International Journal of Geo-Information

MDPI

Article

Visibility-Based R-Tree Spatial Index for Consistent Visualization in Indoor and Outdoor Scenes

Chengpeng Li [1,2,3,4], Xi Kuai [1,*], Biao He [1,2], Zhigang Zhao [1,3], Haojia Lin [1], Wei Zhu [5], Yu Liu [4] and Renzhong Guo [1]

1 Research Institute for Smart Cities, School of Architecture and Urban Planning, Shenzhen University, Shenzhen 518060, China; lichengpeng@szu.edu.cn (C.L.)
2 Polytechnic Center for Territory Spatial Big-Data, Ministry of Natural Resources, Shenzhen 518060, China
3 Shenzhen Key Laboratory of Digital Twin Technologies for Cities, Shenzhen 518060, China
4 Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Natural Resources, Shenzhen 518040, China
5 School of Resource and Environmental Sciences, Wuhan University, Wuhan 430079, China
* Correspondence: kuaixi@szu.edu.cn

Abstract: (1) Background: The smart city management system, with GIS technology as its core, is based on realistic visualization of multiple types of 3D model data syntheses. However, the efficiency barriers to achieving smooth and continuous visualization from outdoor scenes to small indoor scenes remain a challenge. (2) Methods: This paper uses the visibility prediction method to obtain potential visual sets at three levels—outdoor, indoor and outdoor connection, and indoor—and constructs an R-tree spatial index structure for organizing potential visual sets. By integrating these potential visible sets with spatial indexes, scene visualization can be carried out effectively. (3) Results: A near-reality indoor and outdoor scene was used for experimentation, resulting in stable 10% fluctuation visual frame rates around 90 FPS. (4) Conclusions: Spatial indexing methods that combine potential visible sets can effectively solve the continuity and stability problem of indoor and outdoor scene visualization in smart city management systems.

Keywords: potential visual sets; smart city management system; spatial index; consistent visualization

check for updates

Citation: Li, C.; Kuai, X.; He, B.; Zhao, Z.; Lin, H.; Zhu, W.; Liu, Y.; Guo, R. Visibility-Based R-Tree Spatial Index for Consistent Visualization in Indoor and Outdoor Scenes. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 498. https://doi.org/10.3390/ijgi12120498

Academic Editors: Wolfgang Kainz and Florian Hruby

Received: 17 September 2023
Revised: 24 November 2023
Accepted: 10 December 2023
Published: 12 December 2023

Copyright: © 2023 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/).

1. Introduction

The 3D simulation management system, catering to the requirements of smart cities, is progressively adopting realistic 3D GIS technology, which has emerged as a pivotal tool for governmental agencies in achieving territorial space management and facilitating livelihood services [1,2]. The operation of this system relies on constructing superfine 3D scenes by integrating various types of geospatial data and establishing high-performance data scheduling mechanisms to realize the visual basis of urban space management, which constitutes the foundation of a smart city management system.

The visualization of a smart city system encompasses both expansive outdoor scenes and intricate indoor scenes. Effectively visualizing the combination of these complex scenes faces challenges in terms of data loading efficiency and stable scheduling for large datasets. Existing visualization systems typically rely on the 3D Tiles structure for data scheduling and use a spatial distribution to convert the data into a mesh-structured triangular surface collection [3–5]. However, indoor scenes and outdoor scenes possess distinct spatial characteristics. For instance, objects in outdoor scenes are scattered with a wide field of vision, whereas building interiors exhibit high spatial density and obstructed views. For example, BIM-based fine buildings synchronize their physical realization by constructing a digital twin 3D model with accurate geometry and realistic appearance. The browsing mode of the 3D simulation management system needs to simulate human visual effects, enabling users to seamlessly transition between indoor and outdoor scenes during browsing sessions. Traditional single-visual scheduling strategies fail to effectively overcome the challenges posed

by continuous visualization from large outdoor scenes to small indoor ones, particularly in addressing frame rate fluctuations caused by sudden scene changes.

This paper presents a novel approach called the visibility-based spatial index (VESI), which combines potential visible sets with spatial indexes. Utilizing this index to schedule spatial data significantly improves the continuity and stability of transitions in visualizing scenes, especially between indoor and outdoor scenes.

2. Related Works

The efficient visualization of 3D scenes is a research area that encompasses three key aspects: processing the original 3D data, establishing spatially based indexing technology, and pre-calculating visibility results.

2.1. Improved Visualization with Data Processing

In order to achieve the visual implementation of complex scenes, researchers preprocess original three-dimensional data to generate customized versions based on specific requirements. The concept of Level of Detail (LoD) was introduced by Clark in 1976 [6], which involves creating multiple versions with varying levels of detail for the same object in advance. When an object occupies a smaller portion of the screen, a coarser model can be used to represent it; otherwise, a more detailed model is used.

This approach has been widely applied in urban and architectural modeling. To visualize urban data, preconstructed model objects with five levels of detail are available for different applications [7,8]. For instance, the roughest two-dimensional vector plane model (LoD0) can be used for urban plot planning, while the three-dimensional box model (LoD1) is suitable for expressing space ownership, and a more refined LoD3 model is required for detailed facility management [9]. However, since five LoDs may not suffice to achieve a comprehensive digital twin description from a geometric perspective, Filip proposes an improved set of 16 LoDs that focus on grading a building's external geometry [10]. Löwner suggests extending CityGML LoD by providing an extension mechanism for user-defined LoD to refine it further [11]. To address the issue of transitioning from outdoor scenes to indoor scenes in urban settings, Tang develops a full-space LoD integrating indoor and outdoor using CityGML and forms 20 semantic LoD levels that extend structure, connectivity, volume, and other contents [12]. The trend toward refining the hierarchy of LoDs continues. Chen classifies models according to visual variables and constructs possible LoDs to form a four-layer coding capability LoD structure that makes preprocessing data more feasible [13]. As each application has its fixed level of detail design requirement leading to excessive preprocessing data issues; Tang proposes a bottom-up approach by assembling the lowest level of model primitives into an assembly set [14].

Data processing technology enables reducing a substantial volume of 3D data into a compact multi-level data subset while using an appropriate data scheduling method to load the necessary subset under diverse circumstances. However, this approach exhibits certain drawbacks: the utilization of multi-level data subsets leads to a significant increase in computer storage space and results in considerable data redundancy; furthermore, the processed data subset represents lossy information that partially lacks characteristic details from the original dataset. Especially in visualization, the model between the two LoDs brings the observer with an abrupt visual difference.

2.2. Improved Visualization Using Spatial Index Technology

In order to mitigate information loss, spatial indexing technology is used as a means to calculate and record the distribution characteristics of data. By effectively enhancing the retrieval speed of model data, spatial indexing enables faster scheduling and rendering, thereby ensuring a consistent improvement in visualization outcomes.

Zhu utilized K-means clustering results to achieve the node grouping of an R-tree and establish a spatial index for a small model, which exhibits excellent visualization effects in virtual geographic environments [15]. In the context of limited computing

resources, Zhang used spatial indexing technology to enhance the adaptive rendering capability of large and complex spatial data [16]. To cater to the rapid visualization requirements of extensive scenes like geographical space, Yang used a quadtree index for global data organization and a three-dimensional spatial index for local scope [17]. As data structures become increasingly intricate, Ke proposed utilizing a combination of R-trees and B*-trees to establish efficient retrieval mechanisms for spatiotemporal data [18]. When applied to the IoT network, parallel queries involving grids and trajectories can be processed in real-time using an integrated spatial index model [19]. In ocean electronic chart visualization applications, Yu divided ocean objects into corresponding R-tree nodes based on their minimum bounding boxes, significantly enhancing object visualization efficiency in sparse ocean spaces [20]. Liu introduced the HiIndex as a novel spatial index enabling real-time and interactive visualization of large-scale vector data by partitioning global geographic scopes using TQ-tree's quadtree structure; each TQ-tree node represents the spatial extent of specific rules [21], ultimately facilitating rapid visualization for large-scale vector datasets. Wu used a B+ tree-based spatial index to accelerate voxel data processing in tunnel geological environments. Dynamic volume calculation was used to integrate dense collective elements representing complex internal heterogeneous information with sparse voxels representing structural information, thereby achieving accelerated volume rendering and rendering capabilities [22].

The utilization of spatial indexing enhances data retrieval efficiency without altering the original data structure, thereby facilitating faster loading of 3D data in visual data scheduling. However, this approach exhibits certain limitations as it solely considers the geographical distribution of spatial objects and overlooks model visibility. Consequently, during data scheduling in the visualization pipeline, a substantial amount of invisible redundant information may be inadvertently imported into rendering memory, resulting in diminished visualization efficiency.

