International Journal of Geo-Information

MDPI

*Article*

# A Parallel-Optimized Visualization Method for Large-Scale Multiple Video-Augmented Geographic Scenes on Cesium

**Qingxiang Chen [1,2], Jing Chen [3,*], Kaimin Sun [2,3], Minmin Huang [1], Guang Chen [2,4] and Hao Liu [2,4]**

[1] School of Geographic Information and Tourism, Chuzhou University, Chuzhou 239000, China; superchen@chzu.edu.cn (Q.C.); elfishhmm@chzu.edu.cn (M.H.)
[2] Technology Innovation Center for Spatio-Temporal Information and Equipment of Intelligent City, Ministry of Natural Resources, Chongqing 401120, China; sunkm@whu.edu.cn (K.S.); teddycg@hotmail.com (G.C.); cqkcyliuh@163.com (H.L.)
[3] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China
[4] Chongqing Academy of Surveying and Mapping, Chongqing 401121, China
* Correspondence: jchen@whu.edu.cn

**Abstract:** Surveillance video has emerged as a crucial data source for web Geographic Information Systems (GIS), playing a vital role in traffic management, facility monitoring, and anti-terrorism inspections. However, previous methods encountered significant challenges in achieving effective large-scale multi-video overlapping visualization and efficiency, particularly when organizing and visualizing large-scale video-augmented geographic scenes. Therefore, we propose a parallel-optimized visualization method specifically for large-scale multi-video augmented geographic scenes on Cesium. Firstly, our method employs an improved octree-based model for the unified management of large-scale overlapping videos. Then, we introduce a novel scheduling algorithm based on Cesium, which leverages a Web Graphics Library (WebGL) parallel-optimized and dynamic Level-of-Detail (LOD) strategy. This algorithm is designed to enhance the visualization effects and efficiency of large-scale video-integrated geographic scenes. Finally, we perform comparative experiments to demonstrate that our proposed method significantly optimizes the visualization of video overlapping areas and achieves a rendering efficiency increase of up to 95%. Our method can provide a solid technical foundation for large-scale surveillance video scene management and multi-video joint monitoring.

**Keywords:** visualization; WebGIS; VideoGIS; Cesium; WebGL

check for updates

**Citation:** Chen, Q.; Chen, J.; Sun, K.; Huang, M.; Chen, G.; Liu, H. A Parallel-Optimized Visualization Method for Large-Scale Multiple Video-Augmented Geographic Scenes on Cesium. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 463. https://doi.org/10.3390/ijgi13120463

Academic Editors: Wolfgang Kainz and Florian Hruby

Received: 6 November 2024
Revised: 14 December 2024
Accepted: 18 December 2024
Published: 20 December 2024

**Copyright:** © 2024 by the authors. Published by MDPI on behalf of the International Society for Photogrammetry and Remote Sensing. 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

Surveillance video has become ubiquitous in urban landscapes worldwide, offering low-cost, real-time capabilities and spatiotemporal data that are essential for modern Geographic Information Systems (GIS) [1]. By integrating video data with 3D geographic scenes, VideoGIS capitalizes on the cross-platform capabilities and data representation features of WebGIS, enhancing videos with contextual information. This approach aligns with human cognitive patterns, fostering an immersive visual experience [2–4]. Surveillance video-augmented geographic scenes have broad applications in critical areas such as disaster monitoring, traffic enforcement, and military operations [5–12]. Consequently, developing effective visualization methods for large-scale surveillance video-augmented geographic scenes on the Web is crucial for advancing the development and management of smart cities.

Recent studies on integrating surveillance video with 3D geographic scenes can be divided into two categories: GIS-augmented video methods [13,14] and video-augmented GIS methods [15,16]. As depicted in Figure 1, the GIS-augmented video methods display videos directly within the 3D GIS scenes or in billboard ways, with pointers indicating the camera's orientation [13,17]. However, these methods fail to dynamically integrate

the surveillance videos and geographic scenes, as they maintain separate positions and perspectives. In contrast, the video-augmented GIS methods use ShadowMap technology (see Section 2.1) to map video textures onto geographic scenes, aligning more closely with human cognitive habits.



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

**Figure 1.** (**a**) GIS-augmented video method; (**b**) Video-augmented GIS method.

The video-augmented GIS method encompasses two primary aspects: video registration and integration into 3D scenes [18]. (1) Regarding the video registration method, previous studies can be broadly classified into direct and indirect estimation techniques [13,18–21]. Direct methods involve determining the camera's position, orientation, and field of view directly from video frames to project textures onto the 3D GIS scenes. Conversely, indirect methods establish correspondences between specific point features in video frames and their corresponding 3D geographic locations [22–24]. (2) When integrating video into 3D scenes, prior methods primarily relied on ShadowMap technology for organizing and scheduling multiple video fusion scenes [25]. For instance, early research proposed a video-augmented GIS visualization approach using ShadowMap, but this method involved loading each camera directly into 3DGIS, thereby limiting large-scale visualization capabilities. Consequently, it was recommended to process up to five surveillance videos to ensure high rendering efficiency [2]. Li et al. (2021) employed an R-tree to segment the surveillance video scene and modeled the topological relationships between scene objects and cameras. This enabled the rapid selection of up to 10 cameras and significantly improved the visualization efficiency [26]. However, due to the lack of hierarchical organization and asynchronous scheduling, this method fell short of achieving progressive visualization for large-scale surveillance video scenes. In contrast, Fan (2019) leveraged multi-threading and quadtree organization on the WorldWind platform to divide the surveillance video scene. By pre-building a multi-level Level-of-Detail (LOD) model for video files using the texture downsampling technology, this approach enabled progressive visualization of large-scale video fusion scenes through an out-of-core rendering mechanism [27].

However, integrating video into 3D scenes poses significant challenges, particularly when organizing and scheduling large-scale scenes. These challenges become evident in several key areas. Firstly, traditional hierarchical and block-based methods may not always produce satisfactory results, especially in real-time video integration scenarios. For instance, Fang (2019) pre-constructed multiple Level-of-Detail (LOD) models with varying texture resolutions by downsampling non-real-time video files for subsequent scene scheduling [27]. However, this method is impractical for real-time videos. Furthermore, current organization methods index videos separately into different nodes, leading to inefficiencies when unifying the videos with overlapping areas for scene scheduling. Secondly, previous scheduling methods [26,27] employed independent single rendering pipelines for each camera, failing to fully exploit the parallel processing capabilities of the Graphic Processing Unit (GPU). This limitation can cause artifact problems, particularly when fusing and optimizing surveillance videos with overlapping areas independently.

As illustrated in the red area of Figure 2, when the same area is captured by two cameras, real-time video latency and projection errors can result in ghosting or artifact issues in the overlapping area. Therefore, it is imperative to unify the organization and management of videos with overlapping areas by placing them into the same node in large-scale video fusion scenarios. Additionally, during the scheduling algorithm, videos within the same node should utilize the same rendering pipeline to optimize processing, thereby mitigating ghosting issues and enhancing efficiency.



**Figure 2.** Artifact issues in overlapping regions of two surveillance videos.

Furthermore, to address the efficient transmission and large-scale visualization of geospatial data on the network in the context of WebGIS applications, current WebGIS platforms have implemented web map services, such as Web Map Tile Service (WTMS) and Tile Map Service (TMS) [25], as well as large-scale streaming standards, like 3D Tiles, Indexed 3D Scene Layer (I3S), or Spatial 3D Model (S3M) [28–30] for map data. A prime example is CesiumJS, an open-source framework based on the Web Graphics Library (WebGL) and JavaScript libraries, which has become the most popular choice for WebGIS development. Current research primarily focuses on utilizing Cesium's 3D Tiles streaming service, which simplifies original 3D models into multi-level LOD models, enabling progressive visualization of large 3D models through CesiumJS's rendering and scheduling mechanisms [31]. However, when it comes to integrating video with geographic scenes, current WebGIS platforms, including CesiumJS, Mapbox, and Three.js, lack a dynamic scheduling mechanism for integrating large-scale surveillance video into 3DGIS. This limitation significantly hinders the potential applications of large-scale multi-video data in various fields, such as urban management, investigative work, monitoring, and military inspection.

In conclusion, given the constraints of existing research and applications, we introduce a parallel-optimized visualization method for large-scale, multi-video augmented geographic scenes on the Cesium platform. Our proposed approach consists of two main parts:

(1) We propose an octree-based model for integrating large-scale multi-surveillance videos into 3DGIS. This model partitions extensive video data utilizing an octree structure, resulting in a block-based organization. Our approach distinguishes itself from other models by grouping videos, particularly those with overlapping areas, into the same node of the model, which corresponds to a unified rendering pipeline. This facilitates the subsequent integrated management of multiple videos containing overlapping regions. Within each block of video data, this paper employs a dynamic LOD strategy to establish a hierarchical structure.

(2) Building upon the proposed organization, we introduce an innovative Cesium-based scheduling algorithm for integrating large-scale multi-surveillance videos into 3DGIS. This algorithm efficiently leverages the Cesium scheduling mechanism and reimagines the three stages of the rendering process. Specifically, it facilitates model node traversal and visibility determination, video data acquisition and deletion, GPU-accelerated parallel

optimization, and LOD techniques. These features enable efficient video data management, leading to the progressive visualization of large-scale video-fused scenes.

This article is organized as follows: Section 2 illustrates the proposed method in detail; Section 3 illustrates the experimental results; Section 4 detailed discusses the work; and Section 5 concludes the method.

## 2. Materials and Methods

In this paper, we propose a parallel-optimized visualization method for large-scale multiple video-augmented geographic scenes on Cesium, which can be illustrated in three parts (Figure 3):

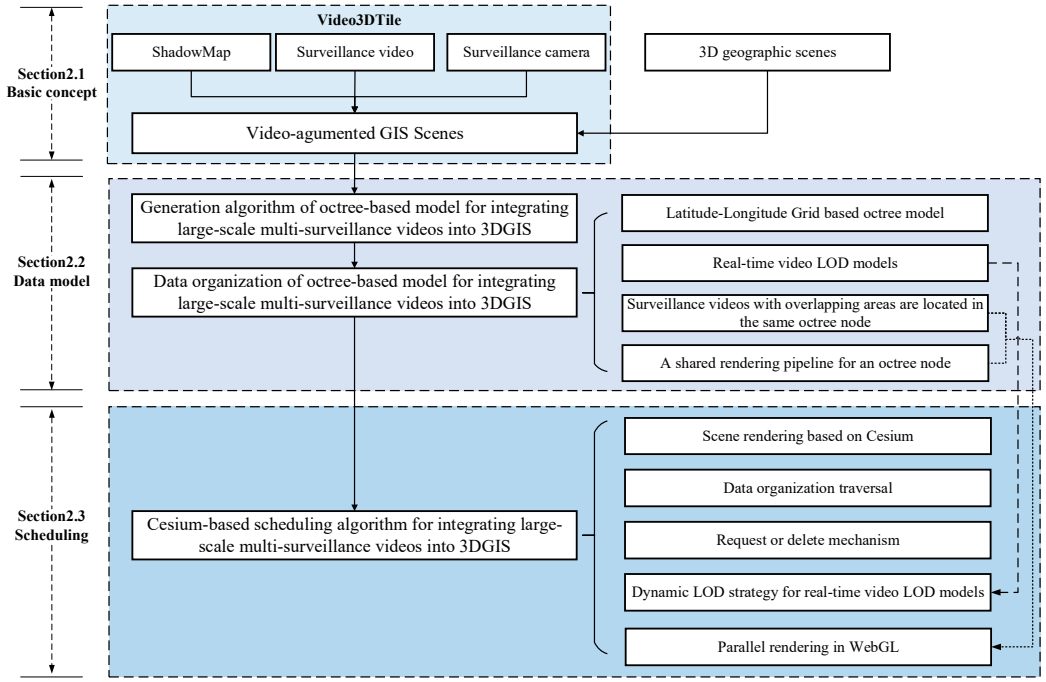



**Figure 3.** Workflow of the proposed method.

(1) Firstly, this paper introduces video-augmented GIS method based on ShadowMap and defines *Video3DTile*. It also presents a multi-video parallel optimization strategy from a mathematical perspective.

