

Article

# 3D Tiles-Based High-Efficiency Visualization Method for Complex BIM Models on the Web

Wenxiao Zhan, Yuxuan Chen and Jing Chen \* 

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China; zhanwenxiao@whu.edu.cn (W.Z.); chen\_yuxuan@whu.edu.cn (Y.C.)

\* Correspondence: jchen@whu.edu.cn; Tel.: +86-13871066034

**Abstract:** Geographic data visualization is an important research area of Web Geographic Information System (GIS). Owing to the detailed subassemblies and exhaustive knowledge database, building information modeling (BIM) plays an important role in geospatial research and industries. The integration of BIM and GIS contributes to the smooth visualization, quick construction, and efficient management of geographic data. However, there are very few methods that can yield high-efficiency data transmission and visualization for complex BIM models while maintaining the integrity of the internal subassembly structure and attributes. To overcome this issue, this paper proposes a 3D Tiles-based visualization method for complex BIM models on the Web-based 3D model viewer. This method is adopted to partition the BIM model according to its assembly without simplifying the BIM model, by using a tiling method for 3D models based on a degraded R-tree, which accounts for the size of tiles. Subsequently, we introduce the “Mask Filter,” a level of detail method that is used to layer the BIM model. Conducting a series of contrast experiments, the result indicates that this method is efficient and feasible, which significantly improves visualization performance of complex BIM with mass data in the geospatial scene and facilitates the integration of Building Information Modeling and Geographic Information System.



**Citation:** Zhan, W.; Chen, Y.; Chen, J. 3D Tiles-Based High-Efficiency Visualization Method for Complex BIM Models on the Web. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 476. <https://doi.org/10.3390/ijgi10070476>

Academic Editor: Wolfgang Kainz

Received: 24 May 2021

Accepted: 8 July 2021

Published: 11 July 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

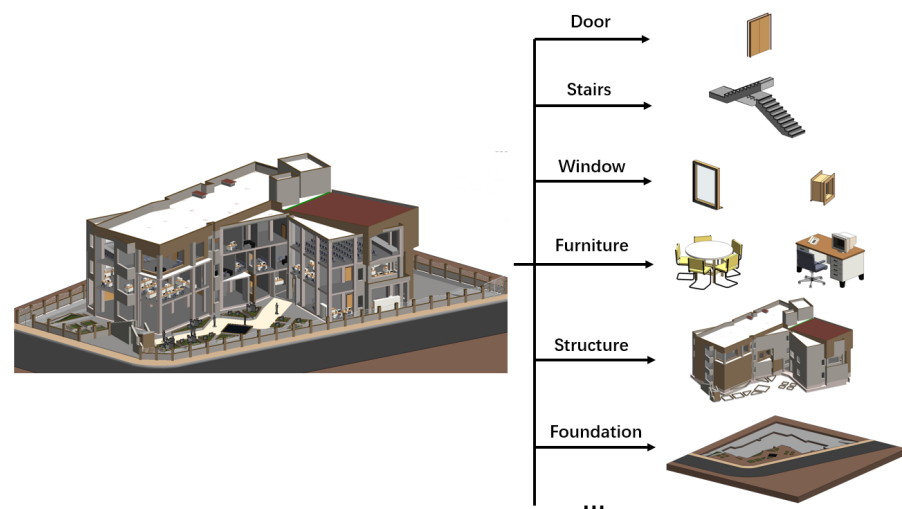


**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** building information modeling; level of detail; data organization; 3D Tiles

## 1. Introduction

Owing to the rapid developments in 3D modeling, more elaborate 3D models are being employed in a variety of applications and industries, including urban planning and analysis [1], virtual earth modeling [2], and 3D geographic information systems (GIS) [3]. In particular, in the architecture, engineering, and construction (AEC) industry [4], building information modeling (BIM) [5] is used to digitize different properties and functions related to the construction process. BIM models assist in building different objects by serving as a reliable basis for engineers, architects, officials and so on to make decisions. Based on the design drawings, the subassemblies of a BIM model are established individually through an intricate process; this results in a model that retains many internal facilities and features. BIM models can include multiple building modules, and each module is composed of multiple subassemblies (Figure 1). Industry Foundation Classes (IFC) is the current international standard BIM format. Generally, the BIM is in IFC format. As a data exchange standard, IFC aims to integrate all information in the entire building life cycle into one BIM, so that all software in the life cycle can share and exchange information. It is composed of four layers, from the top to bottom, namely, domain/application layer, interoperability layer, core layer and resource layer. The geometries and attributes of subassemblies are stored in resource layer. Due to elaborate subassemblies and abundant building information, this type of 3D model can produce an excellent visual effect. However, the model's complex structure and the significant amount of data associated with it pose challenges in terms of efficiently transmitting and visualizing BIM models on the Web-based 3D model viewer.

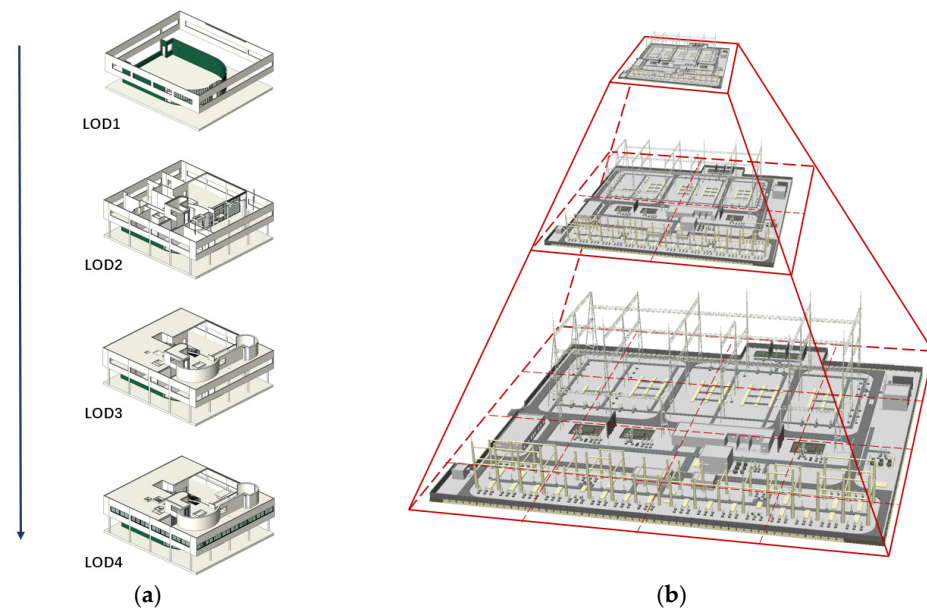


**Figure 1.** Subassemblies of a 3D BIM model.

Due to these challenges, many GIS companies have proposed data formats that can accommodate the streaming of BIM models. Among these companies, Super Map, Cesium, and ArcGIS have each launched their own open 3D model format specifications, i.e., Spatial 3D Model (S3M), 3D Tiles, and Indexed 3D Scene Layer (I3S), respectively. However, there exist significant differences among the data structures in these formats. 3D Tiles [6] supports any form of spatial data structure, making it the most flexible option. It employs a tree structure to manage the 3D model. The index tree is stored in the json files. The spatial data and attribute data of the 3D model are stored in the tile files. I3S [7] supports regular partitions (quadtrees and octrees) and density dependent partitioning (R-tree), which is characterized by a combination of json files and binary files. The 3D model is tiled and stored in the binary files (.bin), which are organized into a tree structure by the index files. S3M [8] is a data structure that has been customized by R&D and features a powerful framework. In S3M, tree structure is also taken as the spatial data infrastructure. Spatial data is stored in the s3mb files and attribute data is stored in the s3md files. These three data formats all take into account the BIM model, so they can be highly compatible with the BIM model. For example, through a series of transition formats, such as Binary glTF and obj file, IFC can be converted to 3D Tiles. However, this conversion is just the change in the file format. Compared with S3M, 3D Tiles is a community standard identified by the Open Geospatial Consortium (OGC). It has higher community activity and is relatively more mature and developed. Compared with I3S, 3D Tiles exhibits greater flexibility, higher customizability, and better portability and openness.

