower level. A smaller D_s leads to a higher amount of points in each node, a lower number of nodes overall, and a shallower tree depth. The optimal value of the distance threshold depends on the performance of hardware, like CPU and GPU. In this work, D_s is fixed to $C_s/128$, where C_s is the corresponding size of the bounding box for each node. In order to generate as much as possible uniformly spaced subsamples while satisfying the constraint that the distance between any two points is not smaller than D_s, we use a cube-level random subsampling strategy for simplification and speedup considerations. In particular, each node is divided into smaller cubes of size D_s for each node, then a point within the cube is randomly chose to store in the node. The resulting data shows visually pleasing patterns and it also provides good coverage with a certain number of points. A 2D illustration of the above operation is shown in Fig. 2, where the dots in (a) and (b) indicate the locations of the original and subsampled points, respectively

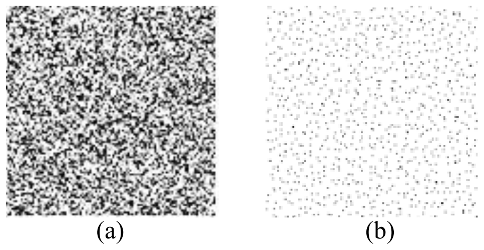


Figure 2. 2D illustration of cube-level random subsampling.

While the above reorganization algorithm is simple to implement and fast enough for middle scale point clouds, such preprocessing can be time consuming for massive datasets. To this end, this work uses a divide-and-conquer approach to speed up the construction of point cloud hierarchy. As illustrated in Fig. 3, the processing consists of the following three steps. First, the spatial extent of the whole hierarchy is computed and a tiling operation is performed to generate nested spatial extent of the hierarchy. Next, a set of independent task is generated according to the spatial hierarchy and the number of available computational threads. All the tasks are executed in parallel with the aforementioned octree construction and subsampling algorithms. Finally, the different octrees generated from the various independent tasks are merged into a single consistent one. This merging operation is fast since it only involves moving and combining files and modifying indexing information.

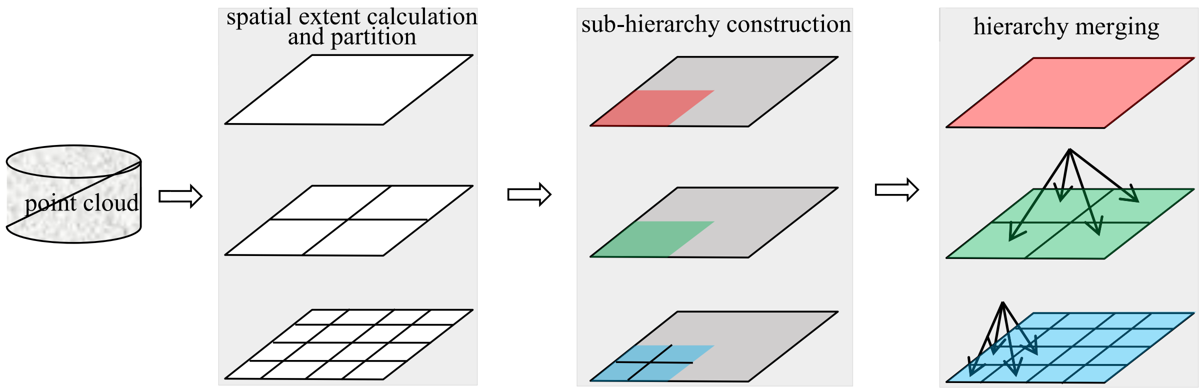

point cloud
spatial extent calculation and partition
sub-hierarchy construction
hierarchy merging

Figure 3. 2D illustration of parallel point cloud hierarchy construction. See text for details.

B. *Realtime rendering*

Aiming at efficient real-time rendering of large point cloud datasets, dynamic point cloud scheduling based on the generated octree structure is utilized (an illustration is shown in Fig. 4). In particular, nodes that are outside the visible region are excluded from the rendering list with the so-called view-frustum culling technique and visible nodes with different LOD according to their distances to the viewpoint are rendered jointly. The LOD in a region is equal to the level of the highest-level node therein. LOD constraints ensure that nodes closer to the virtual camera are favoured over nodes that are further away.

