An Improved Adaptive Dynamic LOD Algorithm Based on Large-Scale Point Clouds

Zhaolong Li[1],Chenzhe Wang[2],Xuewei Chen[3], Shiliang Tao[4]

[1]School of Geomatics and Urban Spatial Informatics, Beijing University of Civil Engineering and Architecture, China - lzlfofficial@163.com
[2] National Geomatics Center of China, Beijing, China - wangchenzhe@ngcc.cn
[3] Tencent, Beijing, China - cxw0911@163.com
[4] Tencent, Beijing, China - 505165517@qq.com

Keywords: Point Cloud, Level of Detail, Dynamic Scene, Adaptive Adjustment, Spatial Distribution Characteristics

Abstract

The rapid development of 3D scanning technology has significantly reduced the cost of acquiring high-precision point cloud data, leading to exponential growth in applications such as digital city modeling, autonomous driving, and virtual reality. However, managing point cloud datasets containing hundreds of millions of points poses severe challenges for storage, transmission, and real-time rendering. Traditional Level of Detail (LOD) techniques struggle to balance efficiency and accuracy, especially under dynamic viewpoints and complex scenes. This paper introduces an enhanced dynamic adaptive LOD algorithm designed to boost the interactivity of large-scale point clouds and eliminate the need for re-computation when data is modified. The research objective is to address the bottleneck in real-time processing of large-scale point cloud data through improved data structures and sampling strategies.Our method builds a multi-resolution data structure that combines octree indexing with spatial sampling to achieve efficient spatial queries. The technical core is a point-voxel hybrid octree based on secondary sampling, which significantly improves visualization efficiency. The innovation lies in the adaptive sampling technique, which dynamically adjusts grid size according to point cloud density, enhancing sampling efficiency and detail precision in both sparse and dense regions.The experimental implementation uses WebGL, Vue, Three.js, Node.js, and MySQL technologies. Test results reveal significant improvements in rendering speed and resource utilization. This research not only enhances real-time rendering capabilities for large-scale point clouds but also has important application value in fields such as GIS and real-time SLAM.

1. Introduction

Driven by rapid progress in 3D scanning technology, high-precision point clouds have become a foundational dataset for numerous applications, even as their scale continues to grow. In order to obtain detailed three-dimensional point cloud data, the data volume has also increased accordingly. (Han et al., 2017) These massive point cloud datasets have been widely applied in multiple fields, including but not limited to digital cities, autonomous driving, and virtual reality. However, large-scale point cloud data usually contains hundreds of millions of points, and such large-scale data volume poses severe challenges to storage, transmission, and rendering. Level of Detail (LOD) technology is one of the key technologies for addressing large-scale point cloud rendering and visualization(Abualdenien and Borrmann, 2022). LOD technology employs models of different precision at different viewing distances, reducing data processing volume while ensuring visual effects, thereby decreasing the computational load of visualization technology. Traditional LOD technology mainly targets continuous surface models, and its application effect on discrete point cloud data is relatively limited.

Currently, LOD technology for point clouds has made certain progress, but still faces many challenges when processing dynamic scenes: most point cloud LOD methods are based on static LOD, which generates different levels of LOD during the preprocessing stage(Han et al., 2017). This approach results in the point cloud being unable to adapt to dynamically changing viewpoints and environments during visualization. Some dynamic LOD methods can adapt to viewpoint changes to a certain extent, but struggle to balance precision and efficiency, especially revealing lower efficiency when processing large-scale point cloud data. Building on this, some dynamic LOD methods generally do not consider the spatial distribution characteristics of point cloud data and lack mechanisms for adaptive adjustment based on the local complexity of the point cloud, resulting in lower resource utilization(Schütz, 2015).