The abovementioned data formats lay the foundation for the streaming of the 3D model. Fast streaming requires the minimum amount of data to be transmitted each time, which requires a more rational organization of 3D model. However, the abovementioned data formats fail to efficiently organize data. To improve the efficiency of scheduling (uploading the corresponding 3D model to the Web based on its index) and rendering BIM models on the Web-based 3D model viewer, it is necessary to ensure efficient data organization when using these data formats. Currently, the data organization methods for BIM models are mainly classified into two categories. The first category uses 3D models that can be organized into a fixed level of detail (LOD) hierarchy based on semantic information (Figure 2a). Here, level of detail is a visualization technique which can control whether the object should be rendered or not based on its position and importance for improving the rendering efficiency. [9] This method simplifies the process of organizing 3D models; however, it is only suitable for BIM models with simple structures, such as offices and residential buildings. If this method is used to process BIM models with complex structures, such as substations, the large number of complex subassemblies involved will considerably increase the quantity of data for every LOD, and the differences between subassemblies

at the same level will be neglected; this results in a lower quality of visualization. The second category employs quadtrees, octrees, or model simplification methods to organize the 3D models (Figure 2b). These methods account for the smoothness of horizontal shifts and vertical zooms on the resultant viewpoint. However, when using these methods, simplifications can distort the 3D model. For quadtrees and octrees, it can be difficult to balance the data volume of nodes and the depth of the index tree. Moreover, these methods may separate one subassembly into several parts, which violates the requirements of maintaining the integrity of the subassembly and results in poor visual effects during browsing. Very few methods can yield high-efficiency visualization for complex BIM with enormous quantities of data while maintaining the integrity of subassembly.



**Figure 2.** Data organization for BIM: (a) semantics-based data organization, and (b) quadtree data organization.

Based on the abovementioned gaps and limitations of the existing mainstream methods, we chose 3D Tiles as the format specification and Cesium as the visualization platform and 3D model viewer for use on the Web. In this study, based on 3D Tiles, a high-efficiency visualization method for BIM on the Web-based 3D model viewer was developed, without simplifying the BIM model. In particular, this study has two main contributions: (1) As for the problem of the inability to maintain the integrity of subassembly, we propose a tiling method for 3D models based on a degraded R-tree; and (2) as for the problem of low-quality visualization, a “Mask Filter” is introduced and a LOD method based on it is proposed. The scientific objective of the study is to realize high-efficiency visualization of BIM models with large data volume, which can help us smoothly interact with these BIM models in GIS, thereby vigorously promoting the integration of BIM and GIS.

## 2. Related Works

Because of the complex structures and massive data quantities of BIM models, real-time rendering on the Web-based 3D model viewer can be difficult. Therefore, it is necessary to choose an appropriate 3D model data format and conduct highly efficient data organization based on that format. 3D Tiles, which was proposed by Cesium, is an open specification for visualizing massive heterogeneous 3D geospatial content, and its hierarchical data structure can achieve fast streaming and precise rendering [6]. However, it does not define explicit rules for visualizing the content. Therefore, the methods for organizing and constructing 3D Tiles differ depending on the purposes and objects. The efficiency of the data organization method has a significant influence over the rendering

effect of 3D Tiles on the Web [10]. As a result, many researchers have studied different data organization methods to be used with 3D Tiles, according to the characteristics of the objects being modeled.

As for some 3D data, such as 3D city model [11], oblique photography models [12], point cloud LiDAR data [13], weather data [14], and flood data [15], it is common to use quadrees, octrees, or grid indices combined with the model simplifications to organize them. As the density of these data is relatively low and the structure of the model is relatively simple, this method can afford satisfactory visual effects.

BIM model can be divided into two categories, federated and integrated BIM model and independent BIM model. Both of them are composed of multiple modules. Each subassembly in each module is an individual entity. Based on these characteristics, the data organization method for 3D BIM models differs from that described above, and it can instead be classified into three categories. The first category is the semantic-based method. Xu et al. (2020) [16] separated residential buildings into four levels, namely the foundation, external contour, walls and windows, and interior [16]. The structure of the resulting 3D Tiles was similar to a linear linked list featuring a branch node with a single child. This method can achieve excellent visualizations when the structure of the BIM model is relatively simple; however, it fails to generate efficient visualizations for complex BIM models. The second category of data organization methods uses quadrees or octrees. Chen et al. (2018) [10] and Kim et al. (2015) [17] adopted octrees to organize 3D building models based on the spatial information of each subassembly. Although the size of individual tiles decreased, the depth of the index tree increased, and when the number of subassemblies in the BIM model increased, the browsing lagged. The third category of data organization methods includes model simplification. Hu et al. (2019) [18] used the quadric error metric algorithm to reduce the size of a BIM model; they subsequently employed the clustering-based normal vector regeneration algorithm to generate the LOD. Chen et al. (2016) [19] employed a texture-related vertex clustering algorithm [20] to simplify the complex BIM model, which reduced the data volume used for network transmission and progressive visualization. However, when using this method, the threshold of the simplification algorithm needs to be determined by the user. Additionally, local characteristics of the BIM model may be lost when the model is simplified, and this method treats the model as a singular entity, thereby neglecting the characteristics of each subassembly.

In summary, the existing methods entail the partitioning of 3D models based on semantic information or spatial locations and establish the LOD through semantic information or model simplification. These approaches afford efficient visualization for certain BIM models. However, if there are more elaborate subassemblies in the BIM model, the performance of these methods will deteriorate.

### 3. Methodology

#### 3.1. Scheduling Strategy for 3D Tiles in Cesium

Cesium is an interactive visualization platform, which allows users to personalize the visualization [21]. Scheduled by Cesium, 3D Tiles can only be rendered on the Web [22]. There are many scheduling parameters in Cesium that directly affect the rendering effect for 3D Tiles. The main parameters are detailed below.

The screen spatial error (SSE) is an indispensable parameter for rendering. According to previous research (see Equations (1) and (2)) [23], whether a tile in 3D Tiles is scheduled depends on two main factors: the bounding volume and the geometric error. The bounding volume defines the size and type of the bounding box for the tile. During browsing, Cesium will judge the topological relationship between the bounding box of a tile and the view frustum. If an intersection exists, it will judge whether the geometric error meets the rendering requirements. Furthermore, the geometric error defines the selection metric

indicating whether a particular tile will be scheduled. The larger the tile, the easier it is to schedule that tile.

$$SSE = \left( \frac{g * k}{d} \right) \quad (1)$$

$$k = \left( \frac{height}{2 * \tan(fovy/2)} \right) \quad (2)$$

where  $g$  is the geometric error,  $d$  is the distance from the viewpoint to the tile,  $height$  refers to the height of the screen, and  $fovy$  denotes the angle of the view frustum.