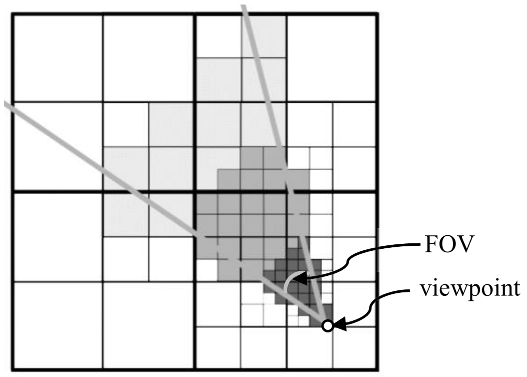


Figure 4. View-frustum culling and node selection.

In practice, the nodes that should be rendered are determined by an hierarchy traverse operation, which is done in a screen-projected-size order. The largest node on screen is visited first, then the second largest, and so on. The projected size is obtained as a function of the field of view (FOV) V, the distance d from the viewpoint to the center of the node, the node's bounding sphere radius R_{bb}, and the height of the screen H_{sc} in pixels. The projected size S_n of the node is inversely proportional to the distance. Formally, it can be calculated as follows:

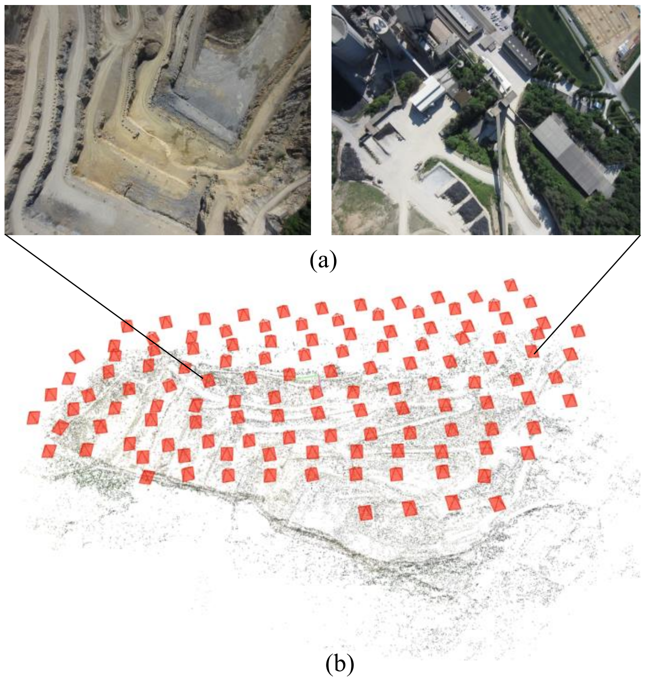
$$S_n = \frac{H_{sc} R_{bb}}{2\, d \times \tan\left(\frac{V}{2}\right)}.$$

A limit on the maximum number of points loaded and rendered for each frame can be set to accommodate for hardware of different performance.

IV. Experiments

In order to evaluate the performance of the approach presented in this paper, we implemented the hierarchy construction method in C and point cloud rendering system based on WebGL, and carried out both qualitative and quantitative experiments. The operation system for our test laptop is Ubuntu 16.04, and it is equipped with an Intel(R) Core(TM) i7-7700HQ CPU at 2.80 GHz (with 4 cores and 8 threads), a NVIDIA GeForce GTX 1060 GPU, and 16GB physical memory.

The point cloud datasets we used is obtained using photogrammetry method. In particular, we have downloaded a publicly available quarry image dataset (https://s3.amazonaws.com/mics.pix4d.com/example_datasets/example_quarry_2.0.zip) which consists of 347 high resolution aerial images which served as the input of photogrammetry point cloud generation. The popular bundler [8] and PMVS [9] algorithms were used to perform structure from motion (SfM) and multi-view stereo tasks, respectively. The raw output of PMVS is a set of dense point cloud, which contains outliers and holes. To produce more visually pleasing point clouds, the raw point clouds are converted into a textured mesh model using surface reconstruction method, and then sampled at different density scales to create a set of different size point cloud datasets. Fig. 5 shows two sample images of the quarry image dataset and the result of SfM including the sparse scene structure and 127 out of the 347 camera locations. Table 1 lists the detailed information of generated point cloud datasets for our experiments.