2.3. Predictive Visibility-Based Visualization Enhancement

Similar to the application of level-of-detail techniques, visibility computation holds significant value in the real-time visualization of large-scale models and data volumes [23]. Within the graphics rendering pipeline, this process involves calculating visible objects within a scene, eliminating invisible components, and ultimately generating an image frame. Visibility calculation can be time-consuming, particularly when dealing with a substantial number of objects from specific perspectives such as interior scenes [24]. The precomputed storage of predetermined visible results enhances both the speed and stability of visualizations.

Masehian uses predictive visibility to construct a visibility graph and integrate a Voronoi diagram to establish a knowledge reserve for robot motion planning, thereby enabling a rapid and stable response speed in wayfinding [25]. Roden suggests that doors within indoor environments can traverse the line of sight and utilizes a ray approximation calculation method across doorways to streamline occlusion elimination, thus enhancing the computational efficiency of potential visible sets [26]. The utilization of ray detection for visible set sampling is susceptible to erroneous judgments. By incorporating the calculation of spatial occupancy of visible objects [27] and grid-based preprocessing techniques [28], a harmonious balance between accuracy and efficiency is achieved. In recent years, Wang proposed utilizing grid predictive visibility for geospatial data scheduling, facilitating three-dimensional navigation and visualization integration both indoors and outdoors within smart cities [29]. When encountering rooms with dense spatial features, a multi-viewpoint joint sampling approach enhances the accuracy of visible object extraction while further improving information fidelity in visualization effects [30].

The spatial locations that viewpoints can reach are infinite for a large scene, resulting in high computational costs when collecting visual sets at each location. Consequently, this leads to increased storage costs and query time during retrieval scheduling [31]. Hladky leverages the computing power support of GPU to establish an estimation function by

performing motion prediction on camera frames [32]. The effectiveness of this estimation function directly impacts the accuracy of the visualization effect. Additionally, a combination of PVSs (potential visible sets) and spatial indexing has been proposed, where the contours of potential visible sets are stored in an octree structure to accelerate rendering scheduling. This method demonstrates excellent performance for small mesh models (e.g., Stanford bunny) [33].

The technology of predictive visibility calculates and stores visible objects in specific spatial positions in advance, enabling the direct use of predictive results for visual data scheduling. Precomputation involves a prediction process that estimates physical objects likely to be observed at particular spatial locations. The more precise the spatial position, the more accurate the prediction result becomes. When predictions can be made for every three-dimensional space point, they closely approximate real-world situations.

3. Methodology

3.1. Context

In a three-dimensional urban management system, users have the flexibility to adjust their viewing angle and navigate through the city as if they were using Google Earth. Alternatively, they can opt for a realistic exploration of both indoor and outdoor spaces within physical constraints. To achieve seamless visual transitions at different regions, an effective data processing and scheduling method is required that considers outdoor scenes as well as highly detailed local models or indoor scenes. By combining predictive visibility with spatial indexing, continuous visualization is made possible by simulating the relational dynamics between the observer (i.e., viewpoint) and the observed (i.e., spatial object).

3.2. Connotation and Mathematical Representation of VESI

The essence of VESI lies in the organization of visual outcomes achievable within a specific spatial location. Its mathematical representation entails the mapping between three-dimensional physical space and three-dimensional visual space, thereby endowing VESI with the characteristics of a four-dimensional space that amalgamates physical and visual domains. This relationship is denoted by a mapping function.

$$f(Viewpoint\ Space) \sim f(Visual\ World) \tag{1}$$

In the formula, $f(Viewpoint\ Space)$ represents a region in which the viewpoint is situated. This region is defined as an integral of the viewpoint along the X-, Y-, and Z-directions, and is referred to as viewpoint space. On the other hand, $f(Visual\ World)$ denotes all visible results that can be observed within this scope of viewpoint space. In essence, it corresponds to the spatial objects associated with this particular viewpoint. The integral form of $f(Viewpoint\ Space)$ is:

$$f(Viewpoint\ Space) = \iiint f(Viewpoint_x, Viewpoint_y, Viewpoint_z)\,dxdydz \tag{2}$$

In practical calculations, it is infeasible to compute all viewpoints within the entire space. Therefore, a discrete sampling method is used to approximate the expression and achieve visibility prediction results. The resulting viewpoint space takes on a set form as follows:

$$f(Viewpoint\ Space) \approx \{f(Viewpoint_1), f(Viewpoint_2), \ldots, f(Viewpoint_n)\} \tag{3}$$

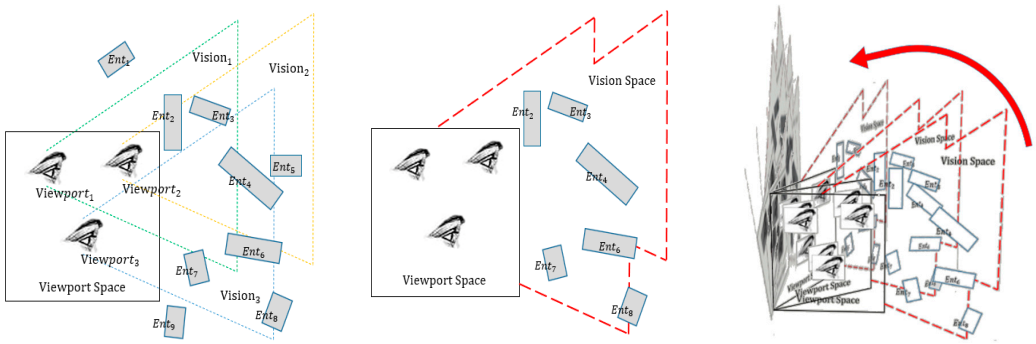
The term $f(Viewpoint_i)$ denotes the entirety of tangible entities that are visible from a three-dimensional spatial location in all directions.

Specifically, $f(Visual\ World)$ represents a comprehensive three-dimensional visual outcome that encompasses two essential components. The first component comprises a collection of viewing units corresponding to all viewpoints in the viewpoint space. The second component is the continuous viewing space that corresponds to the viewpoint space. When expressing viewpoint space with set approximation, the visual world can be

defined as the union of omnidirectional view space and its associated set of visual units under multiple viewpoints. Thus, in the following formula, *Entity* denotes the visual unit while *Vision* represents the visible spatial domain.

$$f(Visual\ World) = \bigcup_{vp=1}^{n}\left(\sum Entity_{i_{vp}}\right) + \bigcup_{vp=1}^{n}\left(\sum Vision_{i_{vp}}\right) \quad (4)$$

The schematic diagram in Figure 1 illustrates the viewpoint space and the visual world. The collection of units visible to the three viewpoints within the viewpoint space, along with their corresponding view field (Figure 1a), collectively constitute a unidirectional visual outcome (Figure 1b). By overlaying comprehensive visual outcomes, the visual world corresponding to the viewpoint space is formed (Figure 1c).



(**a**) (**b**) (**c**)

Figure 1. Viewpoint space and the visual world: (**a**) the spaces and entities; (**b**) the visual results of a single direction; and (**c**) the entire 3D visual world.

In visual computing, an *Entity* refers to the fundamental visual element involved in computations, encompassing basic points, line segments, triangulars, mesh surfaces, volume units, and model objects.

3.3. The Construction Procedure of VESI

The construction procedure of VESI primarily encompasses viewpoint space subdivision, potential visual set detection strategy, spatial index establishment, and matching incremental data scheduling mechanism. The relationships among each component is illustrated in Figure 2.

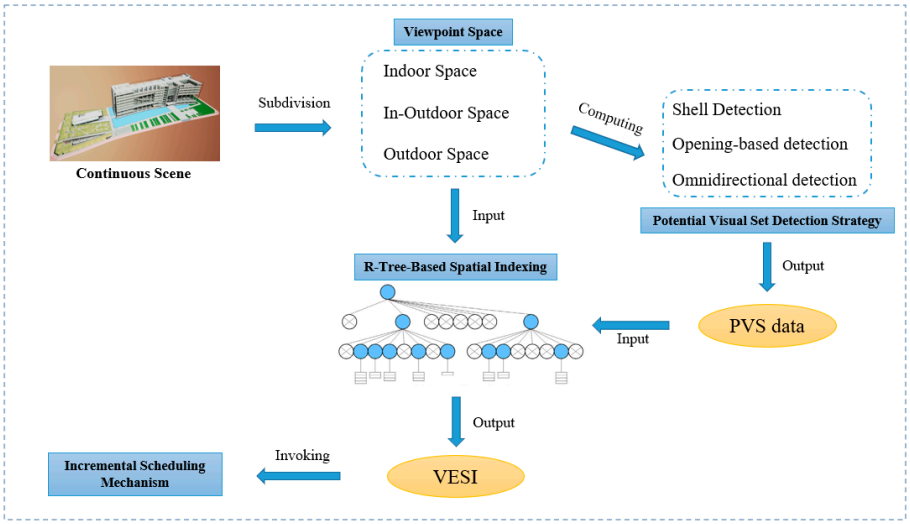


Figure 2. The relationships among each component of VESI.