The scheduling strategy of 3D Tiles can be described as follows (Figure 3):

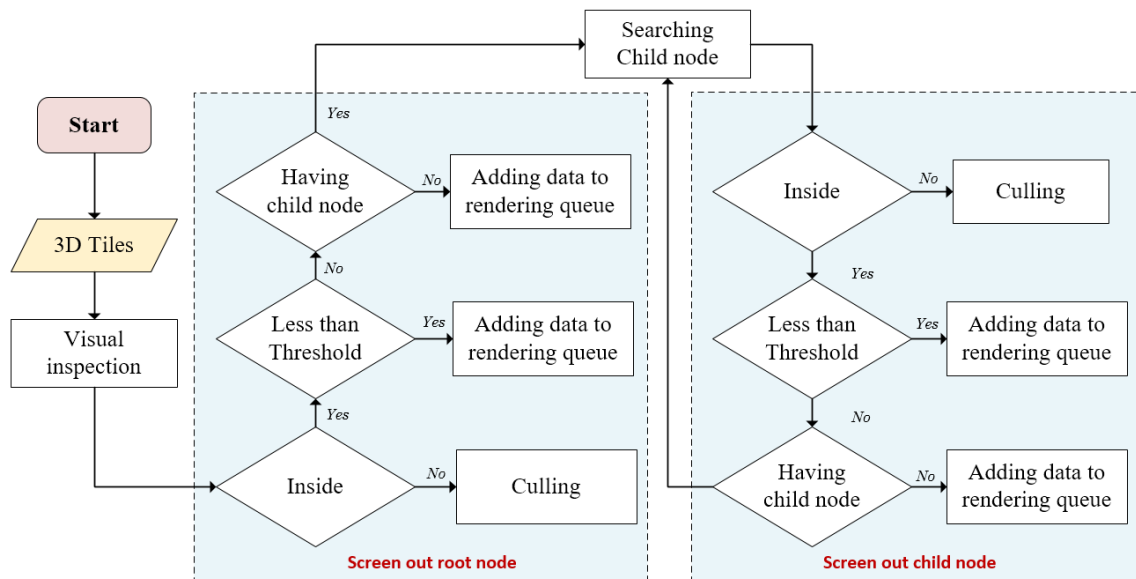


Figure 3. Scheduling strategy of 3D Tiles.

When browsing, Cesium will first perform a visual inspection for the 3D Tiles, based on which it will judge whether the bounding box of the root node and the view frustum intersect. If they do intersect, the corresponding root node will be added to the request queue. Subsequently, root nodes in the request queue will be judged sequentially to determine whether they meet the threshold requirement. In other words, this process determines whether the SSE of the root node is less than the maximum screen space error (MaxSSE). If it is less than this value, the corresponding root node will be added to the rendering queue; otherwise, the corresponding root node will be examined to determine whether it possesses a child node. If it does not, it will be added to the rendering queue directly; otherwise, its children are judged next. The processing of the children is the same as that of the root nodes. This process continues until all nodes which comply with all the rendering requirements are found, after which they are scheduled. It is worth noting that 3D Tiles will be scheduled for rendering only when both the bounding volume and the geometric error meet the rendering requirements.

### 3.2. Data Organization for BIM Models

By relying on model simplification methods and/or multi-resolution data, traditional data organization methods can build a partitioned, LOD 3D model. However, multi-resolution data for BIM models are often unavailable, and simplification algorithms are associated with certain drawbacks. Therefore, in order to achieve high-efficiency visualizations for BIM models on the Web-based 3D model viewer, instead of employing complex model simplification algorithms, this study considers each subassembly of the BIM model as the focal level of granularity. For retaining the original BIM model, a single-layer data or-

ganization method based on subassemblies is proposed, and the SSE is adopted to process the BIM model.

The overall process of data organization is shown in Figure 4. First, the algorithm performs “bottom-up” tiling for BIM, which considers the subassemblies of the BIM model as the level of granularity. As a result, the BIM model is partitioned into several bounding boxes. Each bounding box only contains a single subassembly, and the  $i$ -th bounding box represents  $Block_{k_i}^1$ . Thereafter, according to the threshold of the data volume for a single tile,  $Block_{k_i}^1$  is clustered continuously, which requires a recalculation of the bounding box of multiple  $Block_{k_i}^1$  and deleting the previous  $Block_{k_i}^1$  until the data volumes of all  $Block_{k_i}^j$  meet the threshold. Next, according to the geometric characteristics of  $Block_{k_i}^j$ ,  $GeoError_{k_i}$  and mask filters are set. Finally, Draco [24], which is an open-source library launched by Google, is adopted to compress  $Block_{k_i}^j$  and obtain  $Block_{k_i}^{j'}$ . Here,  $Block_{k_i}^j$  is the  $k_i$  Block after the  $j$ -th clustering,  $GeoError_{k_i}$  is the geometric error of the  $k_i$  block, and  $Block_{k_i}^{j'}$  is the compressed  $Block_{k_i}^j$ . As a result, the BIM model is organized into a set of partitioned 3D Tiles with a one-level index.

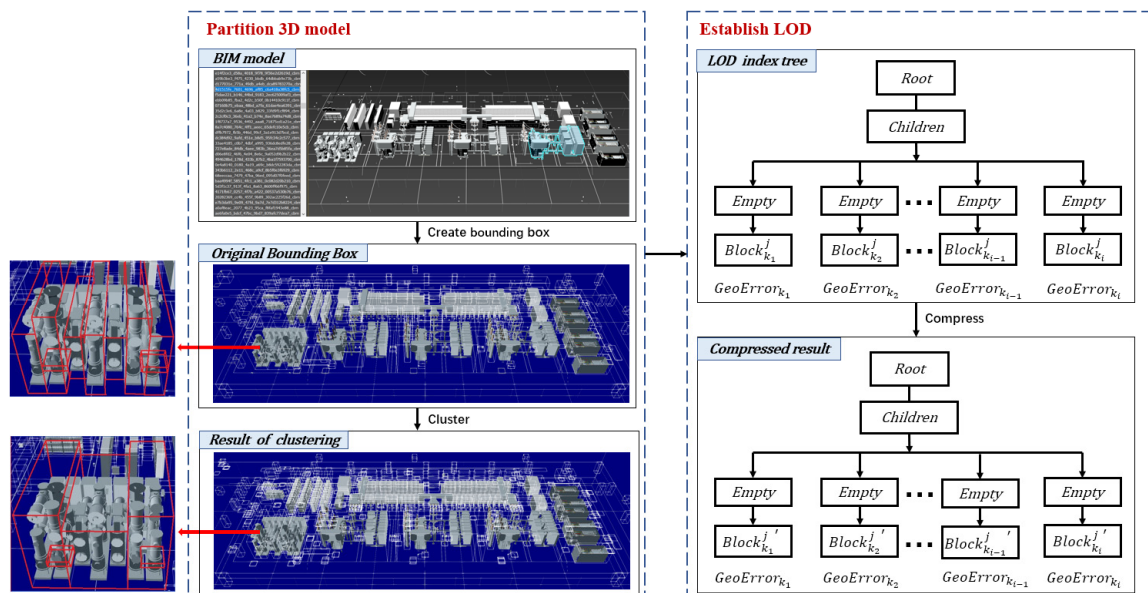


Figure 4. Overall process for data organization of the BIM model.

### 3.2.1. Tiling Method for BIM Models

The tiling method proposed herein is an improvement over the R-tree. The BIM model is partitioned via “bottom-up” clustering, as shown in Figure 4. BIM models consist of multiple subassemblies such as doors and windows. First, according to the minimum bounding cuboids of the subassemblies, the BIM model is partitioned into multiple independent blocks. Subsequently, the distance between each block is calculated. Next, the system evaluates whether the sum of the faces contained in the two closest blocks is less than the threshold (Formula (3)). If it is less than the threshold, the overall minimum bounding cuboids are calculated and the minimum bounding cuboids of each will be deleted. Clustering is continued until the number of faces in each block meets the threshold. During partitioning, only the final tiling result is preserved; tiling results from the other stages are discarded. Algorithm 1 is as follows:

**Algorithm 1** Tiling method for BIM model**Input:**Original BIM model  $M(v, vn, vt, f)$ **Initialize:**

- (1) Analyze the original BIM model  $M$  and determine the number of subassemblies  $Objnum$ .
- (2) Store vertex  $v$ , normal vector coordinate  $vn$ , texture coordinate  $vt$ , and face index  $f$  in the corresponding array of subassembly  $Obj(v, vn, vt, f)$ .
- (3) Initialize the two-dimensional array of distance  $Dis\_array$  and one-dimensional array of minimum bounding cuboids  $Box$ .
- (4) According to Formula (3), calculate the threshold of clustering  $Cluster\_threshold$ .