(2) Secondly, the large-scale video-fused geographic scenes are organized using an octree model formed by expanding a three-dimensional grid that is discretized based on the Latitude–Longitude Grid (LLG). A real-time LOD model is defined for each *Video3DTile*, which optimizes rendering performance by dynamically adjusting the level of detail based on the camera's distance and viewing angle. *VideoTiles* with overlapping areas are placed in the same node, and the remaining nodes are sequentially inserted into the remaining free nodes. Each node shares a rendering pipeline, enabling subsequent unified optimization and scheduling.

(3) Finally, the proposed algorithm traverses the organizational model, obtaining the LOD models of different *Video3DTiles* based on the Cesium scene rendering mechanism. It eliminates those *Video3DTiles* that are expired or invisible based on their spatial and temporal attributes and utilizes GPU parallel optimization to render current video-fused geographic scenes.

### *2.1. Basic Concept*

Surveillance cameras in use today are mainly categorized into two types: fixed cameras and Pan-Tilt-Zoom (PTZ) cameras. Fixed cameras provide a constant, unchanging field of view. In contrast, PTZ cameras, although confined to a spherical area with a fixed

center point, are capable of dynamic surveillance and tracking targets by adjusting various parameters, including focal length, pan angle, and tilt angle [32]. Figure 4 illustrates the process of the video-augmented GIS method, which primarily involves creating a model of the surveillance camera's visible range, known as the view frustum, and merging it with video feeds using the ShadowMap algorithm. This technique allows for the seamless incorporation of surveillance video into 3D GIS environments.

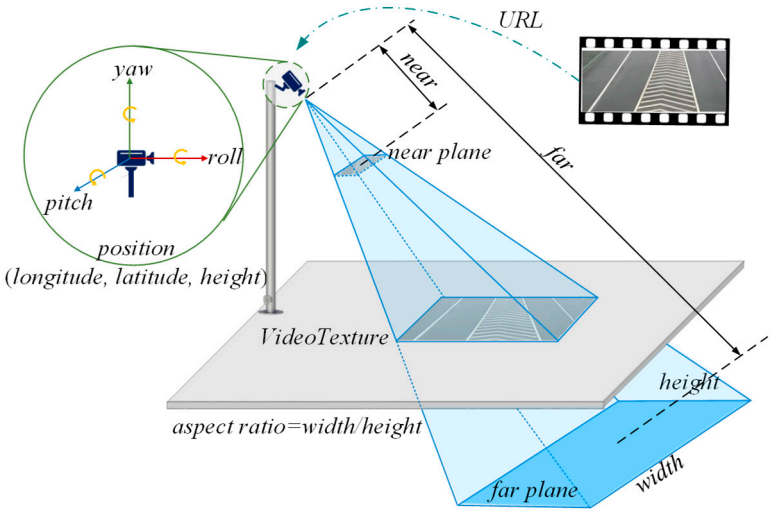



**Figure 4.** Video-augmented GIS method based on ShadowMap.

Specifically, ShadowMap technology, first proposed by Lance Williams in 1978, is a widely utilized technique for generating real-time shadows in 3D computer graphics. Video-augmented GIS method based on ShadowMap involves three primary steps:

(1) First, based on the surveillance camera's parameters, the pose and range are set, enabling the acquisition of the camera's depth texture. This texture represents the depth or distance values from the camera to various points in the scene, thereby forming a depth map. The pixel values within this depth map signify the depths of the fragments in the scene that are nearest to the light source;

(2) Then, the scene is rendered from the current camera's perspective to produce another depth map. By comparing the depth values, we can determine which fragments in the scene are currently 'visible' from the camera's viewpoint;

(3) Finally, for each 'visible' fragment, an additional rendering is conducted using a masked projection of the current video frame texture. This process achieves the desired effect of video fusion with the geographical scene. Typically, a semi-transparent mask is employed during this step to allow for finer control over the projected video frame. For a detailed explanation of the algorithm's implementation, refer to the literature [2].

Without loss of generality, this paper defines the area covered by a single surveillance camera's view frustum and the integration of video into 3DGIS scenes as a "*Video3DTile*". Specifically, a *Video3DTile* is comprised of the surveillance camera's video stream, the camera's pose parameters, a shadow map, and the video texture obtained by video-augmented GIS method based on ShadowMap. So, the concept of a *Video3DTile* can be represented by Equation (1):

$$\begin{cases} Video3DTile = \{S, V, f(S, g(V), a)\} \\ S = \{P, M(P); P = P_i + P_e\} \\ V = \{url, html(url)\} \end{cases} \quad (1)$$

where $S$ represents the acquired shadow map; $V$ represents the video stream; $f(*)$ denotes the process of forming a video texture with spatial features from the video and depth map and overlaying it onto 3DGIS scenes based on the transparency value $a$ (where $a$ ranges from 0 to 1); $g(*)$ represents the extraction of video texture from the video stream. Specifically, the shadow map $S$ is generated using the ShadowMap technique $M(P)$ with

input camera parameters $P$. $P$ can be broken down into: Intrinsic parameters $P_i$, including the Field of View (FOV), near plane, far plane, and aspect ratio of the video; and Extrinsic parameters $P_e$, which primarily consist of the camera's geographical location (longitude, latitude, and height) and orientation (yaw, pitch, and roll). Besides, the video stream $V$ is obtained by parsing the URL and HTML code associated with the video's web address. It is worth noting that while many scholars have conducted in-depth research on camera distortion correction and video registration, these topics are beyond the scope of this paper and will not be discussed further here.

Figure 5a demonstrates that when multiple surveillance cameras cover the same area, with either full or partial scene overlap, they create clustered video tiles, denoted as *Cluster*(*Video3DTile*). Specifically, *Cluster*(*Video3DTile*) represents a collection of *VideoTiles* that have overlapping video areas. Furthermore, the video textures of any two *Video3DTiles* share an intersection, which can be depicted by Equation (2):

$$Cluster(Video3DTile) = \{Video3DTile_i | f(S_k, g(V_k), a_k) \cap f(S_l, g(V_l), a_l) \neq \varnothing,\ k \neq l\} \quad (2)$$

where $Video3DTile_i$ represents the $i$-th *Video3DTile* in the *Cluster*(*Video3DTile*) set; $f(S_k, g(V_k), a_k)$ and $f(S_l, g(V_l), a_l)$ represent the video textures of the $k$-th and $l$-th *Video3DTile* in the set, respectively; and $\cap$ denotes the mathematical intersection.

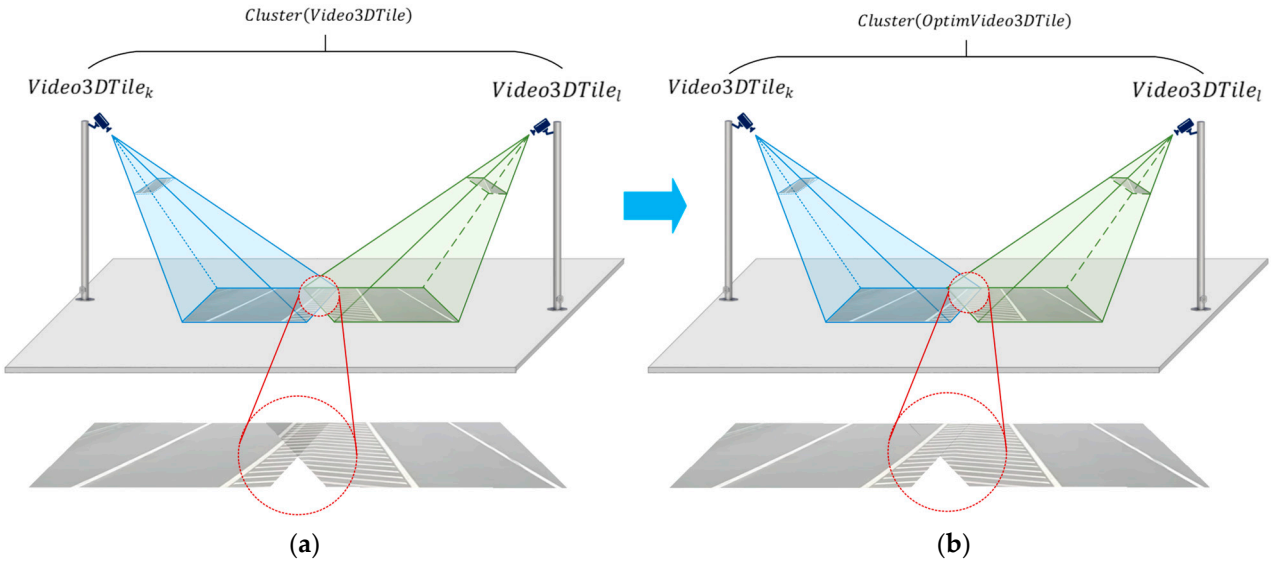



**Figure 5.** Integration from two surveillance videos to 3D geographic scenes ($a = 0.8$). (**a**) Before optimization; (**b**) After optimization.

Within a *Cluster*(*Video3DTile*), the integration of 3D geographic scenes from various cameras is handled separately. This approach, however, may result in visual artifacts due to texture inconsistencies in the overlapping regions of the videos, as indicated in Figure 5a. Thus, this paper concentrates on the integration of video scenes within Video3DTiles to enhance the rendering quality of video-augmented 3D geographic scenes. Specifically, the process mainly includes utilizing a single programmable rendering pipeline for managing overlapping video regions, leading to a texture stitching effect, as shown in Figure 5b.

Therefore, based on Equations (1) and (2), this paper defines the video textures obtained by optimizing the processing of *Video3DTiles* with overlapping regions using only one rendering pipeline as *Cluster*(*OptimVideo3DTile*). Here, $OptimVideo3DTile_i$ indicates that the video textures in $i$-th *Video3DTile* within the set differ from the *Video3DTile* itself and are processed concurrently through the same rendering pipeline to obtain an optimized result. These two concepts are represented by Equations (3) and (4), respectively:

$$Cluster(OptimVideo3DTile) = \{OptimVideo3DTile_i | f(S_k, g(V_k), a_k) \cap f(S_l, g(V_l), a_l) \neq \varnothing,\ k \neq l\} \quad (3)$$

where $OptimVideo3DTile_i$ represents the $i$-th optimized *Video3DTile* in the *Cluster*(*OptimVideo3DTile*); $f(S_k, g(V_k), a_k)$ and $f(S_l, g(V_l), a_l)$ represent the video textures of the $k$-th and $l$-th *Video3DTile* in the set, respectively; and $\cap$ denotes the mathematical intersection.

$$OptimVideo3DTile_i = \left\{ S_i, V_i, f\left(S_i, \bigcup (S_i, g(V_i), a_i)\right)\right\} \tag{4}$$

where $\bigcup(*)$ is used to represent the union of sets; the notation $f(\bigcup(*))$ represents the merging of *Video3DTiles* within the *Cluster*(*OptimVideo3DTile*), resulting in a unified video fusion scene. Obviously, if there is only one *Video3DTile* in the *Cluster*(*OptimVideo3DTile*), then *OptimVideo3DTile* = *Video3DTile*.

