buildings

MDPI

Article

A Three-Dimensional Triangle Mesh Integration Method for Oblique Photography Model Data

Defu Che [1,2,*], Min Su [1,2], Baodong Ma [1,2], Feng Chen [1,2], Yining Liu [1,2], Duo Wang [3] and Yanen Sun [4]

[1] Institute for Geoinformatics & Digital Mine Research, Northeastern University, Shenyang 110819, China
[2] Key Laboratory of Ministry of Education on Safe Mining of Deep Metal Mines, Northeastern University, Shenyang 110819, China
[3] Shen Kan Engineering & Technology Corporation, MCC, Shenyang 110100, China
[4] Feny Corporation Limited, Changsha 410600, China
* Correspondence: chedefu@mail.neu.edu.cn

Abstract: Oblique photography 3D models are increasingly used in 3D modeling and visualization, urban planning and design, smart cities, smart mines, and other fields. To ensure that 3D models can be replaced and updated, it is necessary to deal with the model holes and incomplete areas caused by the incorrect matching of feature points in the modeling process. Moreover, in 3D models of the planning and management of certain mining areas or urban areas to be developed, regions can also be delineated for replacements and updates. Due to the unsatisfactory display effect of the manually modeled models used to replace and update, the authenticity of the model texture and the detail characteristics of the model triangle meshes cannot be guaranteed during integration. Therefore, this paper proposes a 3D model triangle mesh integration method for oblique photography data which integrates the real and complete 3D models used for replacement with the incomplete part or the part of the original model that needs to be replaced. The experimental results show that this method can integrate the original terrain model and the replacement model efficiently and quickly, the integrated model has no gap, and the display effect is good, effectively achieving model repair or model update and replacement.

Keywords: oblique photography data; 3D model; data integration; triangle mesh cutting; 3D reconstruction

check for updates

Citation: Che, D.; Su, M.; Ma, B.; Chen, F.; Liu, Y.; Wang, D.; Sun, Y. A Three-Dimensional Triangle Mesh Integration Method for Oblique Photography Model Data. *Buildings* **2023**, *13*, 2266. https://doi.org/10.3390/buildings13092266

Academic Editors: Binsheng (Ben) Zhang and Krishanu Roy

Received: 17 July 2023
Revised: 31 August 2023
Accepted: 1 September 2023
Published: 6 September 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

With the rapid development of the social economy and the acceleration of digital informatization, the concepts of smart cities and smart mines have been proposed and applied one after another. These cities and mines can be visualized in 3D based on 3D models using oblique photography [1,2]. This has important uses in city mapping, urban planning, green space and landscape planning, ecological restoration, design of mining areas, and so on [3,4]. Oblique photography 3D reconstruction utilizes multiple optical sensors to simultaneously collect data from multiple angles on the target area, and it can quickly and efficiently restore the structure of ground objects and attach real textures, thus providing real information of ground objects and effectively reducing the cost of modeling, especially in the fields of 3D model construction [5]. It has become one of the most used tools for city and mine modeling [6,7].

Oblique aerial photography is achieved with multiple cameras on a UAV to collect image data from multiple angles at the same time. Through software processing, a 3D model is generated from oblique aerial photographs. This 3D model can accurately express the terrain and surface of a 3D scene in a covered route [8], but in the meantime, the production process of oblique photography 3D models is characterized by automatic and non-artificial intervention [9], so the modeling quality is greatly affected by oblique imagery. In these images, weak textures [10] (such as water surfaces), illumination differences, object

blocking, and other factors have a great influence on the automatic extraction of image features and matching [11], and this can easily produce false matching or insufficient matching points. As a result, 3D models from oblique photography are prone to problems such as holes [12], distorted houses [13], uneven roads [14], incomplete street lights [15], broken vehicles [16], and suspended objects [17]. Therefore, the problematic areas in the model need to be replaced. In addition, in order to avoid the generated 3D model being used only once or only for observation and browsing, the 3D models used for mining area management or some urban areas to be developed can also be delineated for replacements and updates.