**Model Tiling:**

**For each**  $Obj_i$ , calculate its minimum bounding cuboid  $Box_i$ , and the barycenter  $W_i$  (calculated using Formula (4)).

**For each**  $Obj_i$ , calculate the distance  $D_{ij}$  between itself and other  $Obj_j$  based on  $W_i$  and  $W_j$ , and add  $D_{ij}$  to  $Dis\_array$ .

**While** the size of  $Dis\_array$  is not zero:

Retrieve the minimum distance  $D_{ij}$  from  $Dis\_array$  and obtain the corresponding subassemblies  $Obj_i$  and  $Obj_j$ .

**If** the sum of  $Obj_i(f)$  and  $Obj_j(f)$  is less than  $Cluster\_threshold$ :

Update the  $Box_i$  to the overall minimum bounding cuboid of  $Obj_i$  and  $Obj_j$ , organize all geometrical data in  $Obj_j(v, vn, vt, f)$  into  $Obj_i(v, vn, vt, f)$ , and delete  $Box_j$  and  $Obj_j$ .

According to Formula (5), recalculate the overall barycenter  $W_i$ .

Delete  $D_j$  in  $Dis\_array$  and recalculate the distance  $D_i$  between  $Box_i$  and other  $Box$ .

**Else:**

Delete  $D_{ij}$  and  $D_{ji}$  in  $Dis\_array$ .

**Output:**Partitioned BIM model  $Obj(v, vn, vt, f)$ 

$$Cluster_{threshold} = M(f)/Objnum \quad (3)$$

$$x', y', z' = \left( \frac{\sum_{n=1}^m \frac{x_{n1} + x_{n2} + x_{n3}}{3}}{m}, \frac{\sum_{n=1}^m \frac{y_{n1} + y_{n2} + y_{n3}}{3}}{m}, \frac{\sum_{n=1}^m \frac{z_{n1} + z_{n2} + z_{n3}}{3}}{m} \right) \quad (4)$$

$$W_i(x_i, y_i, z_i) = \left( \frac{Obj_i(f) * x_i + Obj_j(f) * x_j}{Obj_i(f) + Obj_j(f)}, \frac{Obj_i(f) * y_i + Obj_j(f) * y_j}{Obj_i(f) + Obj_j(f)}, \frac{Obj_i(f) * z_i + Obj_j(f) * z_j}{Obj_i(f) + Obj_j(f)} \right) \quad (5)$$

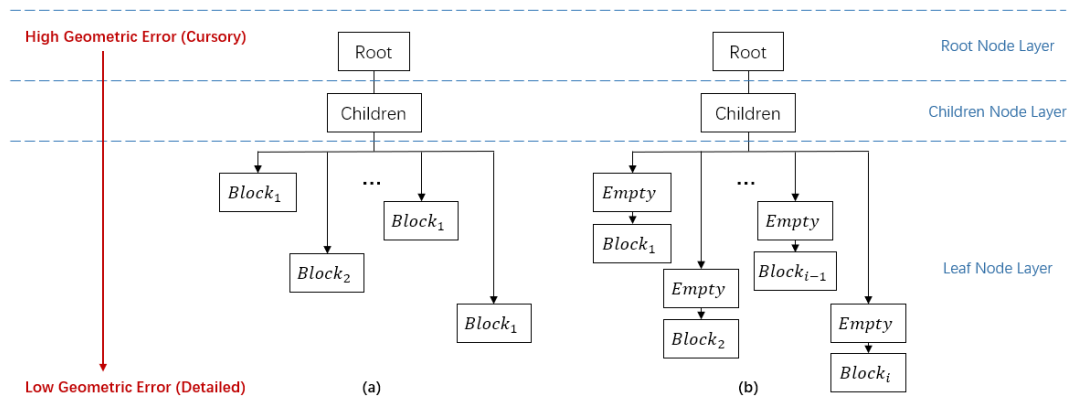
**3.2.2. LOD Method for BIM Models**

According to the format specification for 3D Tiles, the geometric error can control the LOD of the model by affecting the SSE of the tile. The corresponding model will only need to be rendered when the SSE of the tile meets the requirements. Nevertheless, simply inserting the abovementioned tiling results into an index tree as leaf nodes and setting a specific geometric error value for each block, as shown in Figure 5a, does not implement the LOD. Due to the scheduling strategy of 3D Tiles, even if the SSE of the tile is significantly larger than the MaxSSE at the current viewing distance, the tile will still be rendered because there is no parent node with a lower precision to be scheduled.

In addition, when the model is initialized, Cesium will load the overall BIM model, even if the geometric error of some blocks has considerably exceeded the requirements for visualization. As a result, during initialization, the number of concurrent requests from the client will increase sharply and a large amount of 3D model data needs to be uploaded and rendered. As a result, the visualization of the 3D model will be delayed significantly, which will worsen the user experience.

To solve the abovementioned problems, based on the scheduling strategy of Cesium and the format specification of 3D Tiles, this paper proposes a “Mask Filter” strategy, as shown in Figure 5b. The core idea of this strategy is as follows. Empty parent nodes, which are the parent nodes of leaf nodes, are added to the index tree. The uniform resource identifier (URI) of the empty parent node is empty, and its geometric error is slightly greater than or equal to the geometric error of the corresponding leaf node. Thus, when the viewpoint becomes farther away and the rendering requirements of certain leaf nodes

cannot be met, the empty parent nodes of these leaf nodes will be rendered. Because the URIs of these parent nodes are empty, models need not be uploaded and some of the previously visualized models can be unloaded; this considerably reduces the burden on computer memory. Here, the empty parent node is the “Mask Filter.” Based on this “Mask Filter” strategy, the LOD method for BIM models is built as follows.



**Figure 5.** Structure of the index tree: (a) one-level index structure, and (b) “Mask Filter” index structure.

First, the unique geometric error of each block according to the tiling result is calculated. As shown in Formula (6), twice the diagonal length of the minimum bounding cuboids of a block is considered as the geometric error:

$$\text{Geometric error} = \sqrt{(X_{\max} - X_{\min})^2 + (Y_{\max} - Y_{\min})^2 + (Z_{\max} - Z_{\min})^2} \times 2 \quad (6)$$

where  $X_{\max}$ ,  $Y_{\max}$ , and  $Z_{\max}$  represent the maximum value of the block along with the X-, Y-, and Z-axes of the geographic coordinate system, respectively.

Second, a separate “Mask Filter” is set for each block.

Third, the layers are organized sequentially into the JSON index file of 3D Tiles, such that the “Mask Filter” is the parent and the block is the child.

During browsing, as the viewpoint becomes closer, leaf nodes whose geometric errors meet the threshold are rendered; consequently, the model appears more detailed. Conversely, as the viewpoint moves farther away, leaf nodes whose geometric error does not meet the threshold are unloaded and the “Mask Filter” is scheduled; as a result, the model appears rougher. Therefore, with respect to the data storage structure, the model built using this strategy only has one LOD. In terms of visualization, it achieves the effect of a hierarchical LOD, which ensures fluency in scaling.

During initialization, as some leaf nodes cannot meet the rendering requirements at the initial viewing distance, their “Mask Filter” will be automatically scheduled. In this manner, the burden of data requests by the client is reduced significantly. From the perspective of data requests and computer hardware, the “Mask Filter” reduces the amount of concurrent data requests and enables more efficient usage of the computer’s hardware resources, as compared to other methods. Lastly, from the perspective of the data storage structure, the “Mask Filter” contributes toward reducing the depth of the index tree.