The specific research process of this paper is as follows(refer to Figure 1.): Step 1 transforms raw point cloud data through quadtree and octree construction with sparse grid merging to build a global octree; Step 2 creates a point-voxel hybrid octree through voxelization and establishes dynamic LOD structures; and Step 3 implements efficient rendering through view frustum culling and visible node determination, enabling rapid visualization of the processed data.

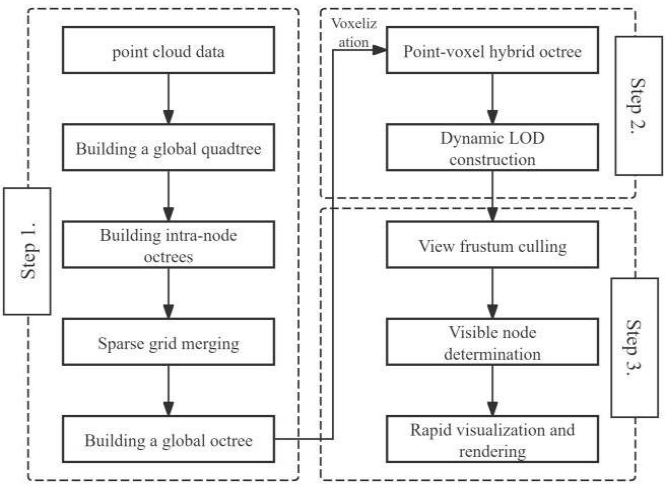


Figure 1.Workflow diagram.(Step 1.Octree construction. Step 2.Dynamic LOD construction. Step 3.High-quality rendering)

2. Construction of Large Scale Point Cloud

The data level of large-scale point cloud data can reach tens of billions or hundreds of billions. Directly reading and rendering these data as a whole on browsers places very high demands on hardware and software, making rendering extremely difficult. Scholars both domestically and internationally have researched many methods for organizing point clouds. To address the characteristics of different data, hybrid tree structures have been most widely applied, including: hybrid structures of octrees and KD trees, hybrid structures of extended quadtrees and three-dimensional R-trees, dual-layer quadtree structures, hybrid structures of quadtrees and KD trees, dual-layer octree structures, hybrid structures of KD trees and chain-linked octrees, hybrid structures using octrees nested with spatial grids, and so on(Schwalbe and Finzel, 2024).Our method efficiently organizes and schedules point cloud data by employing a dual-sampling approach to generate a voxel-point hybrid octree.

2.1 Point Cloud Octree Construction

Adopted in this paper is an octree-based spatial partitioning structure, which samples point cloud data from top to bottom during the construction process to generate a multi-resolution form. The multi-resolution organization of point clouds contains two parts: point cloud data and node information. The point cloud data consists of the geometric data of the node, while the node information mainly includes the total number of points, node bounding box, node name, node size, number of child nodes, and other information, providing necessary traversal parameters for subsequent point cloud visualization and scheduling(Huang, H., 2023).

For efficient data management, we structure the point cloud into a non-redundant octree. The primary goal of this structure is to ensure that every point from the original dataset has a unique location within the hierarchy. This is achieved by partitioning the space and assigning points exclusively to one node. Consequently, the hierarchy is not a series of overlapping, simplified views, but a collection of mutually exclusive data blocks. Higher-level nodes provide a coarse overview, while deeper nodes progressively add finer details. This organization method is critical for minimizing storage footprint and preventing data duplication during rendering and modification. The terminal nodes of this tree, or leaf nodes, represent the finest spatial partitioning of the dataset.