3.3.1. Viewpoint Space Subdivision

Viewpoint space refers to the area accessible to the observer (camera) during the viewing process, which can be categorized into three types: outdoor space, indoor–outdoor transition space, and interior space. The three types collectively constitute a continuous spatial transition between the interior and exterior and possess significant characteristics in terms of visibility observation.

- Outdoor space is defined as a region with a certain distance from the building where observers have access to an empty area with a relatively wide field of view and are often able to see the outer surface of buildings at a distance.
- In–outdoor space represents the transitional zone between the interior and exterior spaces. The creation of in–outdoor space is achieved by strategically establishing the distances before and after entering a building. Within these spaces, observers are afforded the opportunity to simultaneously perceive both the interior objects through windows and doors, as well as the exterior surfaces of neighboring buildings.
- Interior spaces are located at a certain distance after observers completely enter a building, where they can only observe indoor objects that are more densely distributed than those found in outdoor spaces.

According to the visibility characteristics in the three types, the continuous scene data are meticulously structured at the primitive level, including building shell extraction and reorganization of indoor entities. The purposes are (1) in outdoor observations, treating the building shell as an independent entity can reduce data redundancy by loading it only once, while disregarding indoor data aids in streamlining data scheduling and (2) in indoor space observation, indoor objects are balanced and rearranged based on spatial proximity relationships to enhance visual computation efficiency.

The process of shell extraction involves screening doors, windows, walls, columns, and top floors of the building based on semantics. Subsequently, the intersection relationship between each object and the transitional spaces connecting indoors and outdoors is individually assessed to identify intersecting objects recorded as shells. All the filtered objects are grouped as a building shell object.

Regarding interior entity reorganization, entities originally organized according to "material + geometry" are transformed into entities organized based on spatial positional relationships. In Figure 3, the diagram is sequentially divided into four sections from left to right in accordance with the depicted arrows, and the leftmost section represents the unaltered 3D model. The second part implies that the initial model typically relies on model materials to organize entity data, where all red windows in the diagram are designated as one entity. In the third part, the red material entity (windows) is divided into multiple sub-entities based on the spatial connectivity of the triangular surface, resulting in the acquisition of several independently operable windows. In the fourth section, we partition the entire building space into grids, creating multiple smaller blue spaces, and subsequently consolidate all sub-entities within each blue space into a single entity. In this process, the value H_{max} is used to impose an upper limit on the number of triangular surfaces generated after each object reorganization, thereby ensuring that excessive triangular surfaces are avoided for each individual object.

In the visualization system, the viewpoint space is the area that the observer (camera) may reach. But in the three-dimensional environment, a part of the space occupied by the physical entity is the inaccessible space, which is mainly occupied by the building objects and the tree objects. Therefore, the location characteristics and reachable region of viewpoint space should be considered comprehensively in the division of viewpoint space. The outer bounding box serves as an approximation of the physical entity's size. The following strategies are used in this paper to partition viewpoint space:

The outdoor viewpoint space is divided by first removing the outer bounding box of the physical entities within the viewpoint space, followed by calculating the average dimensions (length, width, and height) of all architectural entities' outer bounding boxes.

These average dimensions are then used as voxels to divide the viewpoint space. Finally, the entire scene space is partitioned based on voxel size. So, a voxel is a minimal representation of viewpoint space concretely in Formula (3), recorded as a VS voxel.

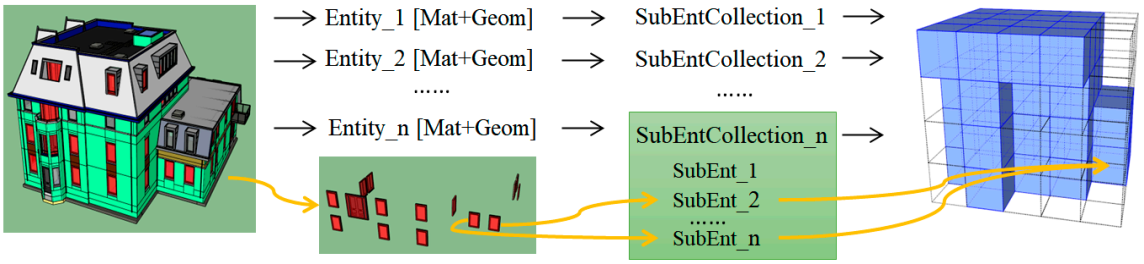


Figure 3. The process of restructuring an indoor entity.

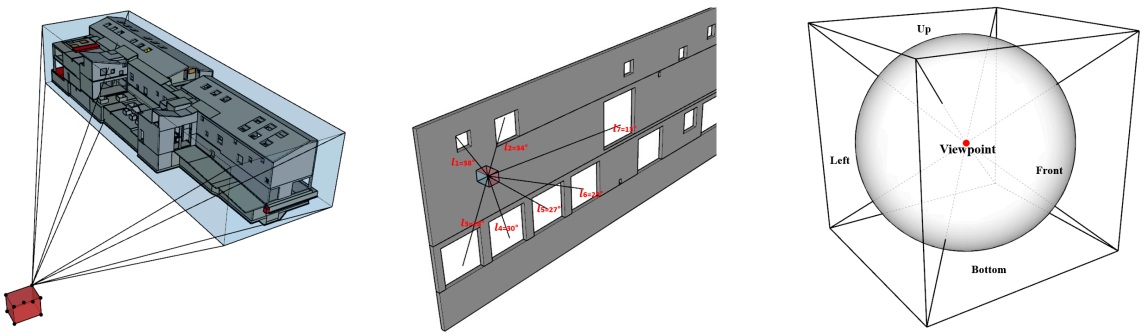
The indoor and in–outdoor viewpoint space refers to the observation place within and outside a building at close range. In visualization, the level of visual detail required should be determined by controllable sampling based on importance. To achieve this, we propose using a voxel cube with a side length denoted as l. The value of l is typically determined by factors such as the size of the building model ($Size_b$), the number of entities in the model (N), the maximum entity size ($Size_{max}$), and the average size across all entities ($Size_{ave}$) (Formula (5)). Here, *size* represents an average measurement encompassing length, width, and height.

$$l = \frac{N \times Size_{ave}}{Size_b} \times Size_{max} \tag{5}$$

3.3.2. Potential Visible Set Detection

We need to calculate the visible objects corresponding to each VS voxel by setting a series limited field of view. This process is the embodiment of the visual world in Formula (4). Entity acquisition in the visual world is called potentially visible set detection, while vision corresponds to the setting of the field of view. For each VS voxel in three scenarios of indoor, outdoor, and mixed scenes, we used visibility set detection methods based on building shells, openings, and omnidirectional views, respectively.

When the VS voxel is positioned outside, we use the shell detection method to obtain the visible set. We establish a fully connected line between the eight corners, the center of the VS voxel, and the minimum bounding box of the building shell, subsequently determining whether these lines intersect with any other objects. In the case that no obstructions are encountered, we consider that the chosen building shell is visible to the VS voxel. For example, as shown in Figure 4a, the corners of the building shell are connected to a VS voxel corner and are not obstructed.



(**a**) (**b**) (**c**)

Figure 4. PVS sampling strategies: (**a**) calculating visible building shells, (**b**) calculating visible wall elements for VS voxels and the other side, and (**c**) calculating visible objects in dense spaces with an omnidirectional view for one viewpoint.

Opening-based detection is primarily used for visualizing transitional areas both indoors and outdoors, where the line-of-sight traverses through an aperture such as a door or window. Within these transition spaces, the view extends beyond the building envelope via openings, encompassing diverse spatial regions.