Two problems arise. One is the selection of the model to be used for the replacement. Replacement models such as buildings can be manually modeled in existing 3D model editing software, but this is time consuming and labor intensive, whereas oblique photography 3D model monomerization can not only reuse the model but also save the time of manual modeling. Ye et al. [18] proposed a cutting monomorphic method using the separability of triangular networks. This method can analyze triangular blocks with different crossing patterns and propose different cutting schemes to achieve solid separation of ground objects in oblique photography 3D models. Since the storage format of the model after monomerization does not change, and it is easy to load into the original data, it is also convenient to edit the model. The other problem is how to integrate the replacement model with the original terrain model. Schneider et al. [19] studied the integrated display of vector spatial elements and terrains. In this study, vector elements were proposed to be superimposed on realistic 3D terrains to facilitate more accurate spatial analysis. The vector elements involved included building models, vegetation, and soil types, etc. Agugiaro et al. [20] proposed a method to integrate a high-resolution surface model and a low-resolution surface model, both embedded in a triangular grid in a three-dimensional space, taking into account the difference in qualities between the low-resolution and high-resolution models and then creating a transition surface. Xie et al. [21] proposed a seamless integration method for TIN and GRID hybrid data structures and multi-resolution models, and conducted experimental verification using a typical highway model. Geng et al. [22] studied a new data fusion algorithm based on an external buffer and TIN tile pyramid to address the issue of existing data fusion algorithms not being able to solve the fusion problem between 3D models of oblique photography and large-scene terrains. Noardo [23] used the processed IFC model to replace an outdated 3D city model, and modified the boundary object to obtain a watertight fusion model.

The existing literature demonstrates a great deal of research on multi-resolution model integration and oblique photography data fusion, but has not solved the problems of the efficient use of model data and the efficient cutting and reconstruction of triangular meshes during model integration. With these problems in mind, this paper proposes a new integration method that can replace and repair the incomplete part or the part that needs to be updated in a three-dimensional oblique photography model. First, load the monomerization model extracted for replacement at the location where the original terrain model needs to be replaced and updated, adjust its position, size and angle to adapt to the scene, then draw the boundary line on the 3D model and project it. Through the position relationships between the triangle mesh and the cutting line and between the triangle mesh cutting and reconstruction algorithm proposed in this paper, the original terrain model and the replacement model are cut and reconstructed, respectively, and finally the seamless integration between the models is completed so that the processed oblique photography 3D model can meet the requirements of digital city or digital mine planning and construction.

2. Data and Methods

2.1. Data Acquisition

The data of the oblique photography 3D model were sourced from images and point cloud information of ground buildings in different paths of a certain region collected by a UAV equipped with multiple sensors pointing in various directions [24]. The sensor

modules included a high-precision inertial navigation module, integrated LiDAR scanner module, three-axis gimbal, and mapping camera [25]. In the process of data acquisition, the LiDAR scanner obtains the point cloud data of ground surface objects [26], the mapping camera obtains the texture information of ground surface objects [27], the high-precision inertial navigation and GNSS module is responsible for recording the position information of the UAV when acquiring image and point cloud information [28], and the three-axis gimbal is used to adjust the angle of the mapping camera and other modules and prevent jitter [29].

The initial data acquired when the UAV passed through the different paths were imported into a mature 3D reconstruction software named "Context Capture". After image correction, aerial triangulation encryption, multi-view image matching, regional network adjustment, point cloud matching, triangulation, texture mapping, and other steps of 3D reconstruction [30], a continuous triangle mesh model with real texture mapping was developed [31]. Finally, it was saved in OSGB [32] (OpenSceneGraph Binary, a binary storage of 3D model data with embedded linked texture data) format to generate the 3D original terrain model. Figure 1 shows the original terrain model used in this paper.



Figure 1. Original terrain model.

2.2. Methods

The proposed 3D triangle mesh integration method for oblique photography model data is divided into three steps: (1) drawing the 3D model boundary line; (2) 3D triangle mesh cutting and reconstruction; (3) 3D model integration.

Artificial structure models such as buildings and gardens are 3D models that can be used for updating and replacing, and are commonly referred to as replacement models in this paper, which will be described in detail in Section 3. The process is shown as follows: import the replacement model into the 3D original terrain model, adjust the position and size, and manually draw the 3D model boundary line. Its projection is used as the cutting line to cut the replacement model and the 3D original terrain model separately. It is worth noting that when drawing the 3D model boundary line, because the coordinates obtained by double clicking the model with the mouse are screen coordinates, it cannot be directly used for drawing on the 3D model. Therefore, the 3D coordinates need to be calculated inversely according to the 2D screen coordinates.