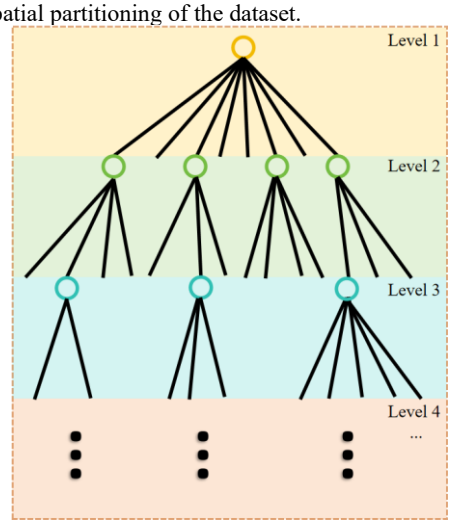
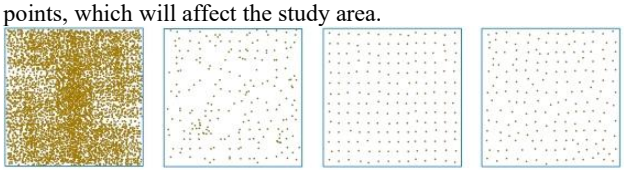


Figure 2.Octree diagram

2.2 Construction of Point Cloud Spatial Index

Different resolution levels between point clouds are mainly achieved through sampling. Commonly used sampling methods include random sampling, grid-based sampling, and Poisson disk sampling. As shown in Figure 3., random sampling exhibits extremely uneven distribution, with significant density variations in the spatial distribution of point clouds. Poisson disk sampling is widely used in two-dimensional image processing for point rendering, and its sampling results ensure that the distance between points does not exceed a specified minimum distance, i.e., the radius of the given Poisson disk, resulting in a uniform spatial distribution. The grid method divides space into square grids, and when points exist within a cell, they are moved to the center of the cell. The distance between all points can be maintained at the grid size, which is a conventional method for generating uniform distances between points. Initially, to reduce the influence of noise points and ensure efficiency, grid-based sampling is used for the first sampling. After generating regular grids, the number of points in each grid is detected, and grids with point counts below the set threshold are removed, thereby eliminating some noise points, which will affect the study area.



(a) (b) (c) (d)

Figure 3. Different sampling methods.(a) Original Data (b)Random Sampling (c) Grid Sampling(d) Poisson Disk Sampling

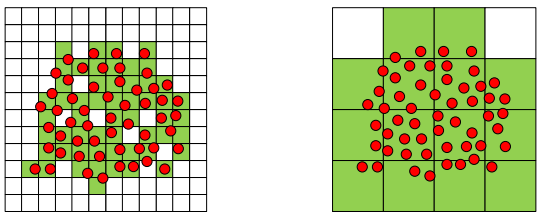
The second processing, Poisson disk sampling is selected as it produces the best sampling results, because it can simultaneously satisfy both uniformity and randomness. This means that the generated points maintain a minimum distance to avoid excessive clustering, while not exhibiting the artificial patterns seen in regular grids. This blue noise characteristic allows the sampled points to better preserve the original geometric features while reducing data volume. This method appears more natural visually, offers high sampling efficiency, and can adapt to different density requirements by adjusting the minimum distance parameter. These qualities make it perform excellently in applications such as point cloud reconstruction, 3D scanning, and geometric processing, achieving an ideal balance between fidelity and computational efficiency.

2.3 Grid Merging Algorithm Construction

Shown in the previous section, after organizing the point cloud into an initial global octree structure, uniform grids are generated based on the octree leaf node ranges through the bounding box generated from the global point cloud, with resolution determined by the node level(Lv et al., 2024). While ensuring the integrity of point cloud geometric features, data storage redundancy is reduced by merging grid blocks in low-density or low-feature areas, thereby improving retrieval and rendering efficiency for large-scale point clouds.

Node undergoes grid division, and for each cell, it's convenient to record both the points falling within it and its surrounding cells. To accelerate the distance-checking process required by Poisson disk sampling, we implement a spatial grid acceleration structure. Instead of comparing a new point candidate against all existing points in the node, the search

space is localized. The algorithm only needs to compute distances to points within the candidate's own grid cell and its immediate 26 neighboring cells. This spatial hashing technique dramatically reduces the number of distance calculations, transforming the problem from a costly global search to an efficient local one. To enable Poisson disk sampling between adjacent nodes, the neighborhood cells of each cell include both cells within the node and cells outside the node. When filling point clouds into tree nodes, distance judgments must be made not only with neighborhood cells within the node but also with cells in adjacent nodes. However, whether inside or outside the node, calculations still involve points from at most 27 cells. Creating spatial grids based on nodes not only improves algorithm efficiency to some extent but also effectively avoids the problem of dense sampling results occurring at the boundaries of adjacent nodes.