In the opening object detection method, the initial selection of opening objects in the building shell includes doors and windows. Subsequently, a connection line is established between the center of the VS voxel and the center of each selected opening object. The angle formed between this connection and the wall containing the opening is then calculated. If the angle falls within the range of 15° to 90°, it is assumed that VS voxels can obtain visible objects through the opening space. For example, as shown in Figure 4b, the VS voxel's center is connected to the centers of seven windows, with angles between the connections and walls measuring $l_1 = 38°$, $l_2 = 34°$, $l_3 = 28°$, $l_4 = 30°$, $l_5 = 27°$, $l_6 = 23°$, and $l_7 = 13°$. It is hypothesized that the connection at angle l_7 may be imperceptible to the VS voxel.

The interior building space contains dense objects. In this paper, omnidirectional detection is used to collect the visible set of VS voxel. In the simulation of human vision in three-dimensional space, whether an object can be seen depends on human sight distance. The sight distance varies depending on the lighting conditions, with shorter distances observed in dim environments and longer distances in well-lit situations. Though omnidirectional visibility is calculated without considering the lighting conditions, the indoor sight distance is adopted using the diagonal length of the smallest building bounding box to ensure that no visible sets are lost.

The range of human visibility dictates that the field of human observation should be represented as a sphere, with the visual range serving as its radius. In omnidirectional detection, the field of view for a VS voxel in virtual space is simplified into six directions: upward, downward, leftward, rightward, forward, and backward. These six directions form a cuboid approximation of the tapered field of view (Figure 4c). So, the collection of visible objects in three-dimensional space can be transformed into a union of visible sets along these six directions.

To obtain a comprehensive collection of visible objects in a specific direction, we leverage the depth map cache within the graphics rendering pipeline. However, due to the inherent characteristics of this pipeline, transparent objects such as glass are erroneously treated as obstructive entities, leading to an omission of objects located behind them in the depth cache map. Consequently, this paper proposes a novel multi-frame recovery algorithm aimed at achieving precise extraction of the visible set. A depth cache map is generated in every frame, capturing the color and unique identity of each pixel as $ObjectID$. Thus, $DepthBuffer = \{(Pixel\_x, Pixel\_y, Color, ObjectID)\}$. By traversing the $DepthBuffer$ set and eliminating duplicate ObjectIDs, we obtain the $VisibleSet$, which represents the remaining visual collection. Building upon this foundation, a multi-frame restoration process (Figure 5) is used to further extract visible transparent objects. The specific process involves:

Step 1: While traversing the $DepthBuffer$ set, eliminate duplicate object IDs and record the number of duplicate deletions as r, denoted as the $VisibleSet = \{(ObjectID, r)\}$. Remove objects with $r > p$ from $VisibleSet$, resulting in a valid collection $VisibleSet = \{(ObjectID)\}$.

Step 2: Label all transparent objects in the view space based on their material property values, denoted as the $TransSet$. Prior to occlusion culling, remove $TransSet$ from the view space and then take and record the depth cache map of the first frame as $DepthBuffer_0$, which includes all visible non-transparent objects.

Step 3: Restore the previously deleted $TransSet$; acquire and compare differences between pixel values in both $DepthBuffer_0$ and $DepthBuffer_i$ for a given frame. The difference value indicates a transparent object. Then, put all the results $VisbleTransObj = \{VisibleTransparentObjects\}$ in the collection of visual transparent objects while removing $\{VisbleTransObj\}$ from $TransSet$.

Step 4: Iterate through frames by taking subsequent depth cache maps (e.g., $DepthBuffer_i$) until it matches with initial depth cache map (i.e., when $DepthBuffer_0 = DepthBuffer_i$). The resulting set of visible objects is obtained by combining $VisbleTransSet$ with objects present in $DepthBuffer_0$.

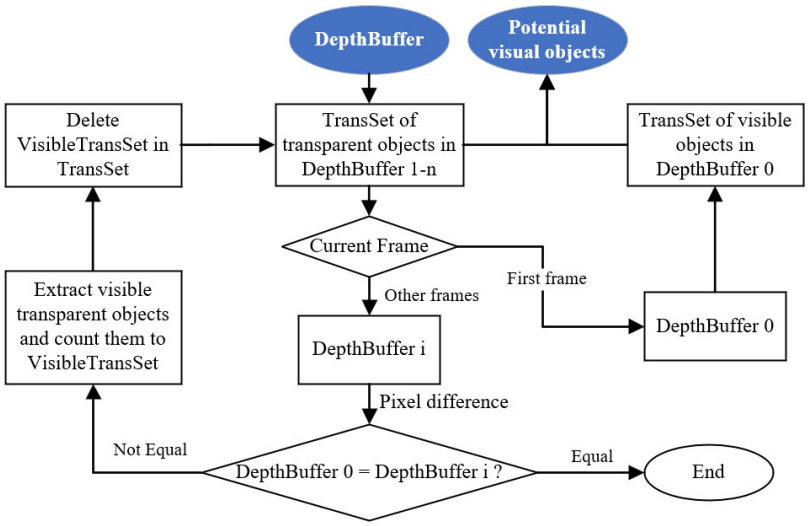


Figure 5. Multi-frame recovery to obtain visible objects.

3.3.3. R-Tree-Based Spatial Indexing

The computation of all potential visible sets was performed, establishing an association record with the VS voxel. However, during the process of visual scheduling, it is imperative for the system to swiftly determine the specific VS voxel occupied by the observer (camera) and promptly load the recorded visible objects within. In addition, a large number of VS voxels in a scene results in high retrieval costs, necessitating the construction of a spatial tree to enhance data scheduling efficiency. In this study, we use R-tree to establish a spatial index based on the following considerations: (1) R-tree possesses the capability to handle high-dimensional data, aligning with VESI's data structure characteristics and (2) R-tree can construct an index tree based on data proximity, where proximity is determined by whether two distinct VS voxels record similar or identical potentially visible sets.

R-tree utilizes minimum bounding boxes (MBBs) as the key for irregular geometries in order to construct a spatial index. In this study, we use a minimal set of visible objects (similar to MBBs at the data level) as the key for each VS voxel, facilitating the creation of a spatial index. Subsequently, we utilize a scenario diagram to visually depict the detailed structure of a potential visible set organized using an R-tree. Figure 6 illustrates a scene containing 12 objects labeled A-L, V1-V8 voxel grids, and outsourced rectangles that identify visible objects within each voxel space. For instance, Objects A, B, and C are considered visible objects within V1.

Figure 6a shows the visible objects recorded by each VS voxel. Though voxel V1 and voxel V2 are not adjacent in space, they share observable Objects A, B, and C. Since both have the same visible set, they will be placed under the same leaf node in the R-tree structure. Similarly, since Voxel V3 observes objects B, C, and D while sharing part observations with Voxel V1 and Voxel V2, the parent node for V1, V2, and V3 will be constructed.

The R-tree structure of the scenario is depicted in Figure 6b. The tree consists of four layers, with two child nodes, namely, R9 and R10, under the Root node. R9 contains records for R3, R4, and R8, while R10 records only R6 and R7. In the third layer of the tree, R3 stores two nodes: R1 and R2. Among them, both V1 and V2 are visible sets present in node R1; however, since they have identical visible sets, they can be recorded only once. By using an efficient recording process within the context of an R-tree data structure, VS voxels exhibiting similar visible sets exhibit shorter distances between their

corresponding nodes. Moreover, VS voxels with larger visible sets also benefit from reduced search levels and distances due to the arrangement provided by the tree structure. This optimization enhances performance for subsequent real-time scheduled data queries.

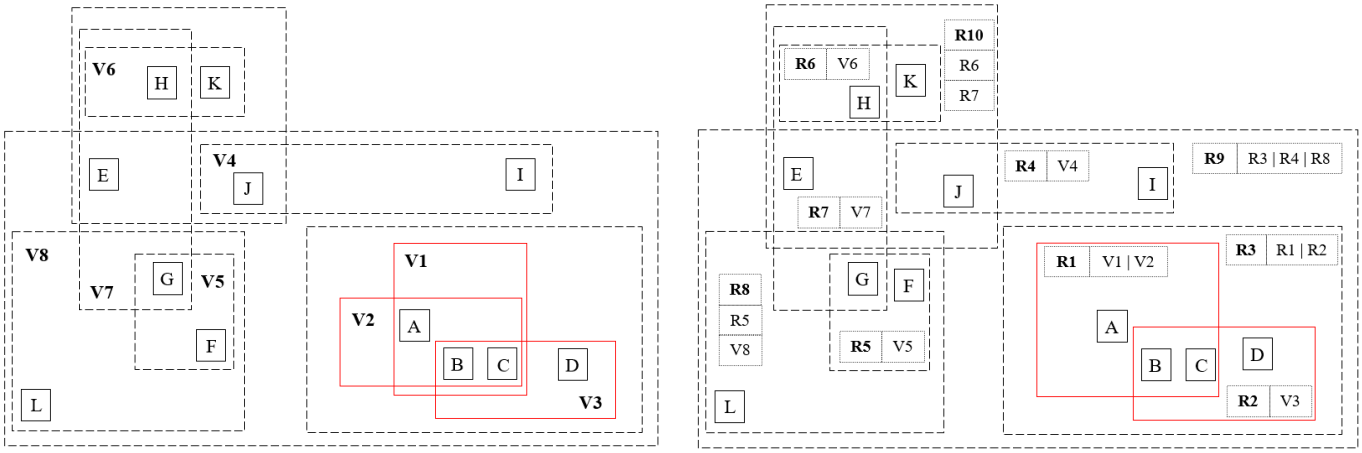


Figure 6. The organization of VESI: (**a**) the viewpoint space voxels that record the visible objects and (**b**) the R-tree for voxels and the associated visible objects.