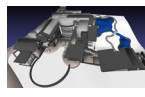
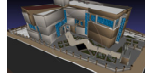
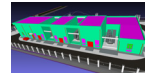
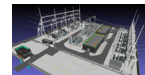
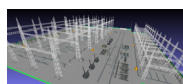
#### 4. Experiments and Discussion

To test the performance of the proposed algorithm, we implemented our algorithms in C++ on 64-bit Windows 10. The experiments were carried out on a DELL computer with an Intel Core i7-9750H (Intel, Santa Clara, CA, USA), 16 GB RAM (Samsung, Seongnam City, Gyeonggi Province, Korea), and NVIDIA GeForce GTX 1650 GPU.

The experiment consisted of two parts. First, the computing performance, rendering performance, and effect of picking and querying 3D Tiles built using the proposed method were tested. The experimental data, namely the villas, housing, offices, and three substation

models, are presented in Table 1. Second, the proposed method was compared with three existing methods: octree (Method 1), model simplification (Method 2), and semantic information (Method 3). The experimental data of second experiment was Electric Power 3, as shown in Table 1.

**Table 1.** Experimental data.

Model Name	Model Size(obj)	Number of Faces	Thumbnail Image
Villa	3.25 MB	47,370	
Housing	25.3 MB	203,187	
Offices	91.0 MB	540,964	
Electric Power 1	694 MB	5,882,088	
Electric Power 2	1.0 GB	7,403,652	
Electric Power 3	2.84 GB	11,021,840	

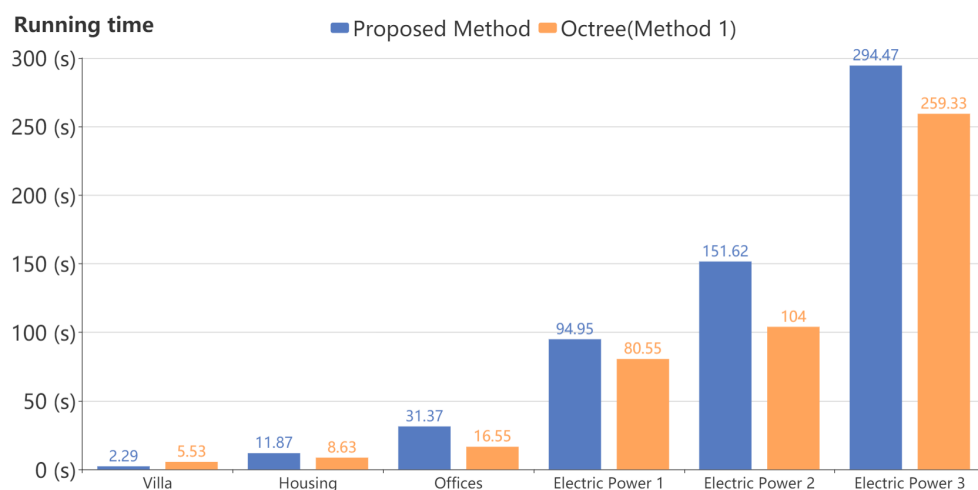
#### 4.1. Performance

The performance of the preprocessing method for BIM models was mainly tested based on computing performance, which we defined as the running time required to complete preprocessing; the rendering performance, which we defined as the frames per second (fps) at different viewing distances on the Web-based 3D model viewer; and the picking effect, which we defined as whether the picked subassemblies were complete.

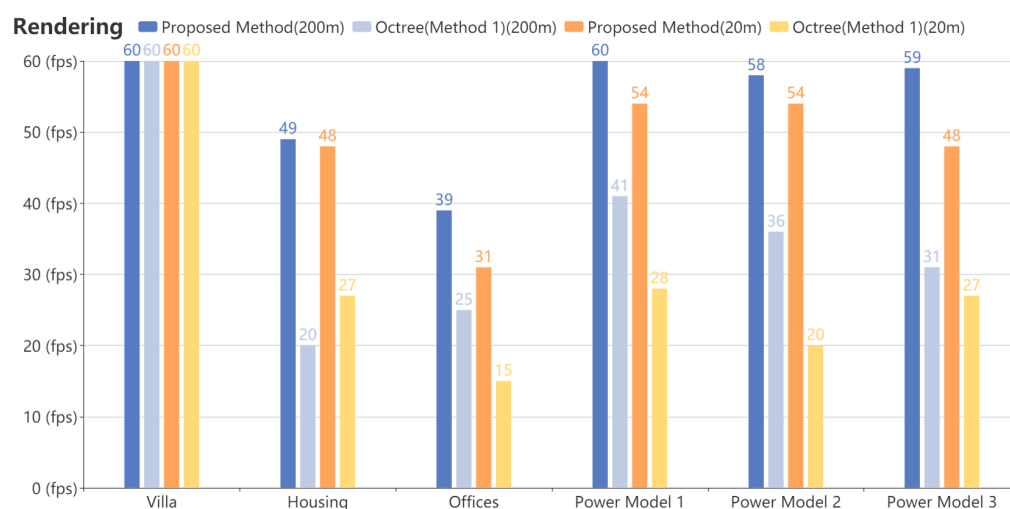
The running time required for the six processing models is shown in Figure 6; the models are processed using the proposed method and the aforementioned Method 1. Figure 6 indicates that, as the model size increases, the running time of both methods increases. The running time of Villa, the smallest model, was only approximately 2 s, whereas the running time of Electric Power 3, the biggest model, was just under 5 min. Although the overall running performance of Method 1 was slightly better than that of the proposed method, the difference in running time between the two methods is small (within seconds); hence, the methods were deemed to be comparably efficient.

The rendering performance of the model processed by two methods on the Web-based 3D model viewer is shown in Figure 7; specifically, the figure indicates the browsing frame rate of the model when being processed by different algorithms for a viewpoint at a distance of 200 m and 20 m from the model. From Figure 7, it can be seen that the rendering performance of the proposed method was better than that of Method 1 for both the viewing distances. Although the increase in model size degraded the rendering performance of both methods, the fps of the model processed using the proposed method only exhibited slight fluctuations, and high rendering performance was maintained. For the proposed method, the fps of all the experimental models exceeded 30, whereas the fps of most experimental models is approximately 50. In addition, the fps of the model processed using both methods was higher at 200 m than at 20 m, indicating that the Web is constantly rendering new blocks and that the model becomes more sophisticated. From Figure 7, it can also be seen that, when rendering the office model, the rendering performance of both methods degraded. This is because the bounding boxes of the subassemblies in the office model are almost of the same size, and the approximate diagonal length leads to

an approximate geometric error. As a result, a large number of subassemblies are loaded at a certain level when zooming in, which reduces browsing fluency. In conclusion, the rendering performance of the proposed method mainly depends on the size of the model and whether the subassemblies of the model are geometrically differentiated.