(a) (b)

Figure 4.Grids of different sizes:(a)Smaller size grid.(b)Larger size grid.

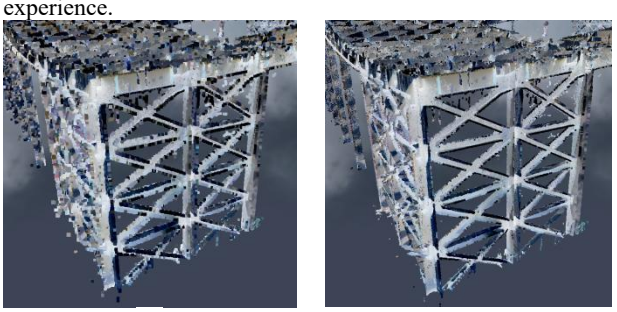
Size of the cell is between the spacing of the current level and the node size.If the size is too small, it will occupy more memory space and reduce performance; if the size is too large, the cost of distance judgment between points cannot be effectively addressed. Figure 4. illustrates the effects of different grid sizes, showing that whenever a point is added, the distance from that point to all points in its cell and adjacent cells will be calculated. Points that pass the distance check will be added to the green cell. When a cell receives its first point, a cell instance will be created, and it will determine whether neighborhood cells already exist. If they do, this cell will be added to the neighborhood list of those neighborhood cells. This process merges into a complete octree, preparing for subsequent LOD (Level of Detail) generation operations.

3. Construction of Point Cloud Dynamic LOD

Detailing the method for generating the point cloud octree structure, we further consider how to utilize this structure to implement dynamic Level of Detail (LOD) display for point clouds. As an efficient spatial partitioning structure, the octree not only provides an organizational framework for point cloud data but also lays the foundation for dynamic LOD technology. By reasonably utilizing the hierarchical characteristics of the octree, we can dynamically adjust the display density and refinement level of the point cloud based on viewing angle, distance, and computational resource limitations, thereby optimizing rendering performance while maintaining visual quality.

3.1 Generation of Point Voxel Octree

Point clouds requires a level of detail structure. When processing large-scale point cloud data, completely loading and rendering all points would lead to excessive consumption of computational resources, especially for real-time application scenarios. Important characteristics of LOD structures include reducing loading time and memory usage, improving rendering performance, and ensuring low computational complexity for each loading operation. This is particularly important for point cloud applications in network environments, where users can first see a low-resolution point cloud model, followed by gradually loading more refined details in Figure 5. This approach not only reduces computational complexity but also optimizes bandwidth usage while improving the user experience.



(a) (b)

Figure 5. (a)Low level LOD.(b)High level LOD.

Computation time for points, this is achieved by voxelizing some of the points. Color-filtered voxels are generated at lower LOD levels to enhance visual quality, creating a point-voxel hybrid octree. The necessity of this hybrid structure is reflected in multiple aspects: First, it effectively balances the contradiction between storage efficiency and data accuracy, significantly saving storage compared to the original point cloud while retaining more detail than pure voxel representation. Second, the hybrid structure supports multi-resolution data access, allowing the rendering system to intelligently select appropriate representation methods based on viewing distance—using voxel representation at a distance to improve efficiency and point representation up close to ensure accuracy.