### 2.2. Octree-Based Model for Integrating Large-Scale Multi-Surveillance Videos into 3DGIS

To address the challenges of fusing large-scale multi-surveillance video scenes, we introduce an octree-based model designed to integrate these videos into 3DGIS. This model is primarily composed of two components: data organization and generation algorithm.

#### 2.2.1. Data Organization of Octree-Based Model for Integrating Large-Scale Multi-Surveillance Videos into 3DGIS

To advance large-scale scene scheduling in VideoGIS and its subsequent applications, it is essential to explore the organization of multiple, large-scale surveillance video scenes, as previously mentioned. In this paper, we assume the existence of a large-scale video-fused scene, named *Scene*(*Video3DTile*), within the 3DGIS environment. This scene includes a total of $n$ surveillance cameras. Among them, $m$ cameras cover $p$ overlapping areas, forming $p$ clusters named *Cluster*(*OptimVideo3DTile*). The remaining $n - m$ cameras, which do not overlap, each constituting a separate *Video3DTile*. This scenario can be represented by Equation (5):

$$\begin{cases} Scene(Video3DTile) = \left\{ Cluster(OptimVideo3DTile)_i,\ Video3DTile_j \middle| 0 \le i \le p-1,\ 0 \le j \le n-m-1 \right\} \\ m = \sum_0^{p-1} SUM(Cluster(OptimVideo3DTile)_i) \end{cases} \tag{5}$$

where $\sum$ represents summation; and $SUM(Cluster(OptimVideo3DTile)_i)$ denotes the total number of optimized *Video3DTile* contained within the $i$-th *Cluster*(*OptimVideo3DTile*).

Meanwhile, this paper considers the processing of $n - m$ non-overlapping *Video3DTiles* using the same programmable rendering pipeline as specified in Equation (5). However, it should be noted that the GPU can handle no more than t video textures in parallel [33]. Therefore, *Video3DTiles* are allocated to $\lceil (n-m)/t \rceil$ programmable rendering pipelines. Consequently, we optimize non-overlapping *Video3DTiles* from Equation (5) using the same rendering pipeline as specified in Equation (3) and define this as *NonCluster*(*OptimVideo3DTile*), which can be expressed using Equation (6).

$$NonCluster(OptimVideo3DTile) = \{ OptimVideo3DTile_i | 0 \le i \le \lceil (n - m/t) \rceil - 1 \} \tag{6}$$

where $OptimVideo3DTile_i$ represents $i$-th optimized *Video3DTile* in the *NonCluster*(*OptimVideo3DTile*).

Based on above equations, *Scene*(*Video3DTile*) consists of *Cluster*(*OptimVideo3DTile*) and *NonCluster*(*OptimVideo3DTile*), which can ultimately be represented by Equation (7):

$$\begin{cases} Scene(Video3DTile) = \{ Cluster(OptimVideo3DTile), NonCluster(OptimVideo3DTile) \} \\ n = \sum_0^{p-1} SUM(Cluster(OptimVideo3DTile)_i) + \sum SUM\left(NonCluster(OptimVideo3DTile)_j\right) \end{cases} \tag{7}$$

where $\sum$ represents summation; $SUM(Cluster(OptimVideo3DTile)_i)$ denotes the total number of optimized *Video3DTile* contained within the $i$-th *Cluster*(*OptimVideo3DTile*); and $SUM\left(NonCluster(OptimVideo3DTile)_j\right)$ represents the total number of optimized *Video3DTile* contained within the $j$-th *NonCluster*(*OptimVideo3DTile*).

In summary, considering the widespread distribution of *Scene*(*Video3DTile*), video fusion scenarios, and GPU's parallel processing capabilities, this paper extends the LLG [34] to an octree organization suitable for large-scale surveillance videos, as shown in Figure 6.

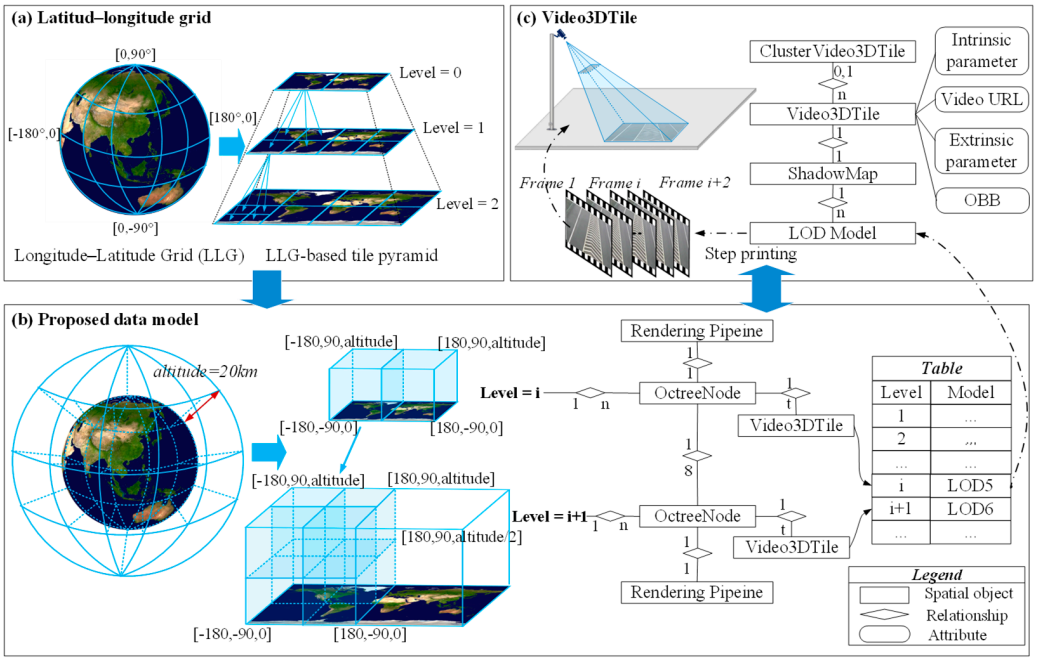



**Figure 6.** Proposed data organization.

As shown in Figure 6a,b, the proposed data organization method primarily refers to the 2D map division approach using the LLG. This method begins at level 0 by dividing the entire 2D Earth space into two subspaces of the same span based on latitude and longitude. Then, these subspaces are progressively subdivided according to the quadtree rule, resulting in a multi-level tile pyramid. Therefore, when extending this concept to 3D space in this paper, to ensure consistency in span, longitude, latitude, and height, the octree root node, OctreeRoot, is divided into subspaces with longitude and latitude spans of 180°, and the initial height is set to 20 km (i.e., half the circumference of the Earth). As illustrated in Figure 6b, according to the octree division characteristics, each octree node, OctreeNode, includes 0 or 8 sub-nodes. Unlike other organization methods, the OctreeNode in this paper defines one programmable rendering pipeline and stores up to a certain number of *Video3DTiles*, limited by the texture maximum number of WebGL [33]. This rendering pipeline is used for subsequent parallel processing of all *Video3DTiles* within the node. Combining the characteristics of WebGIS with the real-time nature of video streams, this paper implements a dynamic LOD strategy by adjusting the video frame sampling frequency. Among these models, the update frequencies for LOD1 to LOD6 are set to 1%, 5%, 10%, 20%, 50%, and 100%, respectively. Obviously, *Video3DTile* (*LOD* = 6) is equivalent to *Video3DTile*. It is important to note that despite having different video stream update frequencies, all LOD models represent the same *Video3DTile*.

### 2.2.2. Generation Algorithm of Octree-Based Model for Integrating Large-Scale Multi-Surveillance Videos into 3DGIS

Based on the aforementioned organization method, this paper proposes a construction algorithm for large-scale surveillance videos based on the octree model. This algorithm ensures unified scheduling and optimized processing of multiple video fusion scenes in the *Cluster*(*OptimVideo3DTile*), as described in Section 2.3. Additionally, it guarantees that Video3DTiles are inserted into available OctreeNode to facilitate subsequent parallel computations. Therefore, the construction process of this algorithm mainly includes the following three main steps:

- Step 1—Construct the scene root node:

Calculate the minimum bounding box for the set of *Video3DTile* in *Scene*(*Video3DTile*). Identify the maximum level of the octree node that contains the *Scene*(*Video3DTile*) according to the aforementioned octree division rules, and set this as the current scene root node (named SceneRoot).

- Step 2—Insert Cluster(OptimVideo3DTile):

Traverse all *Video3DTiles* in *Cluster*(*OptimVideo3DTile*), and insert them into SceneRoot in sequence. If the number of current $Cluster(OptimVideo3DTile)_i$ exceeds the available storage capacity of the current node, split the current node into eight octree sub-nodes (OctreeNode) according to the octree division rules. If a part of an OctreeNode contains *Video3DTiles*, insert them into that OctreeNode and repeat Step 2.

- Step 3—Insert remaining *Video3DTiles*:

Insert the remaining *n-m Video3DTiles* into the OctreeNodes in sequence, ultimately forming the organization method proposed in this paper. The specific formation process of this organization method is described as Algorithm 1:

**Algorithm 1.** Generation algorithm of octree-based model for integrating large-scale multi-surveillance videos into 3DGIS

**Input** $Cluster(OptimVideo3DTile)$, $NonCluster(OptimVideo3DTile)$.
**Step1** Construct SceneRoot: The process calculates the overall bounding volume for SceneRoot based on the aggregated bounding volumes of the smallest enclosing boxes formed by the *Video3DTiles*. The specific conversion formula is provided in reference [35].
**Step2** OctreeNode = SceneRoot.
**For each** $Cluster(OptimVideo3DTile)_i \in Cluster(OptimVideo3DTile)$
**Step3** **If** $BoundingVolume(Cluster(OptimVideo3DTile)_i) \notin BoundingVolume(OctreeNode)$ **then** continue.
**Step4** **For each** OctreeChild ∈ OctreeNode
**Step5** **If** $SUM(Cluster(OptimVideo3DTile)_i) + SUM(Video3DTile) \in OctreeChild \leq t$
$Cluster(OptimVideo3DTile)_i \rightarrow OctreeChild$ and continue.
**Step6** **If** $SUM(Video3DTile) \in OctreeNode \geq t$
OctreeNode splits 8 OctreeChild and goto Step3.
**Step7** $Cluster(OptimVideo3DTile)_i \rightarrow OctreeNode$ and continue.
**End For**
**For** $OptimVideo3DTile \in NonCluster(OptimVideo3DTile)$
**Step8** Execute Step2~Step7.
**End For**
**Output** SceneRoot.