(b)

Figure 5. (a) Two sample images from the quarry image dataset used for point cloud reconstruction and (b) the locations of 127 images calculated using SfM method.

Table 1. Datasets used in our experiments.

Point Cloud Dataset	Original File Size (MB)	Number of Points
Quarry-10K	0.16	11,088
Quarry-100K	1.60	109,672
Quarry-1M	16.50	1,100,095
Quarry-10M	165.00	10,999,760
Quarry-100M	1650.00	100,219,810

In the following, we compare the efficiency of the method described in this paper and that of the one used by Potree. The experimental results on the five point cloud datasets are shown in Fig. 6, from which we can see that our method runs consistently faster than the comparative method, with an overall speedup ranging from 5 to 7 times. In particular, for dataset

Quarry-100M, the preprocessing time for Potree is about 200 seconds while ours takes only 40 seconds, which means our method processes about 2,505,495 points per second. It is noteworthy that the above performance is obtained on a 4-core CPU and the efficiency of our method can be improved even further for a more powerful computer with more cores and threads.

As for online rendering, all the tested datasets achieve a frame rate higher than 50fps. Some visualization results are shown in Fig. 7, where the figure on the second row is a global view of the point cloud dataset Quarry-100M and the figures on the first row are views for the three local regions A, B, and C highlighted by colored polygons.

The above experiments demonstrate that our method is effective and efficient for reorganization and interactive visualization of massive point clouds.

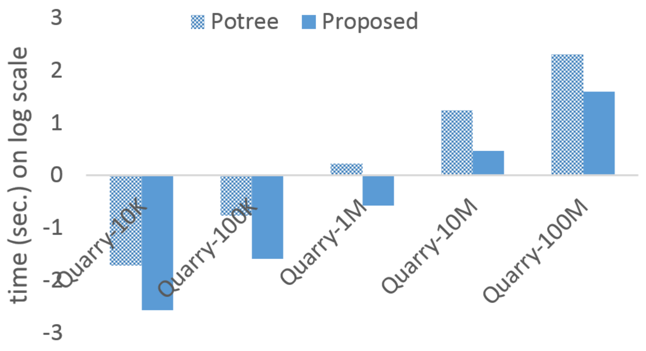


Figure 6. Comparison of processing time for point cloud reorganization.