3.3.4. Data Scheduling Using VESI

When data are scheduled, VESI is utilized to efficiently retrieve physically visible objects from the viewpoint location. The visualization system initially loads VESI files into memory and pre-constructs a spatial index tree. Upon entering a specific voxel, the system queries relevant visible objects through this tree structure, stores them in temp computer memory, and subsequently pushes them to the rendering pipeline for direct rendering.

The pseudo-code in Figure 7 formulates the scheduling process using VESI. Firstly, the system reads the VESI file to input the information in the 3D scene into memory. This information comprises a set of VS voxels and an R-tree pertaining to VS voxels. Then, the 3D scene walkthrough operation and data loading are performed. This includes determining the current position of the camera, setting the update interval, checking if a valid movement has occurred within the update interval, and calculating the set of new visible objects after a valid movement. Finally, the new visible object was rendered and visualized.

The *CurrentVoxel* is calculated by traversing all the recorded VS voxels and determining the x, y, and z values of *CurrentPosition* within the range of a specific VS voxel. So, a *CurrentVoxel* can be found. The initial rendering of the scene will consist of the collection of visible objects captured in the *CurrentVoxel*.

We utilize *ticktime* to facilitate update determination, wherein we calculate the *NextVoxel* based on the *CurrentPosition* prior to the expiration of *ticktime*. If *CurrentVoxel* and *NextVoxel* are identical, no data updates occur; otherwise, we compute the difference between them and visualize this disparity (called incremental mechanism).

The retrieval of R-trees is accomplished using a highly versatile approach. As illustrated in Figure 8, the pathfinding process from *CurrentVoxel* to *NextVoxel* involves a transition between two-layer parent nodes.

The incremental data scheduling mechanism we proposed can improve the efficiency of data rendering. While browsing in three-dimensional space, the viewpoint may traverse multiple voxel spaces or directly jump to a distant location (for example, when the movement speed significantly exceeds the update interval "*ticktime*"). In both scenarios, there exists a possibility of an intersection between visible objects recorded by different voxels, leading to the observation of the same object at different locations. The likelihood of such intersections increases with continuous transformations. Therefore, each time the

viewpoint is shifted to a new voxel, it becomes necessary to update the visible objects in the current location stored in memory. However, this update process uses an incremental mechanism that calculates and updates only the differences between the potentially visible set of the current and upcoming voxels with every change in viewpoint.

```
Pseudo-code: Data Scheduling Using VESI
1   Var RenderData, IncrementData, Rtree, CurrentPosition, CurrentVoxel, NextVoxel, tick-
    time.
2   For all voxels in the scene
3     If CurrentPosition in the voxel
4       CurrentVoxel init
5         RenderData = the pvs in the CurrentVoxel
6   Init ticktime = 0.05 s
7   For each ticktime
8     If CurrentPosition changed
9       For all voxels in the scene
10        find NextVoxel in the Rtree
11        NextVoxel init
12        If CurrentVoxel != NextVoxel
13        IncrementData = the pvs in the NextVoxel– RenderData
14  Using IncrementData to update the 3D scene.
```

Figure 7. The pseudo-code for "Data Scheduling Using VESI".

```
Pseudo-code: Finding NextVoxel start with the position of CurrentVoxel in the R-
tree.
1   Var Rtree, Currentvoxel, Nextvoxel
2   Get the Current-node of the Currentvoxel
3   For all children in the Current-node of the Rtree
4    If child = Nextvoxel
5     End
6    Else
7     Get the parent-node of the Current-node
8      For all children-p in the parent-node
9       For all children in the children-p
10       If child = Nextvoxel
11         End
```

Figure 8. The pseudo-code of "Find *Nextvoxel*".

Figure 9 presents a scheduling scenario, where Voxel5 and Voxel6 represent two arbitrary spatial locations. Figure 9a illustrates two VS voxel records of visible objects, wherein the visible objects recorded in Voxel5 include Ent1, Ent7, and Ent8, and those recorded in Voxel6 include Ent1, Ent2, Ent4, Ent7, and Ent8. Figure 9b demonstrates the utilization of an incremental mechanism for processing when transitioning from Voxel5 to Voxel6. Using the VESI retrieval method and comparing the differences in the visible sets between these voxels, Ent2 and Ent4 are added to computer memory while Ent8 is removed. With efficient incremental updates, newly visible objects can be promptly integrated into the rendering pipeline for executing visualization programs.

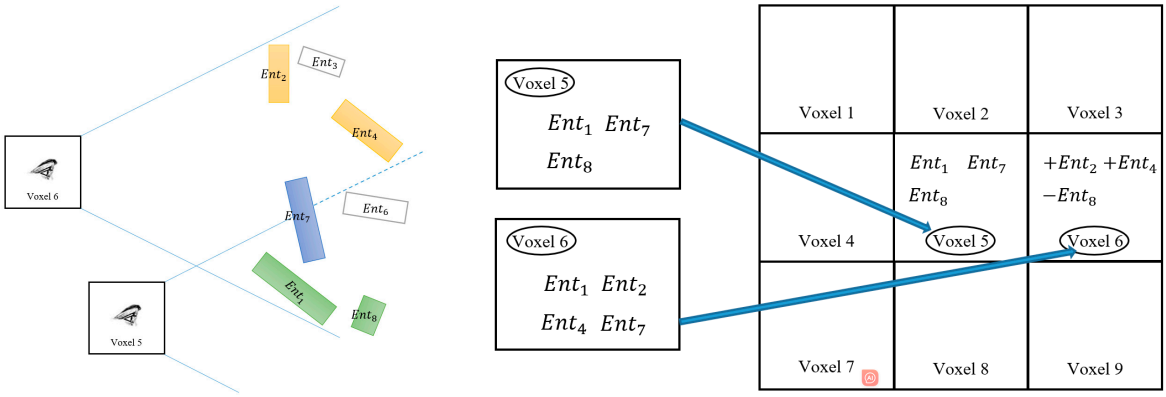


Figure 9. A scheduling sample: (**a**) two VS voxels that record the visible objects with intersect parts and (**b**) incremental objects when the camera moves from Voxel5 to Voxel6.

4. Empirical Exploration and Methodical Analysis

We conducted an experiment using a simulation scene that incorporated three architectural details, both indoor and outdoor spaces. Scene 1-1 accurately replicated the physical environment corresponding to the model, while the implementation of VESI effectively depicted the visualization process during transitions between the indoor and outdoor spaces. The photograph in Figure 10a depicts the current state, where Building 1 in the scene (Figure 10b) will serve as the continuous visualization for both the indoor and outdoor settings during this experiment. We utilized the C++ programming language in conjunction with the fundamental library provided by Unreal Engine 4 to develop a VESI production program and an accompanying visualization platform. The experimental software and hardware environment encompassed Windows 10, an Intel i7 CPU, and 16 GB of memory.

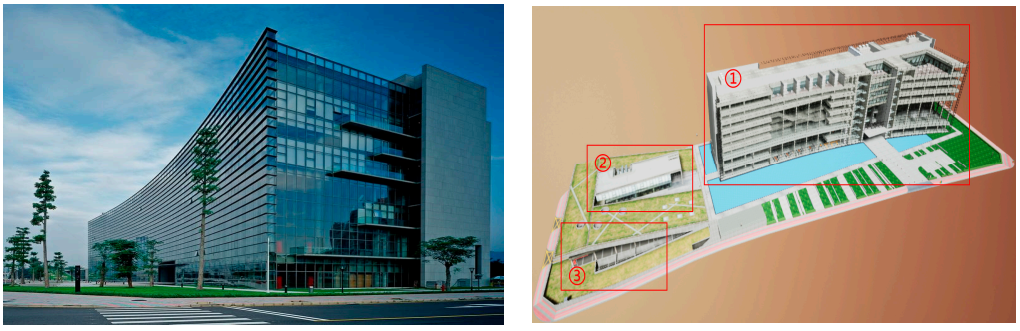


Figure 10. Continuous indoor and outdoor scenes: (**a**) a physical scene as a photo and (**b**) a digital scene as 3D models.