### *2.3. Cesium-Based Scheduling Algorithm for Integrating Large-Scale Multi-Surveillance Videos into 3DGIS*

This paper proposes a scheduling algorithm for integrating large-scale video data from multiple surveillance sources into 3DGIS, utilizing the Cesium scene rendering mechanism and WebGL rendering pipeline. Additionally, based on video scene fusion (see Section 2.1) and the organization of surveillance videos (see Section 2.2), it achieves the integration and dynamic scheduling of these large-scale video data with 3D geographical scenes.

The core idea of this algorithm is to enhance the large-scale visualization of video-fused geographical scenes by integrating Cesium-based rendering strategies with out-of-core rendering techniques. Initially, the algorithm utilizes a Breadth First Search (BFS) traversal on the octree-based model (see Section 2.2) to identify the visible octree nodes within the current scene. From these nodes, it further selects the visible *Video3DTiles*. After these two rounds of filtering, a manageable number of *Video3DTiles* within the current scope are obtained. Next, the algorithm employs a request thread pool to parse each *Video3DTile*'s video data and construct shadow maps. Meanwhile, for the *Video3DTiles* that have been

selected and processed, based on a dynamic LOD strategy, different frame extraction frequencies are applied to obtain the video textures. These textures are rendered into geographical scenes using an improved video-augmented GIS method (see Section 2.1). That is, the algorithm employs GPU's parallel-optimized rendering for video textures within the same node to improve rendering efficiency. Finally, for the invisible *Video3DTiles*, a deletion thread pool is utilized to remove the shadow maps and parse the video stream modules, ensuring that the currently running *Video3DTiles* maintain low data volumes. Besides, this scheduling algorithm adopts threshold values for video processing based on the 3DTiles rendering strategy, with a default maximum threshold for the current number of videos [29]. These thresholds can be adjusted according to specific scenarios in practical applications to enhance the practicality of the scheduling algorithm presented in this paper. The algorithm implementation for each stage is described as follows (illustrated in Figure 7):

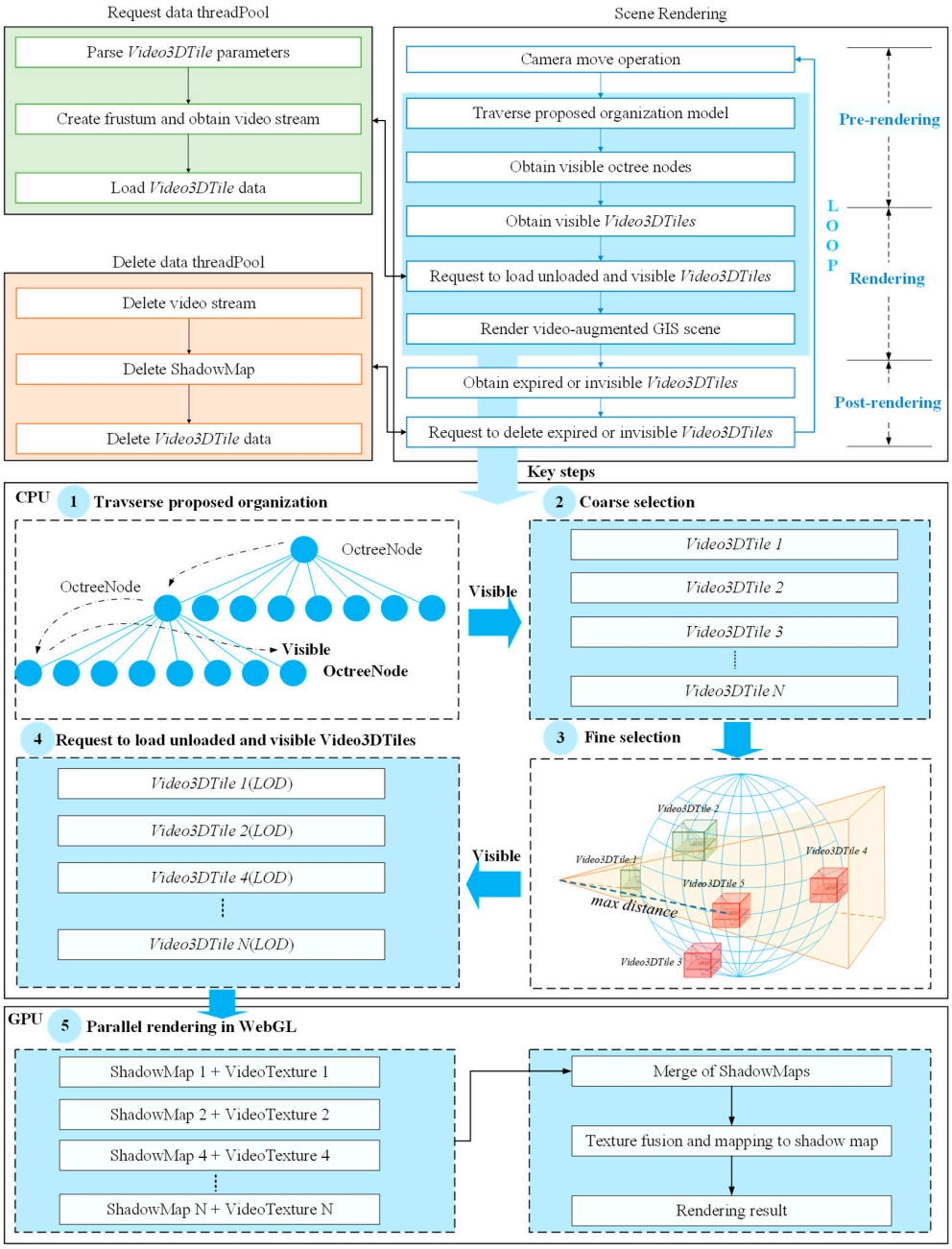



**Figure 7.** Flow chart for the proposed scheduling algorithm.

- Stage 1—Pre-rendering preparation

The algorithm employs a BFS algorithm on the proposed octree-based model to evaluate the visibility of each node relative to the current camera frustum. Initially, the visibility of octree nodes is determined. Further visibility checks are conducted on the *Video3DTile* set within the visible octree nodes. These checks involve assessing whether the bounding box of the node or *Video3DTile* is within the frustum or obscured by the Earth.

As illustrated in Figure 7(3), *Video3DTile1* and *Video3DTile2* are within the camera frustum of geographic scenes and are therefore visible. *Video3DTile*3 is outside the frustum and considered invisible. *Video3DTile4*, though within the frustum, is obscured by the Earth and thus invisible. *Video3DTile5*, while inside the frustum, is defined as invisible due to its distance from the camera's center exceeding a maximum threshold (through testing, this maximum invisible distance is defined as five times the minimum bounding sphere radius of the *Video3DTile*).

- Stage 2—Rendering process

For visible *Video3DTiles*, asynchronous requests are made to the video data service to render the video textures corresponding to the current camera's field of view. During this stage, the time cost of *Video3DTile* data requests, including parsing URLs, constructing frustum geometry, and building multiple shadow maps, is considered. Furthermore, if the number of currently rendered *Video3DTiles* exceeds the default maximum threshold for the current number of videos, no additional requests to load more *Video3DTiles* will be made.

As illustrated in the key steps of Figure 7, the algorithm quickly identifies visible *Video3DTiles* on the CPU side based on the results from Stage 1. Through the LOD strategy, the *Video3DTiles* within the same node are transferred to the GPU side in a single batch, reducing the number of GPU inputs to enhance efficiency. Moreover, on the GPU side, the shadow map regions of multiple *Video3DTiles* are merged through union operations (see Equation (3)). Furthermore, as shown in Figure 7(4), the algorithm requests the appropriate LOD model for rendering. The LOD model selection primarily considers the distance from the current camera viewpoint to the center of the *Video3DTile*'s coverage area, with closer distances resulting in the choice of higher LOD levels (see Section 2.2.1).

- Stage 3—Post-rendering process

In this stage, an asynchronous request mechanism is employed to asynchronously delete *Video3DTile* data for invisible or expired *Video3DTiles* by leveraging a thread pool dedicated to deleting *Video3DTiles*.

The specific process can be described as Algorithm 2:

**Algorithm 2.** Cesium-based scheduling algorithm for integrating large-scale multi-surveillance videos into 3DGIS

| | |
|---|---|
| **Input** | SceneRoot; MaxTileNum. |
| | **While** rendering is not end |
| | **Pre-rendering** |
| **Step1** | Traverse and obtain visible OctreeNodes from SceneRoot. |
| **Step2** | Visible *Video3DTiles* is obtained from OctreeNodes and added into the requestedTiles array. |
| **Step3** | Select the LOD model of $Video3DTile \in requestedTilesInFlight$. |
| | **Rendering** |
| **Step4** | Using a thread pool for asynchronous processing when the number of $requestedTiles \leq$ MaxTileNum, load $Video3DTile \in requestedTiles$ and $Video3DTile \rightarrow requestedTilesInFlight$. |
| **Step5** | Rendering visible $Video3DTile \in requestedTilesInFlight$. |
| **Step6** | Rendering 3D geographic scenes. |
| | **Post-rendering** |
| **Step7** | Remove unloading $Video3DTile \in requestedTilesInFlight$. |
| **Step8** | Using a thread pool for asynchronous processing, remove invisible or expired $Video3DTile \in requestedTilesInFlight$. |
| | **End While** |

## 3. Results

### 3.1. Experimental Environment and Data

To verify the effectiveness of the proposed method, the open-source 3DGIS platform Cesium is employed, and Table 1 details the specific parameters of the experimental environment.

**Table 1.** Detailed parameters of experimental environment.

| Environment | Name | Description |
|---|---|---|
| Software | Operation system | Windows 11 64 bits |
| | Graphical development interface | WebGL 2.0 |
| | Development platform [36–38] | CesiumJS 1.106<br>Microsoft Visual Sudio Code 1.95.1<br>Chrome Browser 131.0.6778.86 (64 bit) |
| Hardware | Central processing unit | 11th Gen Intel(R) Core(TM)i7-11800H@2.30 GHz CPU |
| | Graphic processing unit | NVIDIA GeForce GTX 3080 Laptop GPU |
| | Random access memory | 32 GB |

As shown in Figure 8, the experimental area is selected around the campus of Chuzhou University in Chuzhou City, Anhui Province, China. The data include Tianditu imagery and boundary data, an oblique photography model of Chuzhou University (approximately 10.9 GB), and surveillance video data (as shown in Figure 8b). Additionally, to verify the effectiveness of large-scale visualization using the method proposed in this paper, a simulated 3D scene containing up to 70,000 virtual surveillance video data points was constructed for further testing the method's validity (as shown in Figure 8c,d). The relevant demonstration can be found in Videos S1–S3.