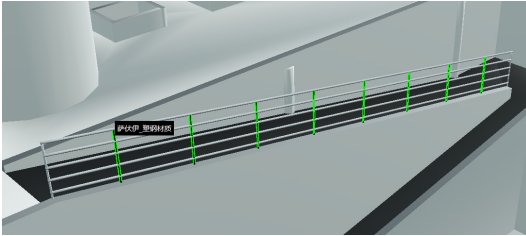
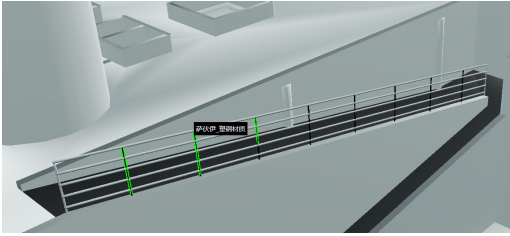
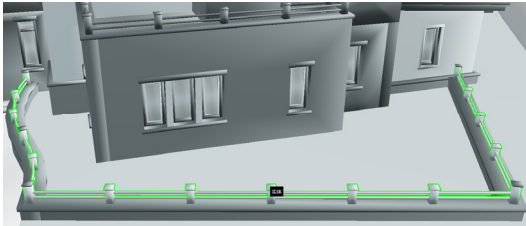
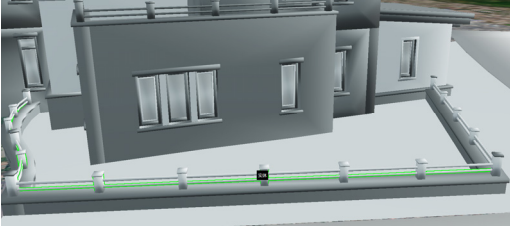
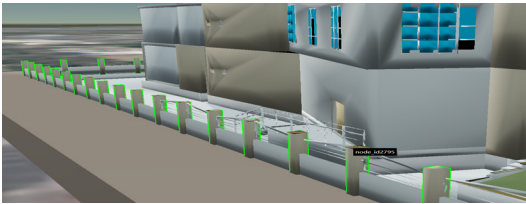
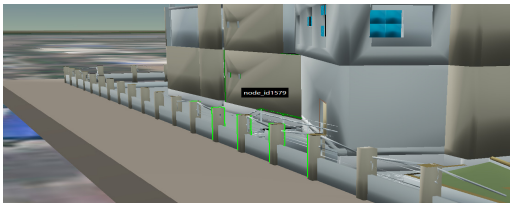
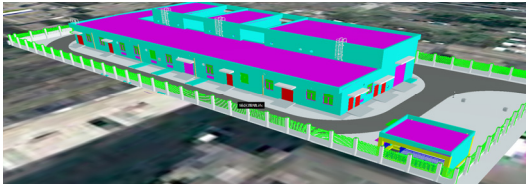

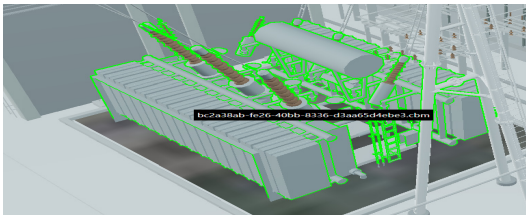
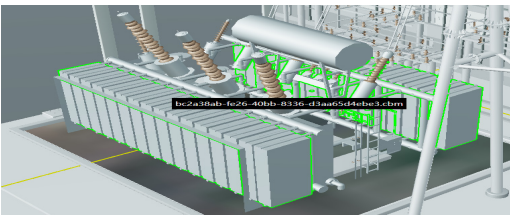
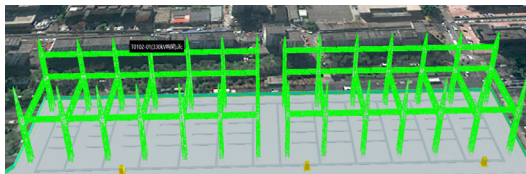
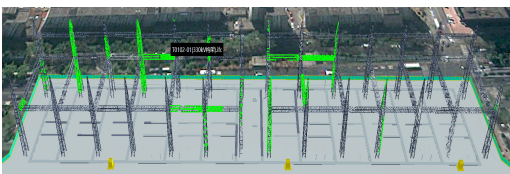
**Figure 6.** Computing performance.



**Figure 7.** Rendering performance.

The effects of picking and querying the model processed by two methods are shown in Table 2. The green portion in the table represents the result returned by one query or one pick. From the table, it is evident that the proposed method ensures the geometric integrity of the subassemblies, regardless of whether these subassemblies belong to objects such as fences (with a wide area) or substations (with a large number of faces). However, the aforementioned Method 1, which considers the faces as the minimum subdivision units, partitions the subassemblies, resulting in an incomplete query result.

Table 2. Effects of picking and querying.

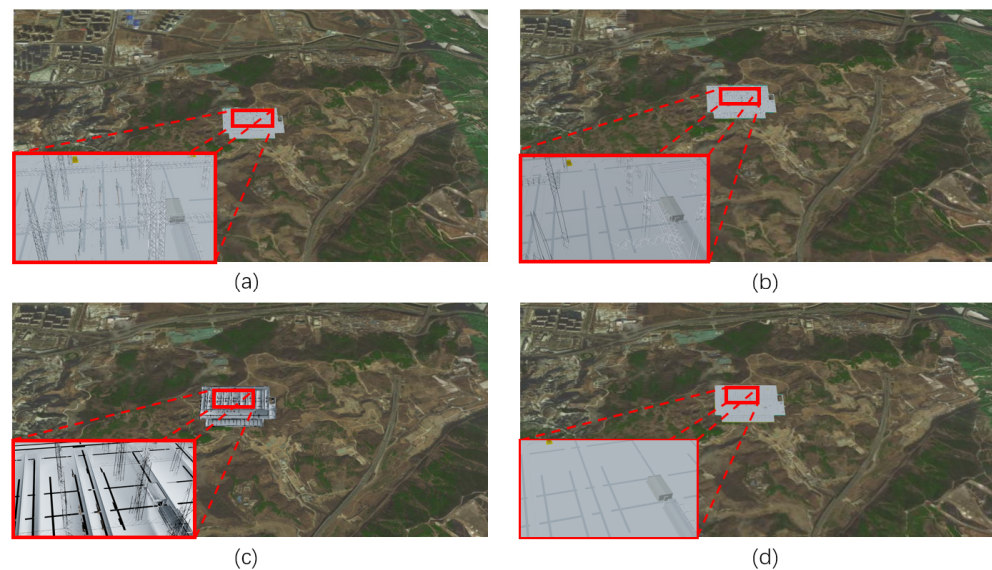
	Proposed Method	Octree(Method 1)
Villa		
Housing		
Offices		
Electric Power 1		
Electric Power 2		
Electric Power 3		

#### 4.2. Comparison of Visualization Afforded by Different Algorithms

In order to test the superiority of the proposed method, a contrast experiment was conducted, comparing the proposed method with three mainstream methods, namely Method 1, Method 2 [25], and Method 3 [16]. The Electric Power 3 model was selected for the experimental data, and the model processed by the four methods was rendered on the Web-based 3D model viewer. The remaining parameters, such as MaxSSE [26], were set to the same values. Thereafter, the LOD and browsing fluency of the model processed by the different methods and at different viewing distances were observed.

When the viewing distance is 1500 m, the viewpoint is far away from the model. From Figure 8, it is clear that the model processed by the proposed method shows the foundation

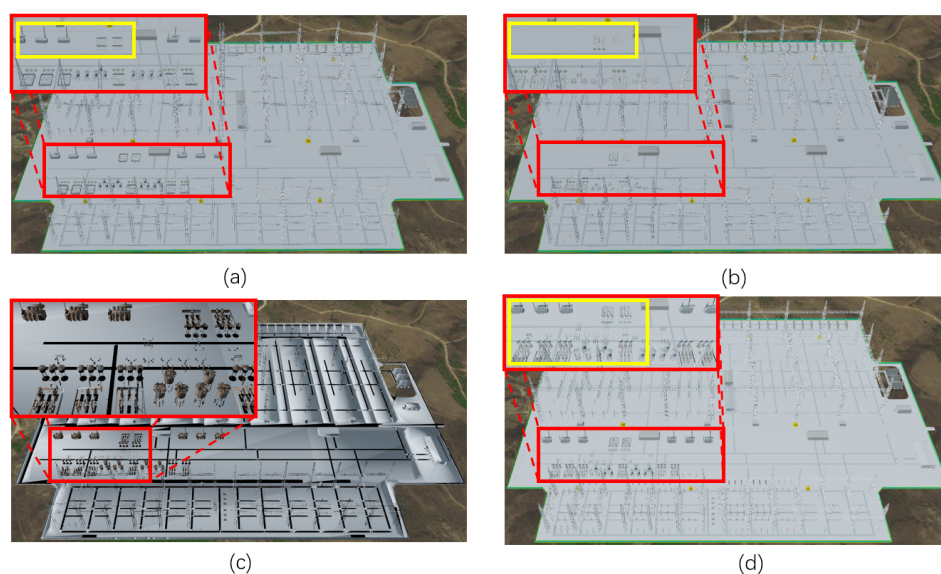
of the substation model, the large-scale transmission tower, and some larger subassemblies. The overall outline of the transmission tower remains intact, as indicated in the red box in Figure 8a. The model processed by Method 1 only shows the foundation and transmission tower of the substation model; however, the outline of the transmission tower is largely lost, as shown in the red box in Figure 8b. Furthermore, the model processed by Method 2 shows the foundation and transmission tower and also some smaller subassemblies, which are excessively detailed. Additionally, due to model simplification, the outline of the transmission tower is considerably fuzzy, as shown in the red box in Figure 8c. Moreover, the model processed by Method 3 only shows the foundation, and the overall outline of the model is largely missing, as shown in the red box in Figure 8d.