3.2 Voxelization and Voxel Based Level of Detail

Under the point-voxel hybrid octree structure, the basic steps to convert part of the point cloud data into voxels are as follows(refer to Figure 6.):

1)Mapping from point cloud to voxels: For each voxel (octree node), find all the points that fall within this voxel. This can be achieved by comparing the coordinates of the points with the boundaries of the voxel. Then, calculate the attributes of the voxel based on the attributes of these points. (based on Figure 6 (a) and (b))

2)Voxel refinement: During the LOD (Level of Detail) generation process, voxels can be dynamically refined or merged as needed. For example, if a voxel is close to the observer, it may need to be further subdivided into smaller voxels. Conversely, if a voxel is far from the observer, it may be possible to merge it with adjacent voxels to reduce rendering complexity.(refer to Figure 6(c))

3)Voxel rendering: In the rendering stage, various voxel rendering techniques can be used to display voxel data. This can include methods such as direct voxel rendering and voxel-to-polygon conversion. (refer to Figure 6(d))

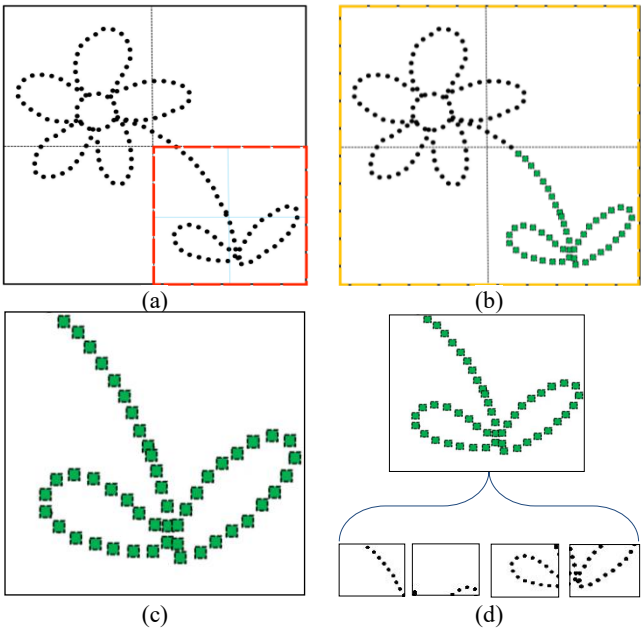


Figure 6.(a)Regions to Be Voxelized After Evaluation.(b)Nodes Contain Point Clouds and Voxels.(c)Voxel Details. (d)Original Point Clouds Below the Voxels

3.3 Efficient Scheduling of Large Scale Point Cloud

The multi-resolution construction addresses the issue of spatial data organization when rapidly visualizing large-scale point clouds on the Web. In addition, it is necessary to combine efficient scheduling algorithms to manage the node data within the multi-resolution structure. This chapter will propose and implement efficient scheduling algorithms for multi-resolution point clouds from two aspects: node visibility determination and multi-resolution structure traversal(Guo and Wang, 2024). By minimizing the loading and rendering of invalid nodes while maintaining the integrity of point clouds within the visible range, the goal of visualizing massive point clouds on the Web can be achieved.

This paper achieves rapid visualization of large-scale point clouds through Three.js. Three.js uses WebGL as the underlying rendering technology and specifically handles point cloud data via the THREE.Points class. The core process of point cloud visualization is converting points in three-dimensional space into pixels on the screen. Specifically, it first creates a geometry to store the positional data of the points, then applies materials to define the appearance of the points, such as size and color. During rendering, Three.js automatically applies frustum culling, rendering only the points within the camera's view, which greatly enhances performance. For large-scale point clouds, it supports memory optimization using BufferGeometry and can dynamically adjust the density of displayed points based on distance through Level of Detail (LOD) techniques. Three.js also provides various shaders and post-processing effects, enabling dynamic changes in point size, color, and transparency, as well as advanced visual effects like Screen Space Ambient Occlusion (SSAO).