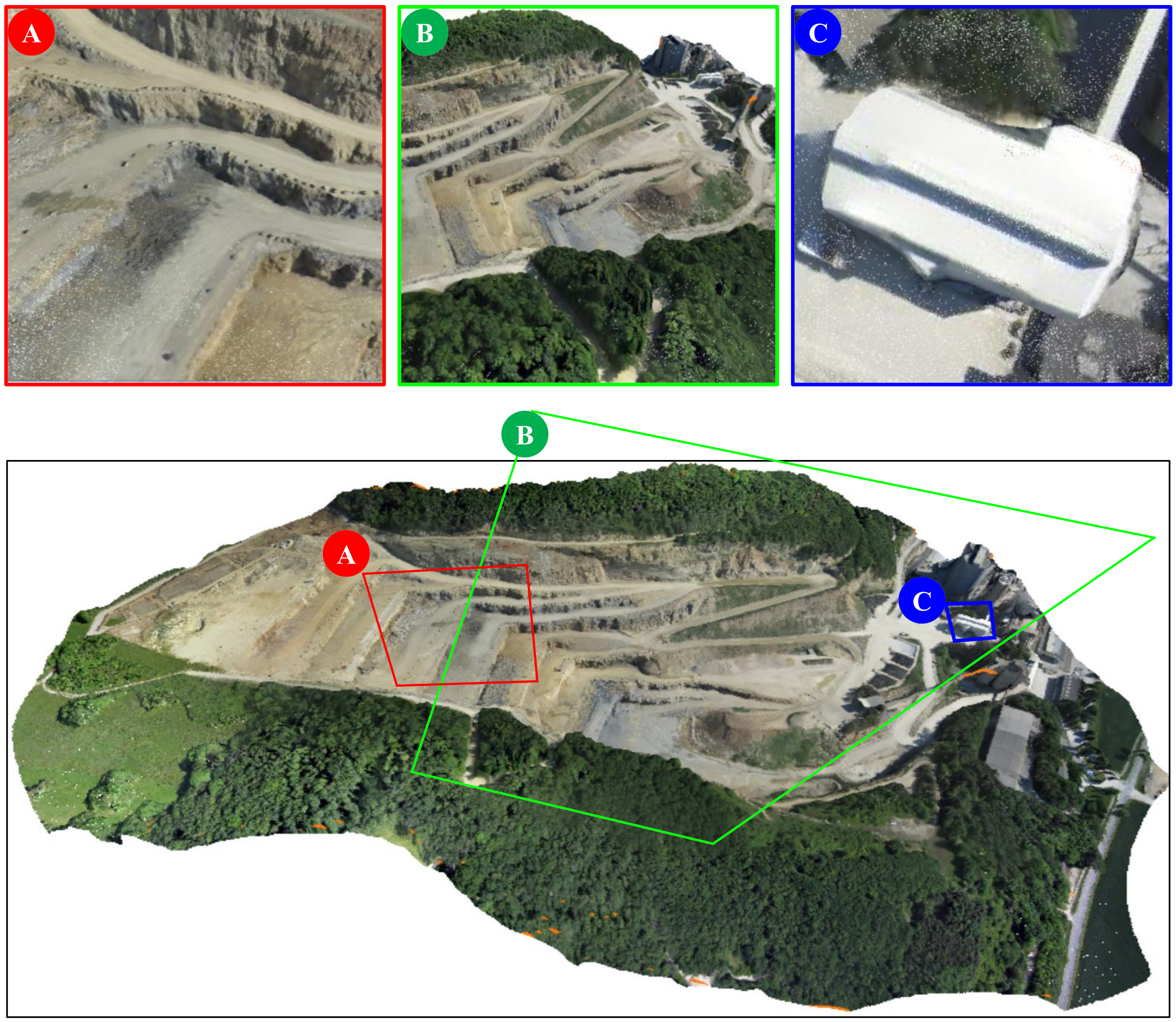


Figure 7. Visualization results of point cloud dataset Quarry-100M on different level of details. See text for details.

V. Conclusion

This paper presents a method to efficiently construct a point cloud hierarchy for large scale point cloud visualization. The processing of data hierarchy construction is accelerated by a fast cube-tile random point cloud subsampling algorithm and a divide-and-conquer scheme which enables parallel data processing. Qualitative and quantitative experimental results on several point clouds demonstrate the effectiveness and efficiency of the proposed approach. As for future work, we plan to extend our method to city or globe scale dataset and investigate its performance.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61873274.

References

[1] Rusinkiewicz, S. & Levoy, M. Qsplat: A multiresolution point rendering system for large meshes. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'00, 343–352.

[2] Gobbetti, E. & Marton, F. Layered point clouds: A simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. Comput. Graph. 28, 815–826 (2004).

[3] Wimmer, M. & Scheiblauer, C. Instant points: Fast rendering of unprocessed point clouds. In Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics, SPBG'06, 129–137.

[4] Goswami, P., Zhang, Y., Pajarola, R. & Gobbetti, E. High quality interactive rendering of massive point models using multi-way kd-trees. In Proceedings Pacific Graphics Poster Papers (2010).

[5] Uitentuis, M. (eds.) Management of massive point cloud data: wet and dry, Green Series, 9–15 (Nederlandse Commissie voor Geodesie, 2010).ows. Tech. Rep., GeoNext BV (2015).

[6] Potree. http://potree.org/ (accessed on April 10th, 2019)

[7] Claus Scheiblauer. Interactions with Gigantic Point Clouds. PhD thesis. Vienna University of Technology, 2014.

[8] Noah Snavely, Steven M. Seitz, Richard Szeliski. Modeling the World from Internet Photo Collections. International Journal of Computer Vision, 2008, 80(2): 189-210.

[9] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 32(8): 1362-1376.