Three-dimensional triangle mesh cutting and reconstruction is realized by physical segmentation and reconstruction, and model triangle meshes and texture images are processed and then re-matched according to the position relationship between the cutting line and triangle mesh. When cutting and reconstructing the 3D model triangle meshes, the 3D model boundary line should be drawn on the replacement model. The 3D model boundary line and the 3D model triangle meshes should be projected onto the XOY plane

along the Z axis, and the triangle mesh projection under each level of detail (LOD) should be cut according to the cutting line.

Three-dimensional model integration uses the special spatial partition data storage structure of the OSGB model, while the operation tasks are processed in different threads according to Tile files, and the triangle meshes and texture information of the oblique photography 3D model for each thread are reconstructed. If using the approach of loading all models in memory to modify and save them locally, firstly, this is very demanding on computer performance and can easily cause the program to crash. Secondly, if a program crash occurs during the saving process, this can easily cause a loss of the source data, resulting in serious consequences. In this paper, considering the above situation comprehensively and combined with the special storage mode of the oblique photography 3D model, a database is created for each Tile folder of the 3D model, and the name of each model is saved. The Tile file and the bounding box coordinates then identify the files that need to be processed through collision detection and divide the specific files into separate threads for processing.

The whole process is shown in Figure 2.

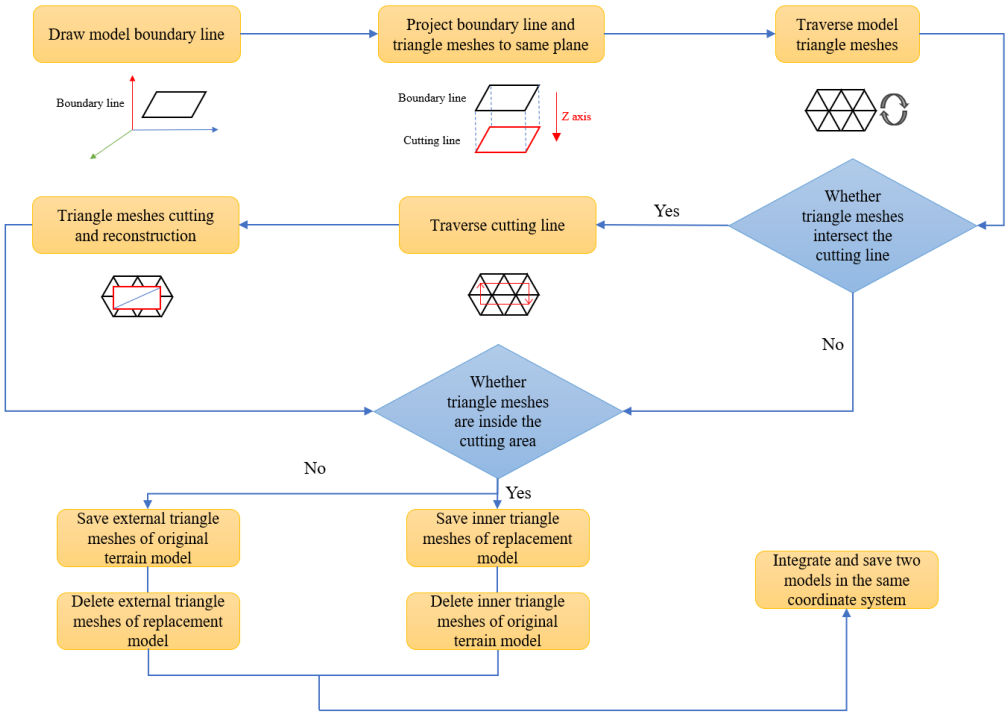


Figure 2. Flowchart of the 3D model integration.

2.2.1. Drawing 3D Model Boundary Line

In the proposed integration method, it is necessary to obtain the 3D model boundary line as the reference line for the 3D model cutting and reconstruction. To ensure that the boundary line intersects with the actual model, we used the manual drawing method, double clicked the mouse to select the point, completed the transformation of the screen coordinates to 3D world coordinates through a series of processes, such as viewport transform inverse transformation, perspective division inverse transformation, projection transform inverse transformation, and model transform inverse transformation, and finally, completed the drawing of the 3D model boundary line.