In traditional Three.js development, Three.js already supports frustum culling during object rendering. However, this culling method is not suitable for the dynamic Level of Detail (LOD) structure proposed in this paper. The scheduling and rendering of multi-resolution point clouds and models involve both scheduling and rendering processes. The frustum culling provided by Three.js only satisfies the rendering stage, while the nodes that need to be rendered are determined during the scheduling process. Therefore, if frustum culling could be applied during the scheduling process to eliminate most nodes, it would reduce unnecessary data requests to the server and improve the system's rendering efficiency(Röttger et al., 1998).

The LOD structure of point clouds is essentially a multi-resolution octree, where each node in the tree contains both geometric data and node description information. The node description information includes the indexing relationships between the current node and its child nodes, as well as the bounding box information of the node. During the initial scheduling, the description information of the root node is first loaded, and the relationship between the root node and its child nodes is established within the scheduling system. At this stage, no geometric data exists in the scheduling system.

Based on the bounding box information of the nodes, an intersection test between the view frustum and the bounding boxes is performed using Three.js, resulting in three possible spatial relationships:

1. The bounding box of the node is not within the view frustum: In this case, the geometric data within this node must also be outside the view frustum, and therefore, it does not need to be loaded or rendered.

2. The bounding box of the node is entirely within the view frustum: Consequently, the geometric data within this node is also entirely within the view frustum. The node is deemed visible and is fully loaded for rendering.

3. The bounding box of the node intersects with the view frustum: Under this condition, a scenario need to be evaluated: Intersection with a voxel or point: It is possible that only a portion of the node lies within the view frustum. By further performing intersection tests between the bounding boxes of the child nodes and the view frustum, additional nodes can be culled. This process is recursively executed until all nodes are determined to be entirely within the view frustum.

Through this approach, the scheduling process effectively leverages frustum culling to minimize unnecessary data requests to the server and enhance the rendering efficiency of the system, thereby facilitating the visualization of massive point clouds on the Web.

4. Experiment and Result Analysis

The point cloud data in this paper originates from self-collected raw point cloud data. Each point consists of three-dimensional coordinate information (longitude, latitude, and elevation). Experimental Data Area a is sourced from the first floor of Building F at Beijing University of Civil Engineering and Architecture, containing 59,367,498 points with a file size of 2.31 GB. Experimental Area b is sourced from Building 2 of a high-efficiency teaching building in Beijing, containing 86,336,488 points with a file size of 2.59 GB. The experimental environments are shown in Table 1.

Category	Configuration	Details
Hardware	CPU	Intel(R)Core(TM)i7-6700 CPU @ 3.40GHz
	RAM	16 GB

Software	ROM	3.64 TB HDD
	GPU	NVIDIA GeForce RTX 3050 (8 GB)
	Operating System	Windows 10 64-bit Professional Edition
	Development Environment	Visual Studio Code
	Graphics Interface	Vue 3 + Three.js

Table 1.Experimental Environment Configuration

The two datasets were partitioned and organized using different methods, and the comparison results are revealed in Figures 7., 8., 9., and 10.:

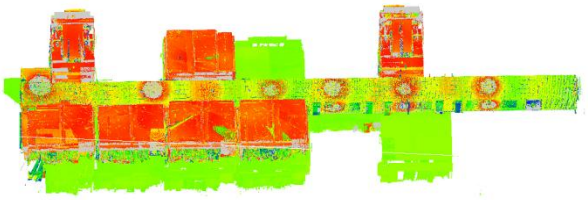


Figure 7. Octree Partitioning Using the Potree Method

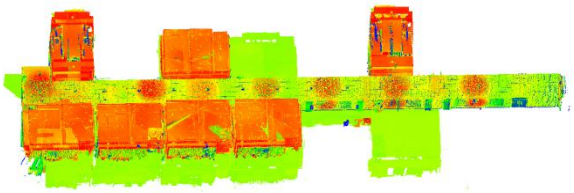


Figure 8. Octree Partitioning Using the Method Proposed in Our Method

Refer to Figures 7. and 8., the octree structure generated by the Potree method is more complex and provides better detail representation, but this reduces loading efficiency. Our method generates a simpler octree structure because some point clouds are pre-loaded as voxels. While this may sacrifice some details in large-scale scenes, when the scale increases, the point clouds within voxels load normally. This approach ensures efficiency while preserving all details.