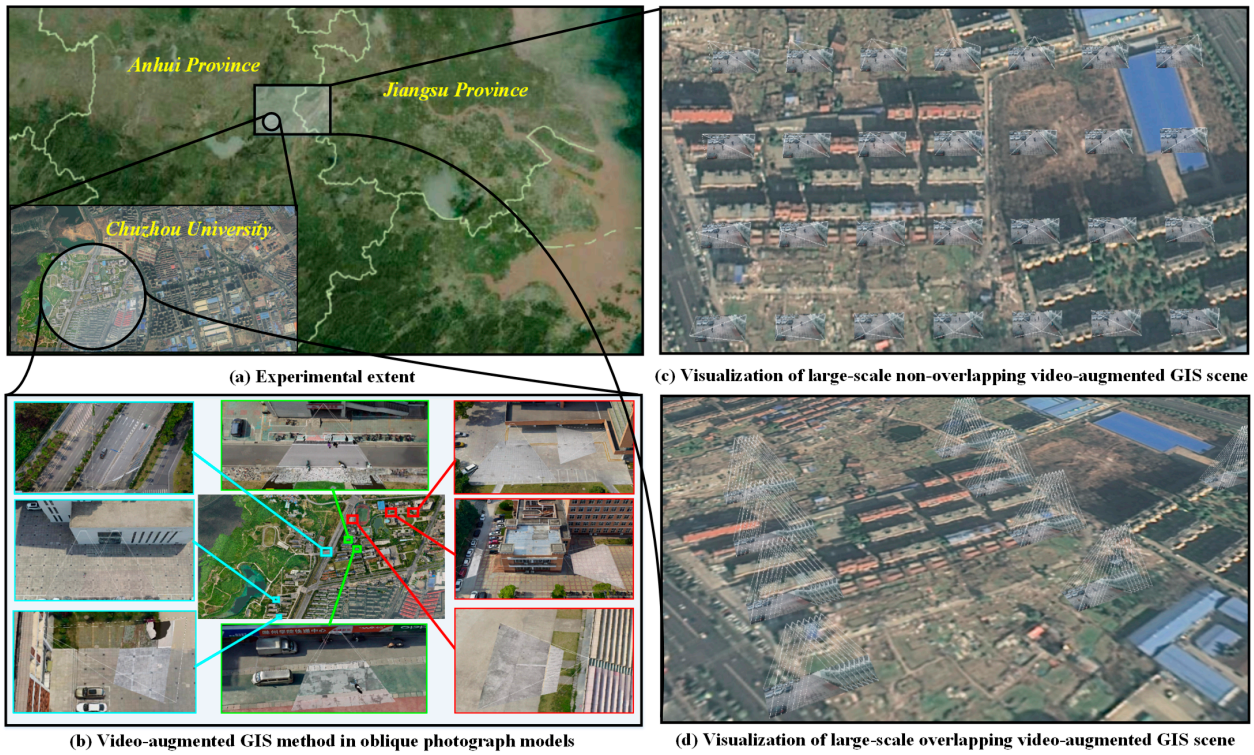



**Figure 8.** Experimental area.

### 3.2. Experimental Result and Analysis

The proposed method focuses more on the organization and scheduling of large-scale video fusion in geographical scenarios. Therefore, the experiments conducted in this paper mainly involve a comparative analysis of the scene organization and scheduling methods of existing approaches. The comparison includes two categories of methods from the relevant literature: Method 1 corresponds to the approach integrated directly into 3D scenes by Milosavljević et al. (2016) [2]; Method 2 corresponds to the approach organized and scheduled by Fan et al. (2019), which utilized the quadtree indexing mechanism [27]. However, Method 2 did not consider the real-time video processing capability, rendering the original LOD strategy ineffective. Therefore, this paper categorizes the approach into Method 2 and Method 2 with the proposed LOD strategy. The proposed LOD strategy typically uses distance-based rules for selecting LOD models. Specifically, the distance value is determined by multiplying the Euclidean distance between the center point of the current scene's camera and the center of the *Video3DTile*'s range. Table 2 shows the correlation between the selection of the LOD models and the multiple of this distance value.

**Table 2.** Detailed parameters of the dynamic LOD strategy.

| Level | Distance Multiplier |
|---|---|
| LOD 1 | 0.8 |
| LOD 2 | 1.7 |
| LOD 3 | 2.5 |
| LOD 4 | 3.3 |
| LOD 5 | 4.2 |
| LOD 6 | 5.0 |

#### 3.2.1. Analysis of Visualization Effects

To test the visualization effects of the proposed method, a comparison experiment was conducted between the proposed method and Methods 1 and 2. This experiment used the oblique photography model of Chuzhou University as the data foundation, with the Zhixing Bridge serving as the experimental area. Furthermore, video imagery for this experimental area was captured using two cameras (as shown in Figure 9). In the figure, the blue area indicates the region captured by a single camera, the green area represents the overlapping region captured by both cameras, and the red area denotes the non-visible region. This experiment compared the visualization effects of the three methods from far to near.

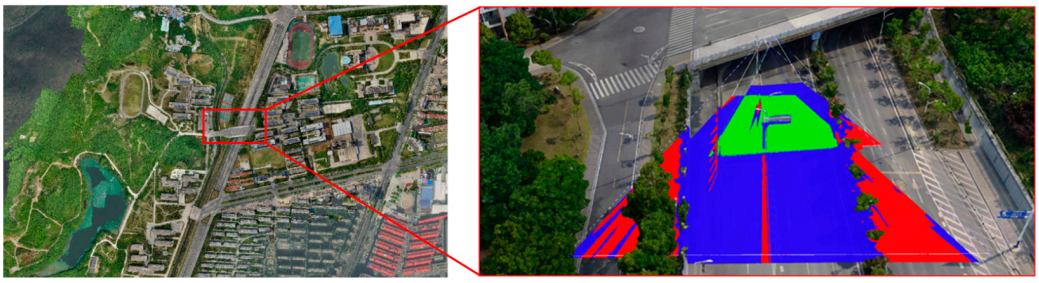


**Figure 9.** Experimental area around the Zhixing Bridge.

As shown in Figure 10, the experiment compared the visualization effects by varying the distances from the current camera position to the ground: 1160 m, 344 m, and 44 m. Both current methods (i.e., Method 1 and Method 2) and the proposed method were employed for comparative experiments, yielding their respective visualization results. It was noteworthy that Methods 1 and 2 produce consistent visualization effects; therefore, only the results of Method 1 were presented in Figure 10. Additionally, in the video-

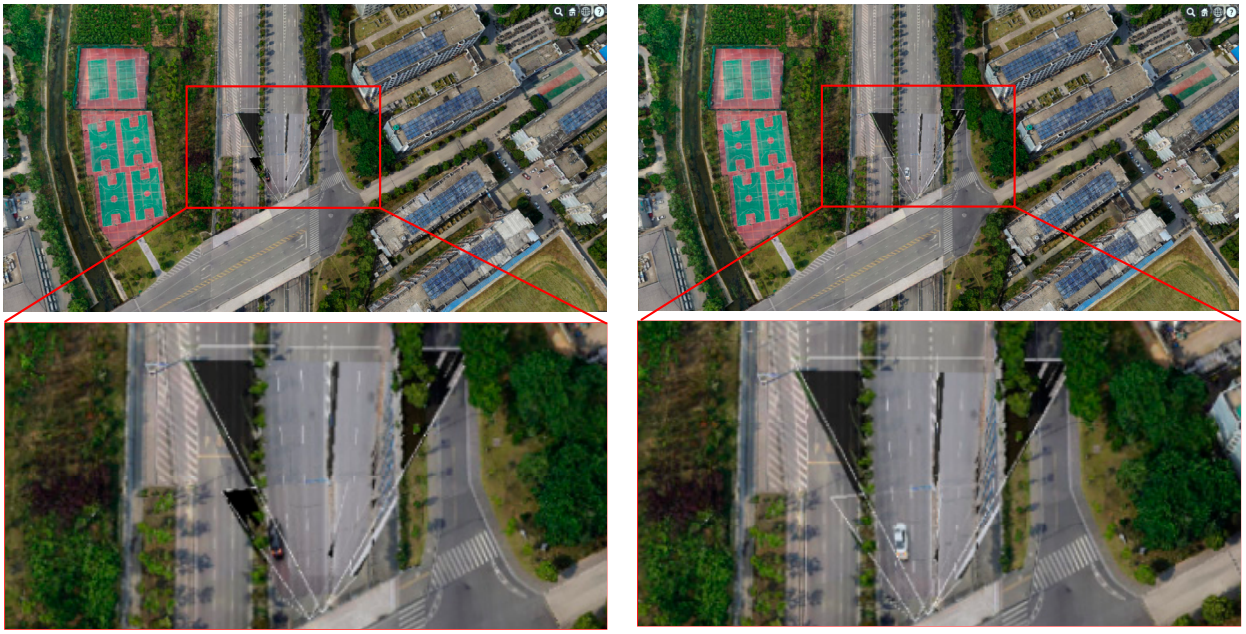
enhancement scenario, the transparency was set to $a = 0.8$ to highlight the differences between the compared methods (for detailed parameter explanations, see Equation (1)).

As indicated by the red-marked area in Figure 10a, when the distance from the current camera position to the ground was 1160 m, previous methods displayed video-fused scene information. However, computer displays have a fixed resolution. When viewed from a distance, the video-fused scene occupies only a small fraction of the screen's pixels, thus decreasing video clarity. Consequently, detailed information features become indistinct. Therefore, the proposed method, considering the inability to view corresponding feature information at excessive distances, achieved the same visualization effect by not displaying the video-fused scene information (see Section 2.2.2). Additionally, when the distance was 344 m, the method proposed in this paper and Method 1 visualized the same effect for video-augmented 3D scenes, as shown in the red-marked area of Figure 10b. At this relatively short distance, the scene covers a substantial portion of the screen's pixels, thereby enhancing video clarity. To ensure high-quality visualization, the proposed method employs a dynamic LOD strategy to appropriately display and update the video stream. Furthermore, as illustrated in Figure 10c, when the distance was less than 44 m, all methods displayed video-fused 3D scenes. However, since Method 1 or Method 2 operated independently for two surveillance videos and there was a certain distortion in the video-fused geographical scenes, the left red-marked area in Figure 10c demonstrated significantly mismatched lane lines, indicating a significant artifact issue. In contrast, this article adopted the highest frequency to update the video stream based on Table 2, achieving high-quality visualization effects. Meanwhile, the red-marked area in the right image adopted the proposed method, which merged and processed the textures of the two videos, thereby enhancing the visualization effect to a certain extent.