In the process of transforming the screen coordinates to 3D world coordinates, coordinate transformation is completed by multiplying the view matrix V, the projection matrix P, and the viewport matrix W. Let the product of the three matrices be VPW, its inverse matrix be $[VPW]^{-1}$, the real coordinates of the model, namely, the world coordinates, be

$Coordinate_{world}$, and the two-dimensional coordinates of the screen be $Coordinate_{window}$, then the two-dimensional coordinate transformation can be expressed as Equation (1) [33]:

$$Coordinate_{world} = Coordinate_{window} \times [VPW]^{-1} \quad (1)$$

Through the above coordinate transformation process, a series of 3D points can be obtained by the transformation of screen coordinates by double clicking the mouse, and then the 3D points can be connected to form a closed region and complete the drawing of the 3D model boundary line. As shown in Figure 3, A1, A2, A3, and A4 are the selected 3D points on the replacement model, and connecting these four 3D points constitutes the boundary line. Since the elevation of the points is not consistent, the boundary line forms a polygon in space, and the 3D model cutting and reconstruction proposed in this paper is based on the XOY plane. Therefore, it is necessary to project the boundary line and the triangle meshes related to this region into the XOY plane. P1, P2, P3, and P4 are the points after projection and the red dashed line is the projection of the boundary line in the XOY plane, namely, the cutting line.

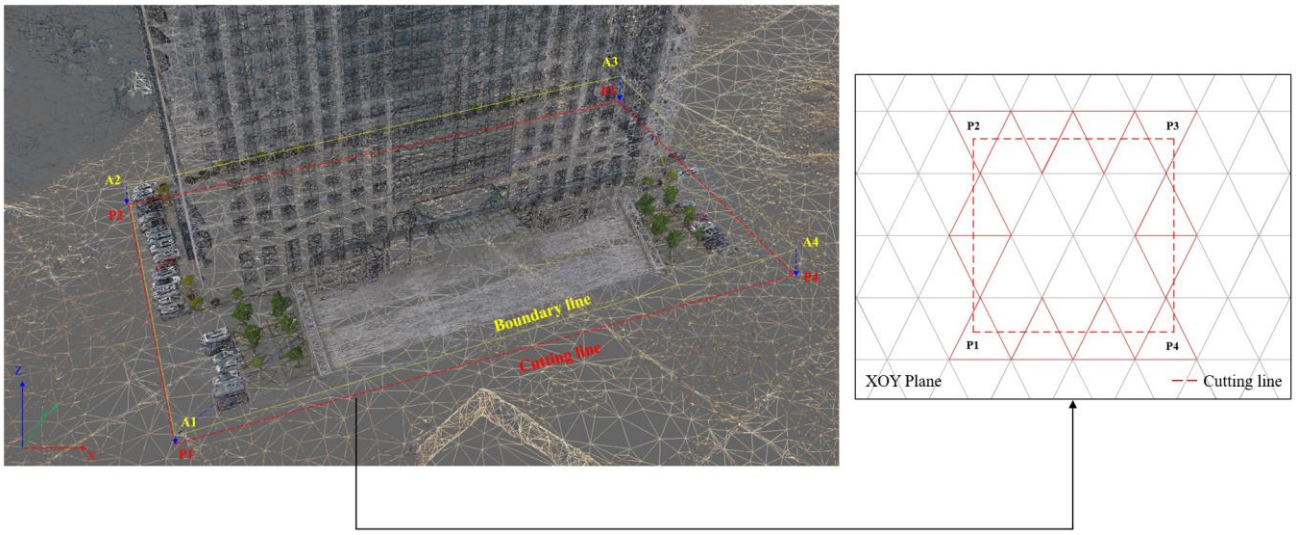


Figure 3. 3D model boundary line drawing and projection.

2.2.2. 3D Triangle Mesh Cutting and Reconstruction

Three-dimensional triangle mesh cutting and reconstruction is the key step in 3D model integration. Its purpose is to cut and process the 3D original terrain model and 3D replacement model, respectively, based on the cutting line. For the 3D original terrain model, we deleted the triangle meshes and texture information inside the cutting line area, and saved the triangle meshes and texture information outside the cutting line area. For the 3D replacement model, the situation was the opposite, and this will ensure the authenticity of the model after integration. This also prevents the appearance of texture stretch, texture flickering, and other problems.