4.1. The Process of Creating a VESI

The original scene structure comprises four parts, each part is a single object with a large data size including three buildings and the building site, with several trees arranged on the site to simulate a realistic environment.

According to the VESI construction method, we extracted the shells of three buildings separately and reorganized their indoor data. The data on the three buildings were organized separately to generate sub-objects with distinct materials. To ensure a balanced granularity of object data, the number of triangular surfaces contained in a single object is limited during the generation process. The results of this reorganization are presented in Table 1. Building 1 exhibits a large shape and complex internal structure, thus adopting a

threshold value of 3000 to restrict the number of triangular surfaces per object. After reorganization, it yields 1409 new sub-objects with an average surface count of 1388. The simplest object consists of only 18 triangular surfaces. On the other hand, Buildings 2 and 3 possess relatively simpler structures and undergo data reorganization using a threshold value set at 1000; resulting in, respectively, generating 37 and 18 new sub-objects that all use a single material for rendering. The restructured data has become controllable metrological objects with fine grain, which meets the need for stable visualization using VESI.

Table 1. The reorganization result of the entire scene data.

Indoor Structure	Reorganization Setting		Post-Reorganization			
	Total Tri(s)	Threshold	Objects	Tri(s) Ave.	Tri(s) Min.	Tri(s) Max.
Building 1	1,956,156	3000	1409	1388	18	2680
Building 2	25,274	1000	37	683	24	912
Building 3	10,148	1000	18	558	24	631

In the division of viewpoint space, the coarse-grained voxel size ($8 \times 4 \times 3$) for outdoor space is determined by averaging the values of surrounding boxes outside the building. The height value of the VS voxel is 3, and it is chosen based on an estimated general building layer height of approximately 3 m. After excluding rigid objects (taking up inaccessible space), the remaining space is divided into a total of 1262 outdoor VS voxels using this size. The interior and connecting spaces are divided into Building 1's VS voxels with a finer size of $2 \times 1 \times 1$, resulting in 4408 units. Building 2 has 204 VS voxels, while Building 3 has 126.

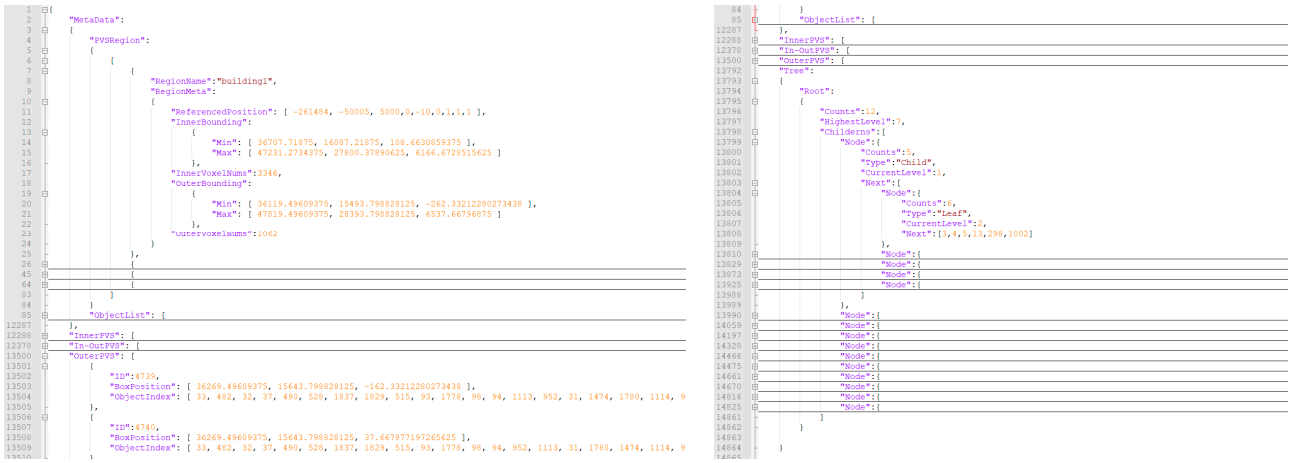
Subsequently, we calculate the visible set for all VS voxels and record the ID of each visible object. Table 2 presents the results obtained from dividing viewpoint spaces into voxels and calculating visible sets. In particular, as exterior walls primarily observed in outdoor spaces are merged into one object, there are fewer visible objects in these areas compared with others. On average, each VS voxel records five visible objects. Due to its complex internal structure, Building 1's corresponding VS voxel captures more visible objects with a maximum count reaching up to 52, whereas Buildings 2 and 3 exhibit relatively lower counts, averaging 10 and 9, respectively.

Table 2. The results obtained by dividing viewpoint spaces into voxels.

Viewpoint Space	Voxel Size	VS Voxel Num.	Visible Objects Per Voxel		
			Ave.	Min.	Max.
Outdoor	$8 \times 4 \times 3$	1262	5	2	16
building 1	$2 \times 2 \times 1$	4408	25	6	52
building 2	$2 \times 2 \times 1$	204	10	7	22
building 3	$2 \times 2 \times 1$	126	9	7	18

The VESI information is documented in Json format. Figure 9 illustrates the organizational structure and content of the VESI file for this particular scenario. In Figure 11a, *MetaData* captures PVS meta information for four regions. For the building area, *ReferencedPosition* specifies the world coordinate position of the building, *InnerBounding* represents the bounding box of its interior space, and *OuterBounding* denotes the bounding box encompassing both interior and transition spaces. *ObjectList* provides a comprehensive record of all reorganized objects within the scene. The file uses *InnerPVS*, *In-OutPVS*, and *OuterPVS* to document VS voxel information across the three kinds of distinct spaces, including VS voxel ID, location coordinates, and associated visible objects (identified by *ObjectIndex*). Figure 11b showcases the hierarchical tree structure of VESI with seven layers of nodes, where each node's *Type* attribute indicates whether it is a leaf node or not. The *Next* property

of intermediate nodes facilitates traversal to obtain child nodes, while the leaf nodes' *Next* property records VS voxel IDs.



(**a**) (**b**)

Figure 11. The structure of the VESI file in Json format (show in Notepad++). (**a**) The list of visible sets in VESI. (**b**) The tree structure of VESI.

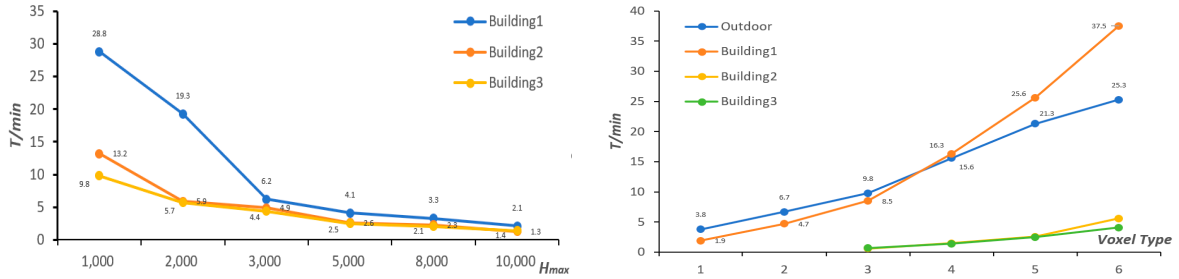
4.2. The Creation Efficiency of VESI

The efficiency issues in the preproduction of PVS and spatial index, which are part of data preprocessing, need to be addressed. Similarly, the calculation and measurement of the preprocessing efficiency of VESI as a combination of both programs is essential.

In comparison with the three-dimensional spatial index, VESI introduces an additional dimension of visual outcomes, thereby enhancing its efficiency with factors such as the scene size, sampling speed of the visual set, number of objects in the scene, and size of the scene viewpoint space (i.e., voxel count). Among these factors, the scene size remains objectively invariant, while the sampling speed relies on hardware conditions. Therefore, during VESI construction, we focus on achieving an optimal balance between efficiency by considering two parameters: the triangular surface threshold H_{max} for individual object data reorganization and VS voxel size (voxel type) for division within the viewpoint space.