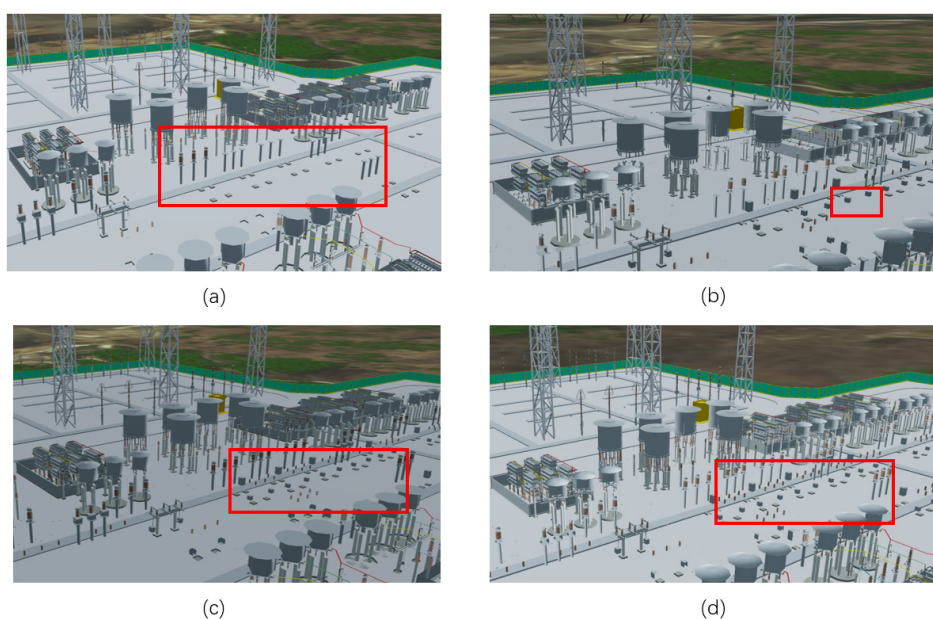
**Figure 8.** Model visualization at a viewing distance of 1500 m when using: (a) proposed method, (b) Method 1, (c) Method 2, and (d) Method 3.

When the viewing distance is 300 m, the model precisely covers the entire visualization window. As shown in Figure 9, the model processed using the proposed method shows appropriate refinement for some moderately sized subassemblies such as transformers, as indicated in the yellow box in Figure 9a; this provides a clearer view of the details of the model. The model processed using Method 1 also shows some refined subassemblies; however, compared with the proposed method, the rendered details are relatively few, as shown in the yellow box in Figure 9b. Furthermore, the models processed by Method 2 and Method 3 both present a larger number of smaller subassemblies, as shown in the red box in Figure 9c and the yellow box in Figure 9d, respectively, which makes parts of the model appear crowded.

When the viewing distance is 50 m, the viewpoint is closer to the model. As can be seen from Figure 10, many subassemblies are added to the model processed by the proposed method. The details of the model become richer, while certain excessively small subassemblies are still appropriately neglected, as shown in the red box in Figure 10a. The model processed by Method 1 also exhibits enriched details; however, some of the excessively small subassemblies such as insulator strings and boxes are also presented, as shown in the red box in Figure 10b. Method 2 and Method 3 both yield complete details of the model, and as a result, the overall model is relatively chaotic, as shown in the red box in Figure 10c,d, respectively.



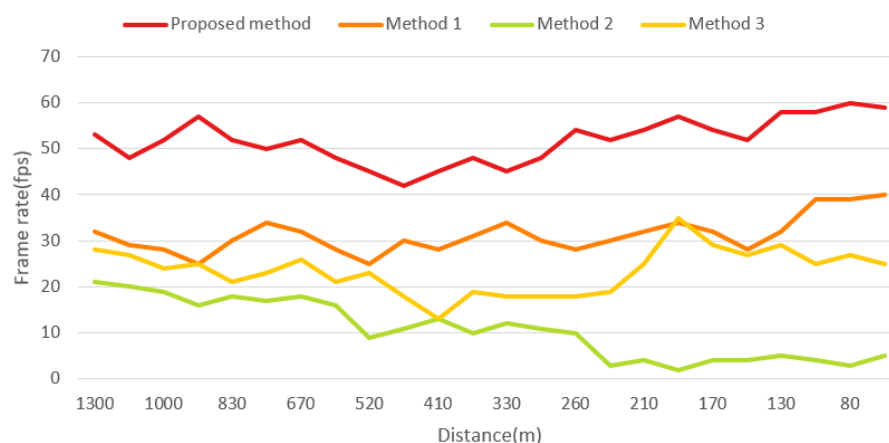
**Figure 9.** Model visualization at a viewing distance of 300 m when using: (a) proposed method, (b) Method 1, (c) Method 2, and (d) Method 3.



**Figure 10.** Model visualization at a viewing distance of 50 m when using: (a) proposed method, (b) Method 1, (c) Method 2, and (d) Method 3.

As the viewpoint gradually moves from 1300 m to 50 m away from the model, the browsing frame rate of the model processed by different methods fluctuates constantly, as illustrated in Figure 11. For the proposed method, throughout the browsing process, the browsing frame rate fluctuates from 40 to 60 fps. In the interval of 800–500 m, the browsing frame rate reduces because certain small subassemblies of the model are scheduled. Beyond 300 m, due to tiling, some blocks located outside the visualization window are unloaded, and as a result, the browsing frame rate increases. This indicates that the proposed method can achieve excellent browsing fluency throughout the browsing process. For Method 1, the browsing frame rate fluctuated from 20 to 40 fps throughout the browsing process. The fluctuations in the fps from 800 to 500 m and beyond 300 m were consistent with those when using the proposed method. For Method 2, the overall browsing frame rate was below 20 fps. There was an evident three-level step in the fluctuation of the frame rate, which corresponded to the loading of the secondary simplified model in the interval of

600–500 m and the most refined model in the interval of 260–200 m. However, the browsing fluency was poor. Lastly, for Method 3, the browsing frame rate fluctuated from 10 to 30 fps throughout the browsing process, and the fluctuations in the local interval were consistent with those when using the proposed method. However, the overall browsing fluency was relatively poor, compared to that afforded by the proposed method.



**Figure 11.** Fps afforded by the methods at different viewing distances.

In addition, the proposed method, Method 2, and Method 3 were all able to detect complete subassemblies. Although Method 1 guarantees the data size of a single tile, it was unable to detect complete subassemblies.

In summary, at different viewing distances, the proposed method can adaptively render each subassembly according to its size; therefore, it can more accurately control the scheduling for each subassembly and make the LOD more prominent. Owing to the tiling method, the integrity of the subassembly and contour of the model is better maintained than with previously developed models. Moreover, the LOD method and the Draco compression algorithm were combined to ensure smoother visualization. Across our experimental validations, in terms of both the quality of visualization and the browsing fluency, the proposed method demonstrated superior performance.