In the process of triangle mesh cutting, the triangle mesh projections under each LOD of the 3D original terrain model and 3D replacement model were cut and reconstructed via sequential sorting according to the cutting line. Finally, the triangle meshes and texture information were completed by cutting, and reconstruction was restored to the 3D space. The specific steps of the 3D triangle mesh cutting and reconstruction are shown below.

Step 1. Project the 3D model triangle meshes in the corresponding region of the Tile folder containing the boundary line to the XOY plane, and project the boundary line to the same plane.

For the cutting line in Figure 3, it is necessary to separate the cutting line into several uncorrelated cutting lines for processing. The cutting line $P1P2P3P4$ needs to be separated into four cutting lines, i.e., $\overrightarrow{P1P2}$, $\overrightarrow{P2P3}$, $\overrightarrow{P3P4}$, and $\overrightarrow{P4P1}$, for processing.

Step 2. Traverse all triangles in the entire triangle mesh when using a specific cutting line to cut the triangle meshes. For a specific triangle, as shown in Figure 4, the three vertices of the triangle are traversed, and the left and right position relationship between each vertex in the triangle and the cutting line are obtained by calculating the slope of the line, as shown in Equation (2):

$$L(E,\ F, A) = (x_E - x_A) \times (y_F - y_A) - (y_E - y_A) \times (x_F - x_A) \tag{2}$$

where E is the starting point of the cutting line, F is the end point of the cutting line, and point A is the vertex to be judged on the triangle. If the result of L is negative, the above equation becomes unequal, and the slope of AE is greater than the slope of AF after transposition. It is concluded that point A to be judged is on the right side of the cutting line EF, and the position attribute *Pos_attribute* of point A is recorded as -1. If L is positive, it means that the slope of AE is less than the slope of AF. It can be concluded that point A to be judged is on the left side of the cutting line EF, and the position attribute *Pos_attribute* of point A is recorded as 1. If the result of L is 0, this means that the slope of AE is equal to the slope of AF, and it can be concluded that point A to be judged is on the cutting line EF and the position attribute *Pos_attribute* of point A is recorded as -1. This is illustrated in Figure 4 below:

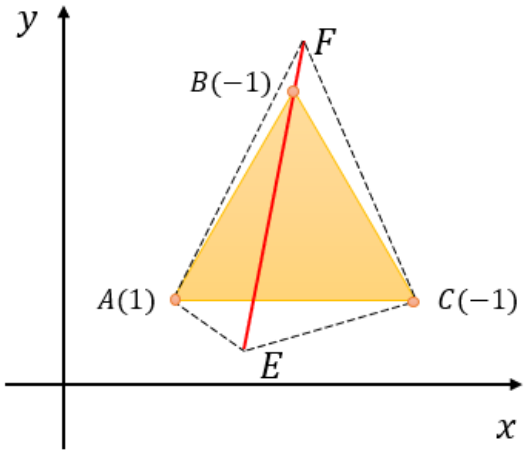


Figure 4. The positional relationship between the triangle vertices and cutting line.

The triangles that do not intersect the cutting line are filtered out by the position attribute *Pos_attribute* if the *Pos_attribute* values of the three vertices in the triangle are judged to be the same, that is, the three vertices of the triangle are on the same side of the cutting line. As shown in Figure 5 below, $\square P1P2P3P4$ is the cutting area, and the triangles with red borders represent the triangles intersecting with the cutting line. When the cutting area is a clockwise polygon, the position attribute of the triangles inside the cutting area that do not intersect with the cutting line is -1, and the position attribute of the triangles outside the cutting area that do not intersect with the cutting line is 1. If the cutting area is a counterclockwise polygon, the situation is the opposite. From Figure 2 of the technical flowchart, it can be seen that for the 3D original terrain model, the triangles located inside the cutting area, that is, the triangles with the same position attribute as triangle E, should be removed, while the triangles located outside the cutting area, that is, the triangles with the same position attributes as triangles A, B, C, and D, should be saved. For the 3D replacement model, the triangles located inside the cutting area should be saved and the triangles located outside the cutting area should be removed.