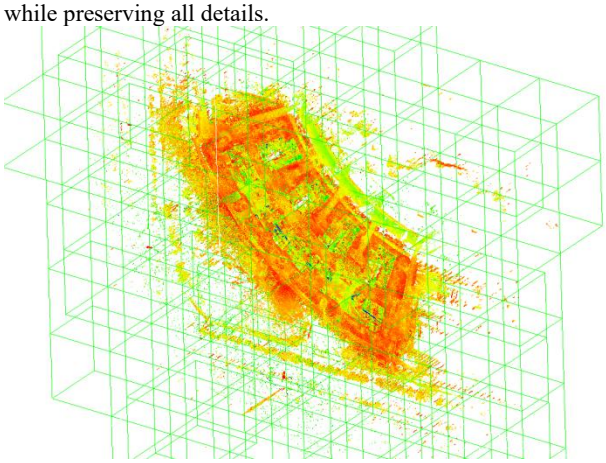


Figure 9. Low-Level LOD of the Initial Octree Partitioning

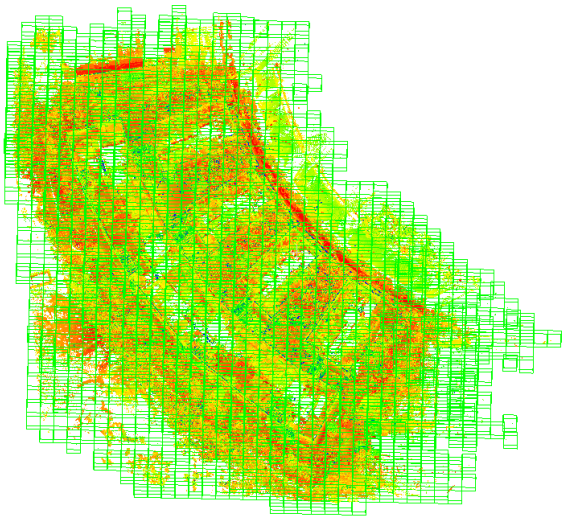


Figure 10. High-Level LOD of the Octree after Secondary Organization

According to Figures 9. and 10., the leaf nodes in low-level LOD have large gaps between them, with noise points causing interference at the edges, resulting in somewhat rough details. After processing with our method, the generated point cloud LOD at higher levels—which display more refined point cloud detail—removes noise point interference and supports dynamic LOD, creating a modifiable point-voxel hybrid octree.

Area	Points	Method	Time	LOD
Area a	59,367,498	Potree	30.18s	Static
Area b	86,336,488	Potree	42.33s	Static
Area a	59,367,498	Our	21.56s	Dynamic
Area b	86,336,488	Our	30.67s	Dynamic

Table 2.Comparison of the construction speed between our method and Potree

Based on the Table 2., our method demonstrates significant advantages in point cloud data processing across two areas. Specifically, for Area a, the Potree method requires 30.18 seconds, while our dynamic LOD method completed the task in just 21.56 seconds, achieving a performance gain of approximately 29% over the Potree method. Similarly, for Area b, the Potree method takes 42.33 seconds, whereas our approach requires only 30.67 seconds, representing an improvement of approximately 28%. Additionally, the Potree employs Static LOD , while our method utilizes Dynamic LOD, which not only improves processing speed but may also provide more flexible and efficient point cloud rendering effects in various application scenarios. Overall, the data indicates that our method can significantly reduce the time required and optimize performance when processing large-scale point cloud data.