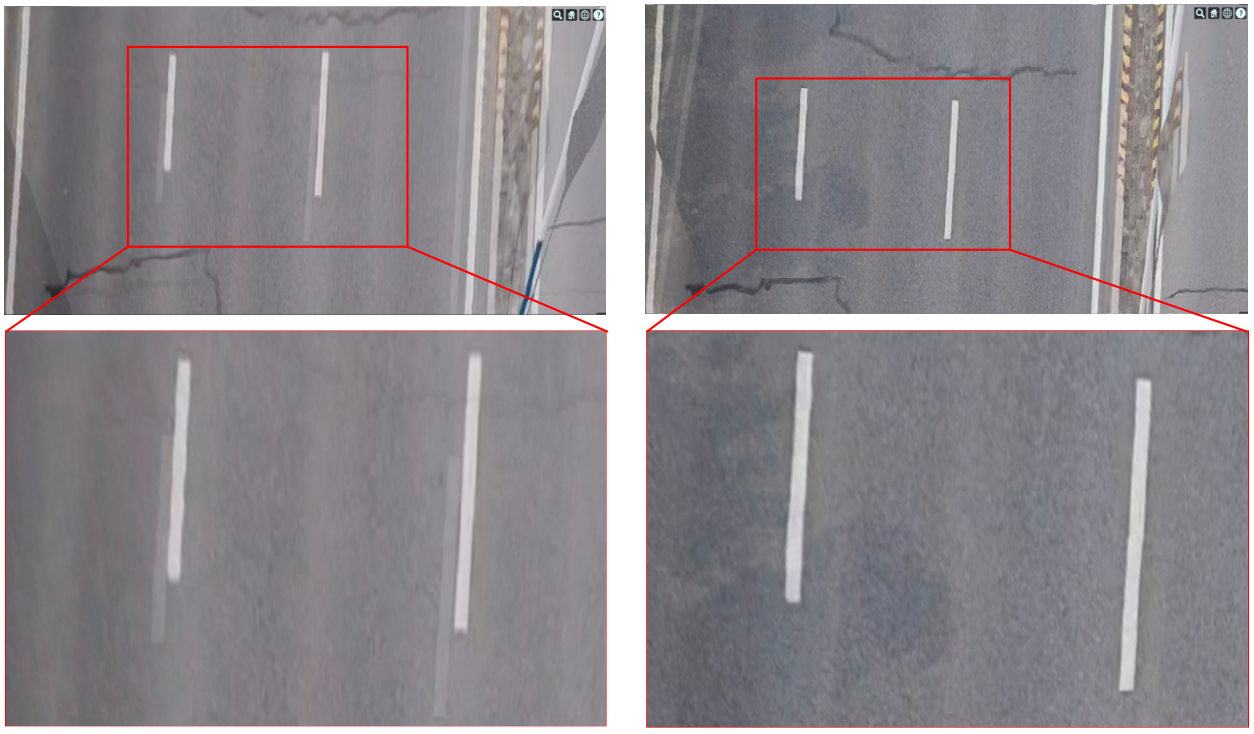


(**a**) Visualization results at 1160 m

**Figure 10.** *Cont.*



(**b**) Visualization results at 334 m



(**c**) Visualization results at 44 m

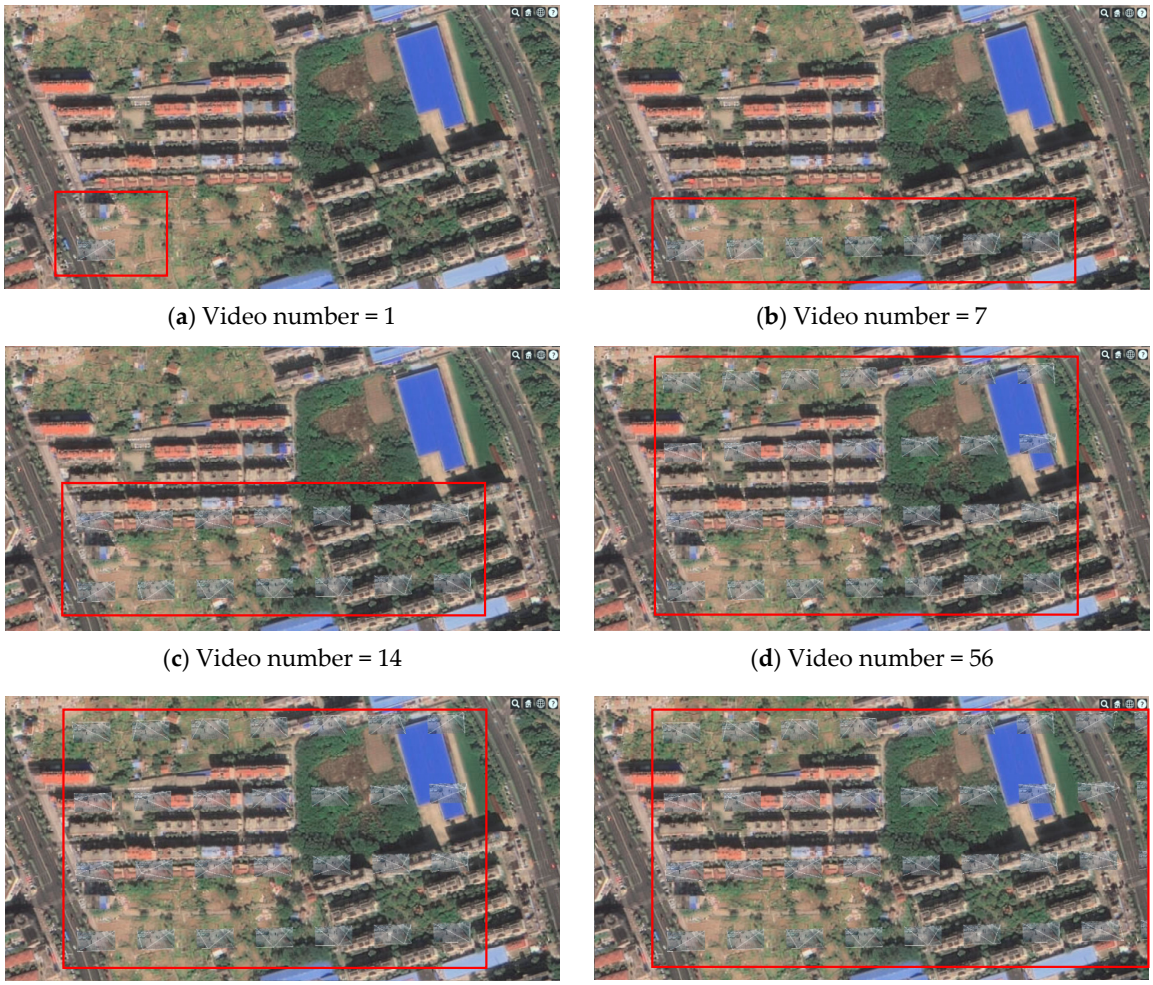
**Figure 10.** Visualization results from different distances (left: Method 1 and 2, right: proposed method).

#### 3.2.2. Analysis of the Visualization Performance

To evaluate the efficiency of the method introduced in this paper, we utilized the widely adopted Frames Per Second (FPS) metric to assess and compare the visualization rendering performance of alternative methods: Method 1, Method 2, and Method 2 combined with our proposed LOD strategy [16,18]. We virtually constructed non-overlapping surveillance scenes with equal intervals for the experiment. Specifically, each surveillance video covered a longitude and latitude span of 0.0007°, and we tested with varying numbers of videos:

1, 7, 14, 56, 70, 100, 700, 7000, and 70,000. The experiment recorded the frame rates and visualization results at 584 m and 126 m above the ground, respectively.

(1) When the scene camera was positioned 584 m above the ground, the experimental comparisons showed that all four methods produced consistent visualization results, primarily due to the absence of overlapping areas between videos. Without losing generality, this article illustrates the visualization results achieved using Method 2 (as shown in Figure 11). Among these methods, when the number of videos was 140 or fewer, they all yielded identical results. Specifically, with 56 or 70 videos, the displayed number remained consistent due to the limited scope of the scene, with some videos falling outside the visible range. However, when the number of videos reached 700 or more, Method 1 failed to render because it attempted to load all video data directly, exceeding computational capabilities and storage space.



(**a**) Video number = 1 (**b**) Video number = 7

(**c**) Video number = 14 (**d**) Video number = 56

(**e**) Video number = 70 (**f**) Video number ≥ 140

**Figure 11.** Visualization results for different video numbers at 584 m.

In contrast, the method proposed in this paper, as well as the other methods, employed strategies such as the out-of-core rendering mechanism to visualize the video-fused geographic scenes. Specifically, due to the excessive number of videos, only a portion could be displayed in the current scene. Therefore, when the number of videos was 140 or more, the visualization results of the proposed method, as well as those of the comparison methods, were consistent with those depicted in Figure 11f.

Figure 12 presents the rendering frame rates of the four methods for different numbers of videos, with the scene camera positioned 584 m above the ground. The recorded results indicate that the proposed method achieved a higher frame rate compared to the existing

methods. Specifically, Method 1 experienced a gradual decline in frame rate as the number of video frames increased. When the number reached 700, Method 1 crashed due to the browser's inability to handle the extensive video data load, exceeding its computational capacity and storage space. In contrast, Method 2 maintained a stable frame rate of approximately 12 FPS without considering the LOD strategy, thanks to the application of the out-of-core rendering technique. However, after incorporating the LOD strategy proposed in this paper, Method 2's frame rate increased to 22 FPS, which closely matched the frame rate maintained by our proposed method. Notably, the consistent frame rate between Method 2 with the dynamic LOD strategy and our method can be attributed to the LOD strategy's significant impact on reducing frame rate drops at greater distances from the ground. To further validate this, a comparative experiment was conducted at 126 m, where the highest-level LOD6 model was utilized for the video-augmented scene. The results of this experiment showed that both methods maintained a stable frame rate, confirming the effectiveness of the LOD strategy in improving the rendering performance.

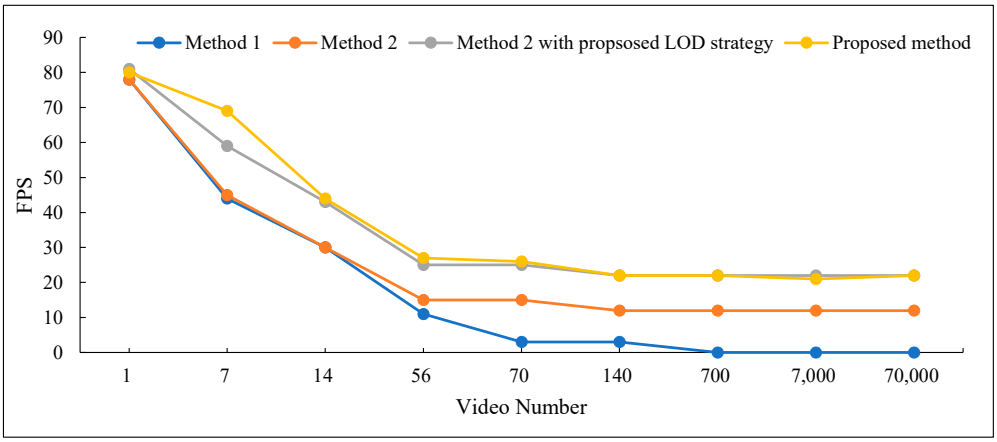



**Figure 12.** Frame rates of different methods at 584 m.

(2) When the scene camera was positioned 126 m above the ground, we conducted a comparative experiment. The results demonstrated that the visualization outcomes of the four methods remained consistent. To explain in detail, we selected Method 2 as an example and presented its visualization results in Figure 13. Moreover, it is worth noting that when the number of videos reached 700 or more, Method 1 failed to render due to the browser's limited computational capacity and storage space. Furthermore, because the scene camera was positioned relatively close to the ground, only a limited number of videos could be displayed. Within the same range, when the number of videos was seven or more, both the proposed method and the comparison methods produced visualization results consistent with those shown in Figure 13b.