The data preprocessing speed of the three buildings under different H_{max} values is depicted in Figure 12a. Overall, there is a decreasing trend in processing time as the threshold increases. Notably, for Building 1, setting the threshold to 3000 leads to a significant reduction in pretreatment time. Figure 12b demonstrates the efficiency of view space partitioning and sampling PVS across four regions with varying voxel sizes: Type1 ($8 \times 4 \times 3$), Type2 ($4 \times 4 \times 3$), Type3 ($3 \times 3 \times 3$), Type4 ($2 \times 2 \times 1$), Type5 ($2 \times 1 \times 1$), and Type6 ($1 \times 1 \times 1$). In terms of the outdoor space, the sampling speed of the viewpoint space division tends to stabilize as the grid size decreases. Conversely, for Building 1, there is a sharp increase in sampling time for the viewpoint space division as the grid size decreases. Building 2 and Building 3 exhibit similar simple internal structures, resulting in low processing times for both data preorganization and viewpoint space division sampling.

During the process of VESI creation, a smaller H_{max} value leads to the reconstruction of smaller and more balanced triangular surfaces, resulting in the formation of more discernible objects. Decreasing voxel size increases the number of VS voxels and consequently affects the extraction of visible sets. With each increment of one in voxel count, there is a corresponding increase by six in PVS calculations. Additionally, for indoor spaces with dense objects, voxel size significantly impacts efficiency due to the large number of visible objects involved in each PVS calculation. To achieve an optimal balance between efficiency and computational load, it is crucial to select threshold settings based on current test results, as outlined in Section 3.3.1.



(**a**) (**b**)

Figure 12. The preprocessing speed of VESI: (**a**) the cost limited by the threshold of the number of triangles (H_{max}) in a single object and (**b**) the R-tree for voxels and the associated visible objects.

4.3. The Visual Effects Analysis of VESI

We chose an "outdoor-indoor-outdoor" roaming path to test the implementation effect of VESI and compared the effect of using only the R-tree spatial index and PVS. Our main goal is to judge the fluency and stability of the visualization, which is estimated by the variation in the frame rate. All tests are executed after the preprocessed file is loaded.

4.3.1. Analysis of Roaming Fluency and Visualization Effects

On the roaming path, we displayed 20 relatively distinctive visual locations, and based on them, a statistical analysis was made in the follow-up. Figure 13 shows the scene browsing process along the entire roaming path. Among them, Figure 13a shows the VESI-based test, Figure 13b shows the spatial index-based test, and Figure 13c shows the PVS-based test. The starting point is situated adjacent to Building 2, encompassing an outdoor setting that primarily includes the building shell, the site, and surrounding trees (mainly pic. 1 to 3). Subsequently, access to the interior of Building 1 is attained through the indoor–outdoor transitional area (mainly pic. 4 to 7) of Building 1. Upon entry, an office corridor featuring a guest reception is traversed before reaching the hall (mainly pic. 8 to 14) within the main entrance vicinity. Finally, proceed through the primary entrance of Building 1 is traversed before re-enter the outdoor scene and observing Building 1 within its exterior context (mainly pic. 15 to 20).

(**a**)

Figure 13. *Cont.*

(**b**)

(**c**)

Figure 13. The visual effects test: (**a**) based on VESI, (**b**) based on the spatial index, and (**c**) based on PVS.

The three methods enable complete scene browsing from a free-roaming perspective. Among them, VESI and PVS exhibit higher fluency without noticeable stalling phenomena. However, in the spatial index-based roaming process, there are varying degrees of delays when transitioning between indoor and outdoor scenes. Furthermore, Figure 13b demonstrates that distant trees are not accurately calculated or loaded when directly visualized using the spatial index method at roaming points 1, 2, and 3. Similar issues occur with objects such as plants and tables/chairs at points 10 and 11. This is not observed with VESI and PVS.

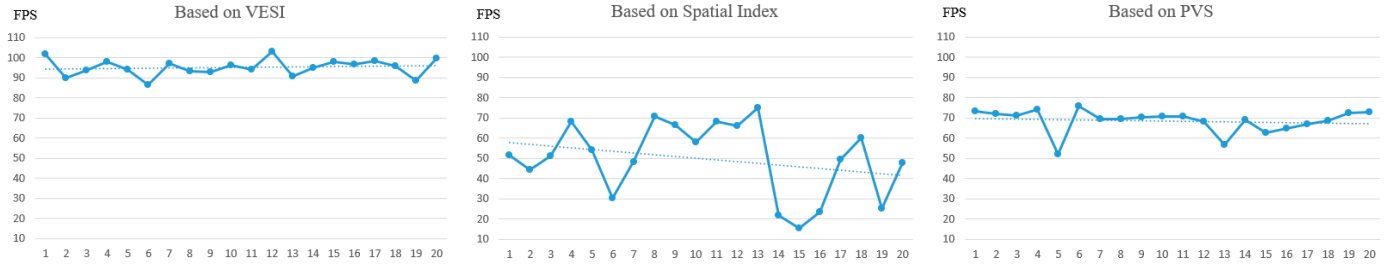
According to the definition of Formula (3), the actual calculated PVSs only provide an approximate estimation of the complete visible set calculation, which may result in potential omissions within the computed outcomes. However, during this experiment, the omissions of visible objects were not significant due to several factors: (1) the original data was reorganized based on "material + geometry", which could lead to spatially separated

parts being considered as a single object (e.g., chairs with identical materials); (2) different VS voxel partitions were used for various spatial regions; and (3) the outer envelope of the building was merged into one.

4.3.2. Visual Stability Analysis under Regional Changes

We conducted a comprehensive statistical analysis on the fluctuations in frame rate during the roaming process, while also contributing to the visual stability analysis.

As shown in Figure 14a, the VESI-based test yielded an average frame rate of 95.17, with the render frame rate at almost all positions along the path fluctuating around 95 frames, within a range of 10% (85–105 frames). The lowest frame rate observed was 86.37, occurring when transitioning from outdoor to indoor environments. Throughout the entire roaming process, the visualization effect remained consistently smooth due to its stable frame rate.



(**a**) (**b**) (**c**)

Figure 14. Visual stability statistics: (**a**) based on VESI, (**b**) based on the spatial index, and (**c**) based on PVS.

As shown in Figure 14b, the spatial index-based test yielded an average frame rate of 49.76, exhibiting two significant instances of frame rate jitter along the trajectory. Firstly, upon transitioning from outdoors to indoors (position 4 to position 8), the frame rate dropped from 70 to 30 and returned after that. Secondly, during the switch from an indoor scene to an outdoor scene (position 13 to position 17), the frame rate decreased from 75 to a minimum of 15.37. These fluctuations in frame rates within the indoor/outdoor transition area can be attributed to the sudden emergence of numerous new visible objects in this region, which necessitates substantial processing time for culling by the graphics rendering pipeline. In addition, the frame rate statistics are consistent with the actual roaming experience.

In the PVS-based tests (Figure 14c), the average frame rate was 68.53, indicating relatively smooth visualizations. However, the overall frame rate was lower compared with VESI-based roaming due to the time-consuming process of retrieving PVS collections. Notably, a significant frame rate jitter occurred along the roaming path during testing; specifically, when transitioning from outside into a room (position 4 to position 6), the frame rate dropped from 70 to its lowest point at 52.05 frames per second (fps). This decrease can be attributed to both slow retrieval speed and sudden changes in visible objects.

5. Conclusions

A realistic 3D visualization platform is gradually becoming an essential capability for smart city management, enabling cross-industry technology integration and collaborative contributions to its strength. With the two-way integration and driving force of 3DGIS technology and game engines (like Unreal Engine), methods to meet users' demands for browsing three-dimensional spatial scenes are continuously improving, as reflected in enhanced visual fidelity, smoother roaming processes, and further visual stability. Data preprocessing offers an effective approach to address these challenges. However, most previous methods have focused on solving the issues of large data loads and stable visualization effects from a single perspective, resulting in redundant data generation. To comprehensively

tackle these problems, this paper proposes the VESI approach, which builds a spatial index based on PVS collection. This method ensures a stable, reliable, and authentic 3D roaming experience without compromising the original data's visual effect.

The method proposed in this paper, however, still has significant potential for expansion. This is evident in the current testing scenario, which solely encompasses a confluence of multiple edifices and situations, akin to a miniature digital park. Nevertheless, further exploration is required to achieve stable visualization at the city level. Such endeavors may encounter challenges related to VESI scene stitching and higher-level data scheduling. So, the method of combining multiple sub-scenes into a large scene, including the re-computation of the PVS for adjacent spaces and the reorganization of the spatial index should be developed.