5. Conclusion

This paper presents a method utilizing secondary point cloud sampling combined with partial voxelization to create a point-voxel hybrid octree. Our approach enables rapid point cloud visualization on WebGL, allowing for display, editing, and other operations across larger areas, with data scheduling efficiency improved by 24.5% and faster visualization rendering speeds (refer to Table 2.). The generation of voxels effectively eliminates the adverse effects of noise points on data quality, performing excellently when processing medium-scale datasets. The dynamic point cloud LOD generation

ensures that the original data's octree structure remains unaffected, eliminating the need for rebuilding after modifying portions of the point cloud. This study was conducted on well-registered and integrated point cloud datasets. Despite its promising results, our method currently does not handle dynamic scenes where the point cloud itself changes over time (e.g., moving objects). The voxelization process is also based on simple color averaging and could be improved with more advanced attribute filtering techniques. Future work will focus on extending our framework to support dynamic point clouds by incorporating incremental update mechanisms into the octree. We also plan to explore more sophisticated voxel attribute computation methods and investigate the application of our algorithm in collaborative VR/AR environments.

References

Abualdenien, J., Borrmann, A., 2022: Levels of detail, development, definition, and information need: a critical literature review. *Journal of Information Technology in Construction* 27.

Bagul, S., Laefer, D., 2022: Three-Dimensional Enablement of Place-Based, Pandemic Behaviors. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 10, 21-28.

Fang, Y., Li, Y., Fan, L., 2024: A Case Study on the 3D Interactive Virtual Geological Scene of the Yangshan Monument for Geology Education. *2024 International Conference on Virtual Reality Technology* 68-73.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2020: Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43(12), 4338-4364.

Han, S., 2018: Towards efficient implementation of an octree for a large 3D point cloud. *Sensors* 18(12), 4398.

Han, X. F., Jin, J. S., Wang, M. J., Jiang, W., Gao, L., Xiao, L., 2017: A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication* 57, 103-112.

Huang, H., 2023: Construction of Multi-resolution Spatial Data Organization for Ultralarge-scale 3D Laser Point Cloud. *Sensors and Materials* 35(1), 87-102.

Lei, H., Akhtar, N., Mian, A., 2019: Octree guided cnn with spherical kernels for 3d point clouds. *IEEE/CVF conference on computer vision and pattern recognition* 9631-9640.

Lv, J., Su, H., Liu, Q., Yuan, H., 2024: No-reference bitstream-based perceptual quality assessment of octree-lifting encoded 3D point clouds. *IEEE Transactions on Visualization and Computer Graphics.*

Röttger, S., Heidrich, W., Slusallek, P., Seidel, H. P., 1998: Real-time generation of continuous levels of detail for height fields *WSCG'98* 315-322.

Rusu, R. B., Cousins, S., 2011: 3d is here: Point cloud library (pcl). *2011 IEEE International Conference on Robotics and Automation* 1-4.

Schütz, M., 2015: Potree: Rendering large point clouds in web browsers. *Doctoral dissertation, Technische Universität Wien.*

Schwalbe, G., Finzel, B., 2024: A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery* 38(5), 3043-3101.

Umemiya, S., Hasegawa, R., Yasumuro, Y., Kubota, S., 2024: *Maintenance Information System of Utility Tunnel Using 3D Point Cloud Data*. Springer Nature, Cham.

Vo, A. V., Truong, H. L., Laefer, D. F., Bertolotto, M., 2015: Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 104, 88-100.

Yeshwanth, K. A., Noufia, M. A., Shahira, K. A., Ramiya, A. M., 2019: Building information modelling of a multi storey building using terrestrial laser scanner and visualisation using potree: An open source point cloud renderer. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42, 421-426.

Zhang, W. Y., Tan, G. X., 2022: Research on semantic 3D building modeling with multiple levels of detail. *Journal of Graphics* 43(1), 163.