As shown in Figure 14, the method proposed in this paper demonstrated higher efficiency compared to the other methods. Specifically, Method 1 demonstrated a similar trend in frame rate as that observed at 584 m. When the number of video feeds exceeds 700, the browser could not run it. Method 2 alone and Method 2 combined with the proposed LOD strategy maintained a generally consistent frame rate, mostly staying above 42 FPS. However, due to the adoption of parallel-optimized strategies, the method proposed in this paper consistently maintained a frame rate above 50 FPS. Compared to the existing optimal method, the frame rate of the method proposed in this paper was improved by 19.0%.

Moreover, to validate the parallel processing capability of the method proposed in this paper for overlapping videos, we constructed an augmented 3DGIS scene incorporating a vast number of overlapping videos, totaling 70,000 surveillance videos. Among these videos, each overlapping area involved seven surveillance videos (with an overlapping longitude span of 0.0003° for each video); meanwhile, the overlapping video areas were separated by 0.005° in longitude and 0.0007° in latitude, respectively.



(**a**) Video number = 1 (**b**) Video number ≥ 7

**Figure 13.** Visualization results for different video numbers at 126 m.

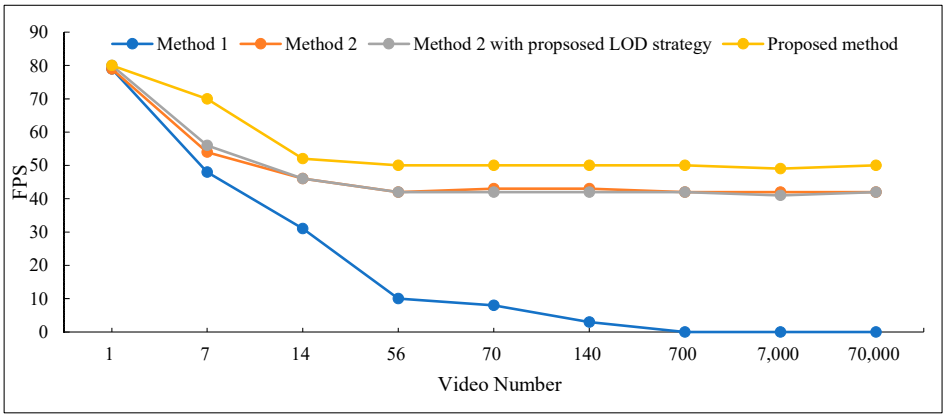



**Figure 14.** Frame rates of different methods at 126 m.

As shown in Figure 15, Method 1, similar to previous experiments, used a direct loading method that resulted in browser malfunction. Method 2 and Method 2 with the proposed LOD strategy exhibited identical visualization results; therefore, only the results of Method 2 are presented in Figure 15. A comparison of these results reveals that when the scene camera was positioned at distances of 533 m and 370 m from the ground, the proposed method displayed relatively consistent visualization results with Method 2 and Method 2 with the proposed LOD strategy. This consistency is primarily because the video-fused scene occupied only a small fraction of the screen's pixels, making detailed features indistinct. However, at distances of 126 m and 64 m, due to the closer proximity of the scene camera to the video-fused area, the video overlap effect became more pronounced. In these cases, the proposed method effectively distinguished itself from other methods. As is clear from Figure 15c,d, the proposed method successfully avoided artifacts caused by video overlap, exhibiting superior visualization effects, while the comparison methods exhibited significant ghosting artifacts.



(**a**) Visualization results at 533 m

**Figure 15.** *Cont.*



(**b**) Visualization results at 370 m



(**c**) Visualization results at 126 m



(**d**) Visualization results at 44 m

**Figure 15.** Visualization of overlapping videos captured from various distances. (left: Method 2 or Method 2 with proposed LOD strategy, right: proposed method).

This experiment documented frame rates of various methods at different distances, and the results are depicted in Figure 16.

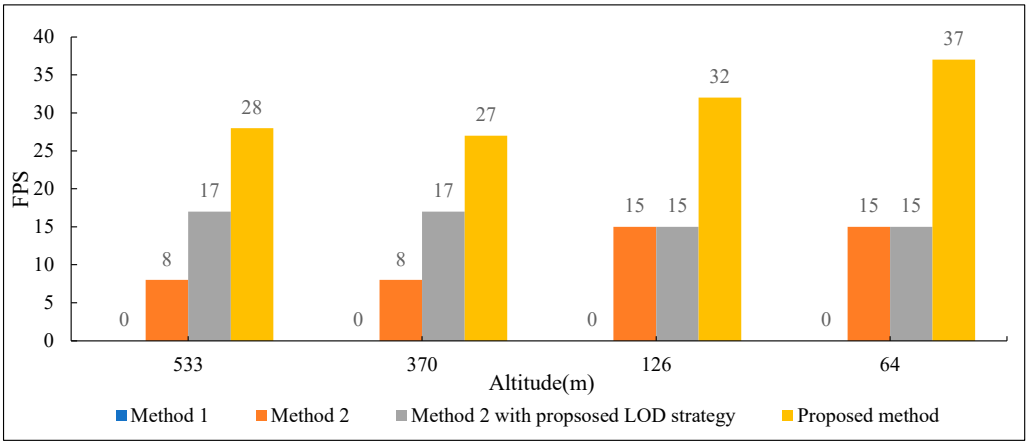



**Figure 16.** Frame rates of different methods at various altitudes.

As shown in Figure 16, this experiment recorded the frame rates of different methods at 533 m, 370 m, 126 m, and 64 m, respectively. Among them, Method 1 could not run, and Method 2 maintained FPS within the range of 8~15 at different altitudes. Compared to Method 2 with the proposed LOD strategy, Method 2 had a higher FPS by 7 at 533 m and 370 m, while the two methods performed similarly at 126 m and 64 m. This is primarily due to the LOD reaching its maximum level at closer distances, leading to similar efficiency for both methods. However, the proposed method achieved higher frame rates at all altitudes. Compared to previous methods, the rendering frame rate of the proposed method increased by 11FPS, 10FPS, 17FPS, and 22FPS at different altitudes, respectively. In other words, the rendering frame rate was improved by an average of 95%. Therefore, the proposed method can effectively enhance the rendering efficiency in overlapping areas. It is worth noting that the proposed method needed four rendering pipelines at 126 m, whereas only a single rendering pipeline was used for parallel processing at 64 m.

In summary, the proposed method adopts the proposed organization method and scheduling algorithm to achieve visualization for large-scale geographical scenes enhanced with surveillance videos on Cesium. Additionally, compared to other methods, this method achieves better display effects and operational efficiency.

## 4. Discussion

This study primarily focuses on the organization and scheduling of multi-video fusion scenarios in large-scale geographic settings. It proposes a parallel-optimized visualization approach specifically for large-scale geographic scenes augmented with multiple videos on the Cesium platform. While previous studies have explored the visualization of large-scale video-integrated scenes, they have not adequately addressed the challenges posed by differences in WebGIS scenarios, real-time video requirements, multi-video overlaps, and GPU efficiency optimization. The proposed method optimizes the handling of overlapping effects among multiple videos and significantly enhances the rendering efficiency of large-scale video-augmented geographic scenes, providing more real-time video information for subsequent 3D geographic applications.

The comparative experiments have validated the effectiveness and efficiency of the proposed method. However, several inherent performance limitations remain. Firstly, the Cesium environment necessitates browser support for WebGL, which varies across different software and hardware environments, including graphics card configurations, system setups, and browsers, leading to inconsistent performance. Secondly, a significant limitation of the method proposed in this paper, stemming from WebGL, is that it typically restricts the simultaneous processing of textures to only 14 within a single rendering

pipeline during parallel computations. Specifically, the proposed method, combined with the shader algorithms defined here, allows each shader to handle a maximum of seven video fusions concurrently. Fortunately, the practical deployment of surveillance cameras typically considers both cost and utilization rates, aiming to achieve optimal coverage during installation. Consequently, the number of overlapping videos is generally not excessive [39]. Thirdly, transmitting large-scale video data between the GPU and CPU can create a bottleneck.

Additionally, the method proposed in this paper faces research method limitations, such as the video data processing of surveillance cameras, the fusion of videos with geographic scenes, and the scheduling of large-scale video scenes, which require further investigation in future research.

First, the pose parameters of surveillance cameras are assumed to be precise after camera calibration. However, in practical applications, there may be issues with inaccurate positioning, or the pose parameters of each surveillance camera may be unknown, which can affect subsequent fusion results [22]. Therefore, in future research, we plan to address these issues by employing visual matching positioning methods and camera distortion correction techniques, and by leveraging precise spatial features obtained through visual analysis of geometric relationships between geographic scenes and surveillance camera videos [40,41].

Second, while this study examines multi-video fusion scenarios in large-scale geographic settings and improves visualization effects compared to existing methods, it primarily focuses on the multi-video fusion scenario method and does not consider strategies such as large inclination angles for multi-video fusion cameras, optimal segmentation, and transitions. In future research, issues related to large inclination angles between videos and geographic scene textures, as well as segmentation strategies, will be incorporated into our method to further enhance the rendering effects.

Lastly, this study introduces a scheduling algorithm tailored for large-scale video-augmented geographic scenes using the Cesium platform. Among them, the dynamic LOD algorithm within the scheduling algorithm sets the LOD models based on distance values, suitable for most current scenario applications. However, the effectiveness of our LOD scheduling algorithm may be compromised in scenarios involving frequent movement or varying levels of video importance. Therefore, in future research, the proposed dynamic LOD scheduling algorithm should take into account distance factors and a maximum threshold for the current number of videos. Additionally, it should comprehensively rank videos based on their importance, prioritizing the loading of more important video-augmented scenes. Considering both of these factors will cater to the diverse applications in real-world scenarios [42,43].

## 5. Conclusions

Videos, a crucial data source for WebGIS, provide extensive spatio-temporal characteristic information. This paper addresses the challenges with integrating large-scale videos into geographic scenes, particularly issues such as artifacts and low rendering efficiency. To overcome these challenges, we leverage the out-of-core rendering technology and GPU parallel optimization techniques. The contributions of this article can be summarized as follows: Firstly, we introduce an octree-based model tailored for large-scale surveillance videos. This model enhances the visualization effects by uniformly organizing and managing overlapping areas within data organization and scheduling algorithms, eliminating potential artifact issues, and ensuring appropriate rendering. Secondly, the proposed scheduling algorithm is implemented using GPU parallel optimization, dynamic LOD strategy, and scheduling mechanisms on the Cesium platform, which significantly improves the rendering efficiency of integrating large-scale surveillance videos into 3DGIS. Experimental results demonstrate that our method outperforms the existing approaches in terms of visual effects and efficiency when fusing multiple videos with 3D real scenes. Therefore, the proposed method holds certain applicability to current practical applications.

Fusing large-scale multi-video into geographical scenes enhances the representation of dynamic and real-time spatial information within the virtual environment, offering users an immersive visual experience. Furthermore, this approach provides reliable data support for digital city management and surveillance. The method proposed in this article enables the visualization of large-scale video-fused scenes. However, in practical systems, other functional computations may limit the application of the proposed method. Therefore, we will elaborate on specific applications, such as smart city monitoring, real-time surveillance systems, and multi-video trajectory extraction [44–46].

**Supplementary Materials:** The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/ijgi13120463/s1, Video S1: Video-augmented GIS method in oblique photograph models; Video S2: Large-scale visualization for non-overlapping surveillance scenes; Video S3: Large-scale visualization for overlapping surveillance scenes.