Furthermore, the current VESI solely captures physical components, disregarding real-time or near-real-time rendering elements in its calculation. However, it is crucial to investigate factors such as illumination (light, daylight, reflected light, and so on) that significantly impact human visual perception within the digital twin scene for future advancements in VESI. In terms of long-term development, the application exploration of VESI will also constitute a crucial undertaking. In particular, disaster visualization for smart city management is important as it can help realize an evacuation decision in the case of fire by integrating dynamic elements (pedestrians, cars, etc.) into an existing scene.

Author Contributions: Conceptualization, Chengpeng Li and Biao He; methodology Chengpeng Li and Xi Kuai; software, Chengpeng Li, Haojia Lin and Wei Zhu; validation, Chengpeng Li and Wei Zhu; formal analysis, Chengpeng Li; resources, Zhigang Zhao, Yu Liu and Renzhong Guo; writing—original draft preparation, Chengpeng Li; project administration, Biao He; funding acquisition, Biao He, Yu Liu and Xi Kuai. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, the Ministry of Land and Resources (KF-2022-07-007), the National Key Research and Development Program of China (2022YFC3800604), the Guangdong Basic and Applied Basic Research Foundation (2022A1515010117), the Shenzhen Science and Technology Program (20220810160944001), and the National Natural Science Foundation of China (41901325).

Data Availability Statement: The data presented in this study are available on request from the first author or corresponding author. The data are not publicly available due to the realistic location and appearance.

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

References

1. Kalogianni, E.; van Oosterom, P.; Dimopoulou, E.; Lemmen, C. 3D Land Administration: A Review and a Future Vision in the Context of the Spatial Development Lifecycle. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 107. [CrossRef]
2. Qi, C.; Zhou, H.; Yuan, L.; Li, P.; Qi, Y. Application of BIM+ GIS Technology in Smart City 3D Design System. In Proceedings of the International Conference on Cyber Security Intelligence and Analytics, Shanghai, China, 30–31 March 2023; Springer: Cham, Switzerland, 2023; pp. 37–45.
3. 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. [CrossRef]
4. Huo, Y.; Yang, A.; Jia, Q.; Chen, Y.; He, B.; Li, J. Efficient Visualization of Large-Scale Oblique Photogrammetry Models in Unreal Engine. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 643. [CrossRef]
5. 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]
6. Clark, J.H. Hierarchical Geometric Models for Visible Surface Algorithms. *Commun. ACM* **1976**, *19*, 547–554. [CrossRef]
7. El-Mekawy, M.; Östman, A.; Shahzad, K. Towards interoperating CityGML and IFC building models: A unified model based approach. In *Advances in 3D Geo-Information Sciences*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 73–93.
8. Gröger, G.; Plümer, L. CityGML—Interoperable semantic 3D city models. *ISPRS J. Photogramm. Remote Sens.* **2012**, *71*, 12–33. [CrossRef]
9. Zhu, Q.; Hu, M.Y. Semantics-based 3D dynamic hierarchical house property model. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 165–188. [CrossRef]
10. Biljecki, F.; Ledoux, H.; Stoter, J. An improved LOD specification for 3D building models. *Comput. Environ. Urban Syst.* **2016**, *59*, 25–37. [CrossRef]

11. Löwner, M.O.; Gröger, G.; Benner, J.; Biljecki, F.; Nagel, C. Proposal for a new LOD and multi-representation concept for CityGML. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf.* **2016**, *4*, 3–12. [CrossRef]
12. Tang, L.; Li, L.; Ying, S.; Lei, Y. A full level-of-detail specification for 3D building models combining indoor and outdoor scenes. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 419. [CrossRef]
13. Chen, Z.; Pouliot, J.; Hubert, F. A First Attempt to Define Level of Details Based on Decision-Making Tasks: Application to Underground Utility Network. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **2021**, *46*, 137–144. [CrossRef]
14. Tang, L.; Ying, S.; Li, L.; Biljecki, F.; Zhu, H.; Zhu, Y.; Yang, F.; Su, F. An application-driven LOD modeling paradigm for 3D building models. *ISPRS J. Photogramm. Remote Sens.* **2020**, *161*, 194–207. [CrossRef]
15. Zhu, Q.; Gong, J.; Zhang, Y. An efficient 3D R-tree spatial index method for virtual geographic environments. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 217–224. [CrossRef]
16. Zhang, L.Q.; Guo, Z.F.; Kang, Z.Z.; Zhang, L.X.; Zhang, X.M.; Yang, L. Web-based visualization of spatial objects in 3DGIS. *Sci. China Ser. F-Inf. Sci.* **2009**, *52*, 1588–1597. [CrossRef]
17. Yang, J.; Huang, X. A hybrid spatial index for massive point cloud data management and visualization. *Trans. GIS* **2014**, *18*, 97–108. [CrossRef]
18. Ke, S.; Gong, J.; Li, S.; Zhu, Q.; Liu, X.; Zhang, Y. A hybrid spatio-temporal data indexing method for trajectory databases. *Sensors* **2014**, *14*, 12990–13005. [CrossRef]
19. Han, J.; Na, C.-W.; Lee, D.; Lee, D.-H.; On, B.-W.; Lee, R.; Park, M.-W.; Lee, S.-H. An Unified Spatial Index and Visualization Method for the Trajectory and Grid Queries in Internet of Things. *J. Korea Soc. Comput. Inf.* **2019**, *24*, 83–95.
20. Yu, Y.; Zhu, H.; Yang, L.; Wang, C. Spatial indexing for effective visualization of vector-based electronic nautical chart. In Proceedings of the 2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration, Wuhan, China, 3–4 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 323–326.
21. Liu, Z.; Chen, L.; Yang, A.; Ma, M.; Cao, J. Hiindex: An efficient spatial index for rapid visualization of large-scale geographic vector data. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 647. [CrossRef]
22. Wu, H.; Zhu, Q.; Guo, Y.; Zheng, W.; Zhang, L.; Wang, Q.; Zhou, R.; Ding, Y.; Wang, W.; Pirasteh, S.; et al. Multi-level voxel representations for digital twin models of tunnel geological environment. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *112*, 102887. [CrossRef]
23. Andújar, C.; Saona-Vázquez, C.; Navazo, I.; Brunet, P. Integrating Occlusion Culling and Levels of Detail through Hardly-Visible Sets. In *Computer Graphics Forum*; Blackwell Publishers Ltd.: Oxford, UK; Boston, MA, USA, 2000; Volume 19, pp. 499–506.
24. Cohen-Or, D.; Fibich, G.; Halperin, D.; Zadicario, E. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. In *Computer Graphics Forum*; Blackwell Publishers Ltd.: Oxford, UK; Boston, MA, USA, 1998; Volume 17, pp. 243–253.
25. Masehian, E.; Amin-Naseri, M.R. A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *J. Robot. Syst.* **2004**, *21*, 275–300. [CrossRef]
26. Roden, T.; Parberry, I. Portholes and planes: Faster dynamic evaluation of potentially visible sets. *Comput. Entertain.* **2005**, *3*, 3. [CrossRef]
27. Li, B.; Wang, C.; Li, L. Efficient occlusion culling with occupancy proportion. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December 2008; pp. 1058–1061.
28. Bittner, J.; Mattausch, O.; Wonka, P.; Havran, V.; Wimmer, M. Adaptive global visibility sampling. *ACM Trans. Graph.* **2009**, *28*, 1–10. [CrossRef]
29. Wang, W.; Lv, Z.; Li, X.; Xu, W.; Zhang, B.; Zhu, Y.; Yan, Y. Spatial query based virtual reality GIS analysis platform. *Neurocomputing* **2018**, *274*, 88–98. [CrossRef]
30. Kim, J.; Lee, S. Potentially Visible Hidden-Volume Rendering for Multi-View Warping. *ACM Trans. Graph.* **2023**, *42*, 1–11. [CrossRef]
31. Voglreiter, P.; Kerbl, B.; Weinrauch, A.; Mueller, J.H.; Neff, T.; Steinberger, M.; Schmalstieg, D. Trim Regions for Online Computation of From-Region Potentially Visible Sets. *ACM Trans. Graph.* **2023**, *42*, 1–15. [CrossRef]
32. Hladky, J.; Seidel, H.P.; Steinberger, M. The camera offset space: Real-time potentially visible set computations for streaming rendering. *ACM Trans. Graph.* **2019**, *38*, 1–14. [CrossRef]
33. Kobrtek, J.; Milet, T.; Herout, A. Silhouette extraction for shadow volumes using potentially visible sets. *J. WSCG* **2018**, *26*, 9–16. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.