## 5. Conclusions

To generate smooth visualizations of complex BIM models on the Web-based 3D model viewer, this paper proposed a highly efficient data organization method based on 3D Tiles. This method does not simplify the model. Based on retaining the original model, the model is partitioned into several blocks according to the subassemblies of the BIM model and the data size of a single tile. Thereafter, the concept of the “Mask Filter” is introduced to achieve the LOD of the model. Through several contrast experiments, we demonstrate that the proposed method achieves high efficiency, as compared to traditional methods. With respect to the rendering performance, the proposed method considerably outperforms traditional algorithms, thereby tremendously improving the browsing fluency of complex BIM models on the Web-based 3D model viewer. With respect to the effect of picking and querying, the proposed method can guarantee the integrity of picked subassemblies. In conclusion, the proposed method can achieve high-efficiency visualization for BIM models with large data volume on the Web-based 3D model viewer. Further, through the proposed method, we can smoothly interact with these BIM models in Web-3D GIS, which gives new impetus to the integrated application of BIM and Web-GIS.

Nevertheless, certain aspects of this research warrant further optimization. First, only an experimental platform on a PC was considered. Given the popularization of mobile devices, the visualization of large-scale 3D models on mobile devices should also be considered. In addition, if the sizes of the blocks obtained via clustering are identical, it is critical to further consider methods to set the geometric error for each block, such that

the geometric distinction between each block is increased. In future research, we plan to focus on these aspects to further improve the rendering performance and applicability of this method.

**Author Contributions:** Conceptualization, Wenxiao Zhan and Jing Chen; methodology, Wenxiao Zhan and Yuxuan Chen; software, Wenxiao Zhan; validation, Wenxiao Zhan, Yuxuan Chen and Jing Chen; formal analysis, Wenxiao Zhan and Jing Chen; investigation, Wenxiao Zhan and Yuxuan Chen; resources, Yuxuan Chen; data curation, Yuxuan Chen; writing—original draft preparation, Wenxiao Zhan; writing—review and editing, Wenxiao Zhan, Yuxuan Chen and Jing Chen; visualization, Wenxiao Zhan; supervision, Jing Chen; project administration, Wenxiao Zhan and Yuxuan Chen. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the National Key R&D Program of China (grant number 2018YFB0505302).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author. The data are not publicly available due to restrictions of privacy and morality.

**Acknowledgments:** The authors would like to thank the anonymous reviewers and the editor, whose comments and suggestion greatly improved this paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Howell, S.; Hippolyte, J.-L.; Jayan, B.; Reynolds, J.; Rezgui, Y. Web-Based 3D Urban Decision Support through Intelligent and Interoperable Services. In Proceedings of the 2016 IEEE International Smart Cities Conference (ISC2), Trento, Italy, 12–15 September 2016; pp. 1–4.
- Liang, J.; Gong, J.; Li, W. Applications and Impacts of Google Earth: A Decadal Review (2006–2016). *ISPRS J. Photogramm. Remote Sens.* **2018**, *146*, 91–107. [CrossRef]
- Qiu, G.; Chen, J. Web-Based 3D Map Visualization Using WebGL. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 759–763.
- Azhar, S. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadersh. Manag. Eng.* **2011**, *11*, 241–252. [CrossRef]
- Hooper, M. Automated Model Progression Scheduling Using Level of Development. *Constr. Innov.* **2015**, *15*, 428–448. [CrossRef]
- 3DTiles. Available online: <https://github.com/AnalyticalGraphicsInc/3d-tiles/tree/master/specification#tile-format-specifications/> (accessed on 6 July 2020).
- I3S. Available online: <http://docs.opengeospatial.org/cs/17-014r7/17-014r7.html/> (accessed on 7 April 2021).
- S3M. Available online: <https://github.com/SuperMap/s3m-spec/tree/master/Specification/> (accessed on 22 June 2021).
- Yang, L.; Zhang, L.; Ma, J.; Xie, J.; Liu, L. Interactive Visualization of Multi-Resolution Urban Building Models Considering Spatial Cognition. *Int. J. Geogr. Inf. Sci.* **2011**, *25*, 5–24. [CrossRef]
- Chen, Y.; Shooraj, E.; Rajabifard, A.; Sabri, S. From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium. *ISPRS Int. J. Geo. Inf.* **2018**, *7*, 393. [CrossRef]
- Mao, B.; Ban, Y.; Laumert, B. Dynamic Online 3D Visualization Framework for Real-Time Energy Simulation Based on 3D Tiles. *ISPRS Int. J. Geo. Inf.* **2020**, *9*, 166. [CrossRef]
- Song, Z.; Li, J. A Dynamic Tiles Loading and Scheduling Strategy for Massive Oblique Photogrammetry Models. In Proceedings of the 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 27–29 June 2018; pp. 648–652.
- Kulawiak, M.; Kulawiak, M.; Lubniewski, Z. Integration, Processing and Dissemination of LiDAR Data in a 3D Web-GIS. *ISPRS Int. J. Geo. Inf.* **2019**, *8*, 144. [CrossRef]
- Lu, M.; Wang, X.; Liu, X.; Chen, M.; Bi, S.; Zhang, Y.; Lao, T. Web-Based Real-Time Visualization of Large-Scale Weather Radar Data Using 3D Tiles. *Trans. GIS* **2021**, *25*, 25–43. [CrossRef]
- Herman, L.; Rusznák, J.; Řezník, T. Flood Modelling and Visualizations of Floods Through 3D Open Data. In *International Symposium on Environmental Software Systems*; Hřebíček, J., Denzer, R., Schimak, G., Pitner, T., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 139–149.
- Xu, Z.; Zhang, L.; Li, H.; Lin, Y.-H.; Yin, S. Combining IFC and 3D Tiles to Create 3D Visualization for Building Information Modeling. *Autom. Constr.* **2020**, *109*, 102995. [CrossRef]
- Kim, J.; Hong, C.; Son, S. A Light Weight Algorithm for Large-Scale BIM Data for Visualization on a Web-Based GIS Platform. *Build. Inf. Model. (BIM) Des. Constr. Oper.* **2015**, *149*, 355–367.

18. Hu, Z.-Z.; Yuan, S.; Benghi, C.; Zhang, J.-P.; Zhang, X.-Y.; Li, D.; Kassem, M. Geometric Optimization of Building Information Models in MEP Projects: Algorithms and Techniques for Improving Storage, Transmission and Display. *Autom. Constr.* **2019**, *107*, 102941. [\[CrossRef\]](#)
19. Chen, J.; Li, J.; Li, M. Progressive Visualization of Complex 3D Models Over the Internet. *Trans. GIS* **2016**, *20*, 887–902. [\[CrossRef\]](#)
20. Chen, J.; Li, M.; Li, J. An Improved Texture-Related Vertex Clustering Algorithm for Model Simplification. *Comput. Geosci.* **2015**, *83*, 37–45. [\[CrossRef\]](#)
21. Ziolkowska, J.R.; Reyes, R. Geological and Hydrological Visualization Models for Digital Earth Representation. *Comput. Geosci.* **2016**, *94*, 31–39. [\[CrossRef\]](#)
22. Kulawiak, M.; Kulawiak, M. Application of Web-GIS for Dissemination and 3D Visualization of Large-Volume LiDAR Data. In *The Rise of Big Spatial Data*; Ivan, I., Singleton, A., Horák, J., Inspektor, T., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 1–12.
23. Cozzi, P.J. Visibility Driven Out-of-Core Hlod Rendering. Master's Thesis, University of Pennsylvania, Philadelphia, PA, USA, 2008.
24. Draco. Available online: <https://github.com/google/draco.html/> (accessed on 21 July 2020).
25. Hoppe, H. View-Dependent Refinement of Progressive Meshes. *ACM SIGGRAPH* **1997**, 189–198. [\[CrossRef\]](#)
26. Cesium3DTileset. Available online: <https://cesium.com/docs/cesiumjs-ref-doc/Cesium3DTileset.html/> (accessed on 6 July 2020).