**Author Contributions:** Conceptualization, Qingxiang Chen and Jing Chen; methodology, Qingxiang Chen and Jing Chen; software, Qingxiang Chen and Jing Chen; validation, Qingxiang Chen and Minmin Huang; formal analysis, Qingxiang Chen, Guang Chen and Hao Liu; investigation, Minmin Huang; resources, Minmin Huang; data curation, Qingxiang Chen and Minmin Huang; writing—original draft preparation, Qingxiang Chen and Jing Chen; writing—review and editing, Qingxiang Chen and Jing Chen; visualization, Qingxiang Chen and Minmin Huang; supervision, Jing Chen, Kaimin Sun, Minmin Huang, Guang Chen and Hao Liu; project administration, Qingxiang Chen, Kaimin Sun and Minmin Huang; funding acquisition, Qingxiang Chen, Kaimin Sun, Minmin Huang, Guang Chen and Hao Liu. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Key Project of Natural Science Research of Anhui Provincial Department of Education, grant number 2023AH051592; the research project of Chuzhou University, grant number 2023qd05; the Open Project of Technology Innovation Center for Spatio-temporal Information and Equipment of Intelligent City, grant number STIEIC-KF202302; and the Key Project of Natural Science Research of Anhui Provincial Department of Education, grant number 2023AH051633.

**Data Availability Statement:** The datasets are available on request from the first author.

**Acknowledgments:** Comments from the reviewers and editors are appreciated.

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

## References

1. Myagmar-Ochir, Y.; Kim, W. A Survey of Video Surveillance Systems in Smart City. *Electronics* **2023**, *12*, 3567. [CrossRef]
2. Milosavljević, A.; Rančić, D.; Dimitrijević, A.; Predić, B.; Mihajlović, V. Integration of GIS and Video Surveillance. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 2089–2107. [CrossRef]
3. Xie, Y.; Wang, M.; Liu, X.; Wu, Y. Surveillance Video Synopsis in GIS. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 333. [CrossRef]
4. 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]
5. Agrawal, A.; Kar, S. Development of a Landslide Detection Surveillance System. In Proceedings of the 2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP), Sofia, Bulgaria, 1–3 June 2022; Volume CFP2255E-ART, pp. 1–4.
6. Munyo, I.; Rossi, M.A. Police-Monitored Cameras and Crime. *Scand. J. Econ.* **2020**, *122*, 1027–1044. [CrossRef]
7. Cho, K.; Cho, D. Autonomous Driving Assistance with Dynamic Objects Using Traffic Surveillance Cameras. *Appl. Sci.* **2022**, *12*, 6247. [CrossRef]
8. Xie, Y.; Wang, M.; Liu, X.; Mao, B.; Wang, F. Integration of Multi-Camera Video Moving Objects and GIS. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 561. [CrossRef]
9. Jamonnak, S.; Zhao, Y.; Curtis, A.; Al-Dohuki, S.; Ye, X.; Kamw, F.; Yang, J. GeoVisuals: A Visual Analytics Approach to Leverage the Potential of Spatial Videos and Associated Geonarratives. *Int. J. Geogr. Inf. Sci.* **2020**, *34*, 2115–2135. [CrossRef]
10. Shao, Z.; Li, C.; Li, D.; Altan, O.; Zhang, L.; Ding, L. An Accurate Matching Method for Projecting Vector Data into Surveillance Video to Monitor and Protect Cultivated Land. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 448. [CrossRef]
11. Han, S.; Dong, X.; Hao, X.; Miao, S. Extracting Objects' Spatial–Temporal Information Based on Surveillance Videos and the Digital Surface Model. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 103. [CrossRef]
12. Luo, X.; Wang, Y.; Dong, J.; Li, Z.; Yang, Y.; Tang, K.; Huang, T. Complete Trajectory Extraction for Moving Targets in Traffic Scenes That Considers Multi-Level Semantic Features. *Int. J. Geogr. Inf. Sci.* **2023**, *37*, 913–937. [CrossRef]

13. Milosavljević, A.; Dimitrijević, A.; Rančić, D. GIS-Augmented Video Surveillance. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 1415–1433. [CrossRef]
14. Cui, X.; Khan, D.; He, Z.; Cheng, Z. Fusing Surveillance Videos and Three-dimensional Scene: A Mixed Reality System. *Comput. Animat. Virtual Worlds* **2023**, *34*, e2129. [CrossRef]
15. Jian, H.; Liao, J.; Fan, X.; Xue, Z. Augmented Virtual Environment: Fusion of Real-Time Video and 3D Models in the Digital Earth System. *Int. J. Digit. Earth* **2017**, *10*, 1177–1196. [CrossRef]
16. Xie, Y.; Wang, M.; Liu, X.; Wang, X.; Wu, Y.; Wang, F.; Wang, X. Multi-camera Video Synopsis of a Geographic Scene Based on Optimal Virtual Viewpoint. *Trans. GIS* **2022**, *26*, 1221–1239. [CrossRef]
17. de Haan, G.; Scheuer, J.; de Vries, R.; Post, F.H. Egocentric Navigation for Video Surveillance in 3D Virtual Environments. In Proceedings of the 2009 IEEE Symposium on 3D User Interfaces, Lafayette, LA, USA, 14–15 March 2009; pp. 103–110.
18. Zhu, G.; Zhang, H.; Jiang, Y.; Lei, J.; He, L.; Li, H. Dynamic Fusion Technology of Mobile Video and 3D GIS: The Example of Smartphone Video. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 125. [CrossRef]
19. Zhang, X.; Shi, X.; Luo, X.; Sun, Y.; Zhou, Y. Real-Time Web Map Construction Based on Multiple Cameras and GIS. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 803. [CrossRef]
20. Zhang, X.; Li, Q.; Li, X.; Shi, X.; Li, H. A New Method for Orthographic Video Map Construction Based on Video and GIS. *Geocarto Int.* **2023**, *38*, 2289450. [CrossRef]
21. Zhang, X.; Sun, Y.; Li, Q.; Li, X.; Shi, X. Crowd Density Estimation and Mapping Method Based on Surveillance Video and GIS. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 56. [CrossRef]
22. Milosavljević, A.; Rančić, D.; Dimitrijević, A.; Predić, B.; Mihajlović, V. A Method for Estimating Surveillance Video Georeferences. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 211. [CrossRef]
23. Tao, W.; Xu, D.; Chen, X.; Tan, G. A Powerful Correspondence Selection Method for Point Cloud Registration Based on Machine Learning. *Photogramm. Eng. Remote Sens.* **2023**, *89*, 703–712. [CrossRef]
24. Shi, L.; Lan, X.; Lan, X.; Zhang, T. Quick Calibration of Massive Urban Outdoor Surveillance Cameras. *ISPRS J. Photogramm. Remote Sens.* **2024**, *214*, 231–243. [CrossRef]
25. Lewis, P.; Fotheringham, S.; Winstanley, A. Spatial Video and GIS. *Int. J. Geogr. Inf. Sci.* **2011**, *25*, 697–716. [CrossRef]
26. Li, C.; Liu, Z.; Zhao, Z.; Dai, Z. A Fast Fusion Method for Multi-Videos with Three-Dimensional GIS Scenes. *Multimed. Tools Appl.* **2021**, *80*, 1671–1686. [CrossRef]
27. Fan, Z. Research on Registration and Rendering Method of Video to Enhance 3D Scene. *Acta Geod. Cartogr. Sin.* **2019**, *48*, 801.
28. 3D Tiles—Open Geospatial Consortium. Available online: https://www.ogc.org/standard/3DTiles/ (accessed on 9 October 2024).
29. Indexed 3D Scene Layers (I3S). Available online: https://www.ogc.org/standard/i3s/ (accessed on 9 October 2024).
30. S3m-Spec/Specification at Master · SuperMap/S3m-Spec. Available online: https://github.com/SuperMap/s3m-spec/tree/master/Specification (accessed on 9 October 2024).
31. 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]
32. Persad, R.A.; Armenakis, C.; Sohn, G. Automatic Co-Registration of Pan-Tilt-Zoom (PTZ) Video Images with 3D Wireframe Models. *Photogramm. Eng. Remote Sens.* **2015**, *81*, 847–859. [CrossRef]
33. Texture—OpenGL Wiki. Available online: https://www.khronos.org/opengl/wiki/Texture (accessed on 9 October 2024).
34. Cheng, C.; Tong, X.; Chen, B.; Zhai, W. A Subdivision Method to Unify the Existing Latitude and Longitude Grids. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 161. [CrossRef]
35. Cheng, R.; Chen, J.; Cao, M. A Virtual Globe-Based Three-Dimensional Dynamic Visualization Method for Gas Diffusion. *Environ. Model. Softw.* **2019**, *111*, 13–23. [CrossRef]
36. CesiumJS 1.106. Available online: https://github.com/CesiumGS/cesium/releases/tag/1.106 (accessed on 1 December 2024).
37. Microsoft Visual Sudio Code 1.95.1. Available online: https://www.filepuma.com/download/visual_studio_code_64bit_1.95.1-42581/ (accessed on 1 December 2024).
38. Chrome Browser 131.0.6778.86 (64 Bit). Available online: https://www.neowin.net/software/google-chrome-1310677886-offline-installer/ (accessed on 1 December 2024).
39. Lan, X.; Zhou, L.; Lin, B.; Li, J.; Lv, G. Rapid Survey Method for Large-Scale Outdoor Surveillance Cameras Using Binary Space Partitioning. *ISPRS J. Photogramm. Remote Sens.* **2024**, *207*, 57–73. [CrossRef]
40. Liu, X.; Hu, Q.; Huang, X. A Keyframe Extraction Approach for 3D Videogrammetry Based on Baseline Constraints. *Photogramm. Eng. Remote Sens.* **2024**, *90*, 171–180. [CrossRef]
41. Ye, J.-F.; Tsay, J.-R.; Fritsch, D. Wavelets for Self-Calibration of Aerial Metric Camera Systems. *Photogramm. Eng. Remote Sens.* **2024**, *90*, 575–587. [CrossRef]
42. Zahra, A.; Ghafoor, M.; Munir, K.; Ullah, A.; Ul Abideen, Z. Application of Region-Based Video Surveillance in Smart Cities Using Deep Learning. *Multimed. Tools Appl.* **2024**, *83*, 15313–15338. [CrossRef] [PubMed]
43. Nallola, S.R.; Ayyasamy, V. Twenty-Five Years of Real-Time Surveillance Video Analytics: A Bibliometric Review. *Multimed. Tools Appl.* **2024**, *83*, 69273–69306. [CrossRef]
44. Li, X.; Duan, H.; Liu, B.; Wang, X.; Wang, F.-Y. A Novel Framework to Generate Synthetic Video for Foreground Detection in Highway Surveillance Scenarios. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5958–5970. [CrossRef]

45. Zhang, X.; Feng, Y.; Angeloudis, P.; Demiris, Y. Monocular Visual Traffic Surveillance: A Review. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 14148–14165. [CrossRef]
46. Jian, H.; Fan, X.; Yan, Z.; Huang, M. 3D Path Prediction of Moving Objects in a Video-Augmented Indoor Virtual Environment. *Int. J. Digit. Earth* **2021**, *14*, 1818–1834. [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.