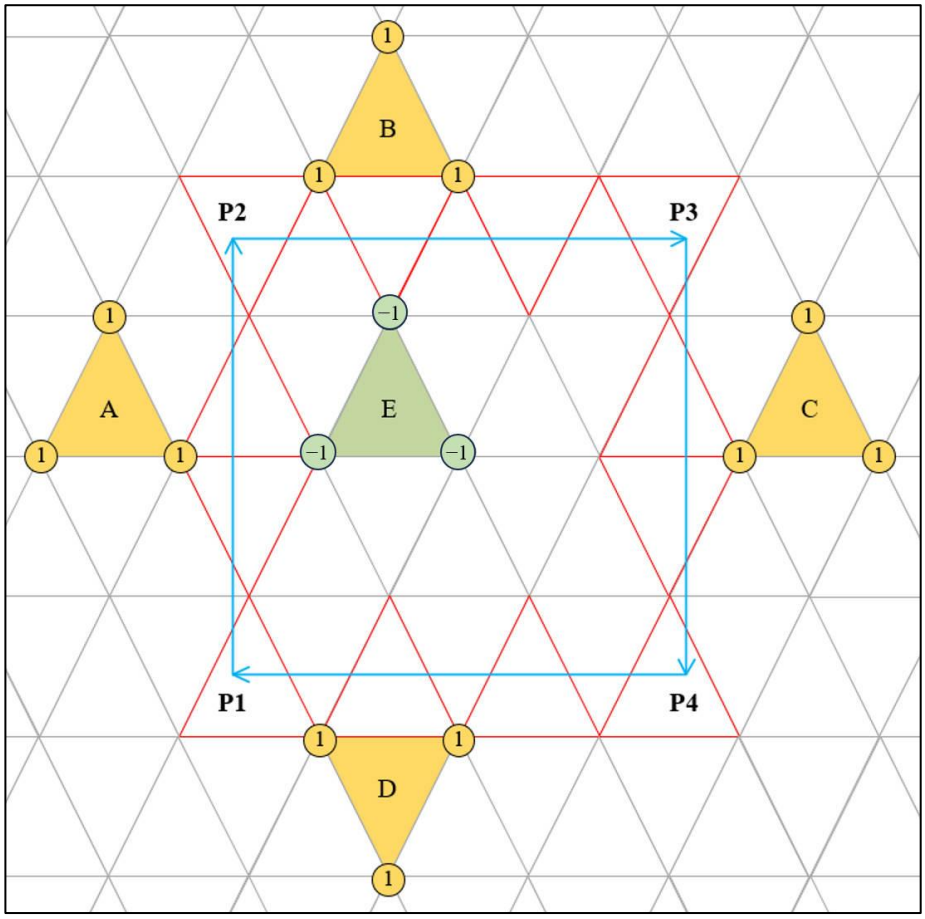


Figure 5. The case where there is no intersection relation between the triangle and the cutting line.

Step 3. For triangles with different position attributes of their vertices, the attribute values of these triangles need to be analyzed to determine the special vertex whose attribute value is different from the other two vertices. The triangle cutting in T3 is shown in Figure 6, where the direction of the cutting line d is $\overrightarrow{E3F3}$. According to Equation (5) in step 2, the position attribute values of A3 and C3 are 1 and that of B3 is -1. That is, B3 is a special vertex in this triangle.

Step 4. The coordinates of the intersection points E3 and F3 of the cutting line d and the triangle should be calculated, and the special points B3, E3, and F3 used to construct the triangle $\Delta B3E3F3$. For the remaining quadrilateral $\square A3E3F3C3$, E3C3 should be connected to construct $\Delta A3E3C3$ and $\Delta E3F3C3$ to complete the triangle cutting. The triangle cuttings in T0 and T2 in Figure 6 are simple forms of the triangle cutting in T3, which will not be repeated here.

Step 5. For cases where the cutting line end point is located inside the triangle, such as the triangle cutting in T4 in Figure 6, extend the cutting line a to intersect triangle $\Delta A4B4C4$ at point G4, and repeat steps 3 and 4 to cut triangle $\Delta A4B4C4$ into three triangles, namely, $\Delta A4E4G4$, $\Delta E4C4G4$, and $\Delta E4B4C4$. Then, use the cutting line d to re-cut the triangle just formed. This process is repeated until the cutting of the triangle mesh is complete. The triangle cutting in T1 in Figure 6 is a simple form of the triangle cutting in T4, which will not be repeated here.

Step 6. Traverse all triangles in the triangle mesh, and the cutting and reconstruction are processed according to the above steps. The results are shown in Figure 7.

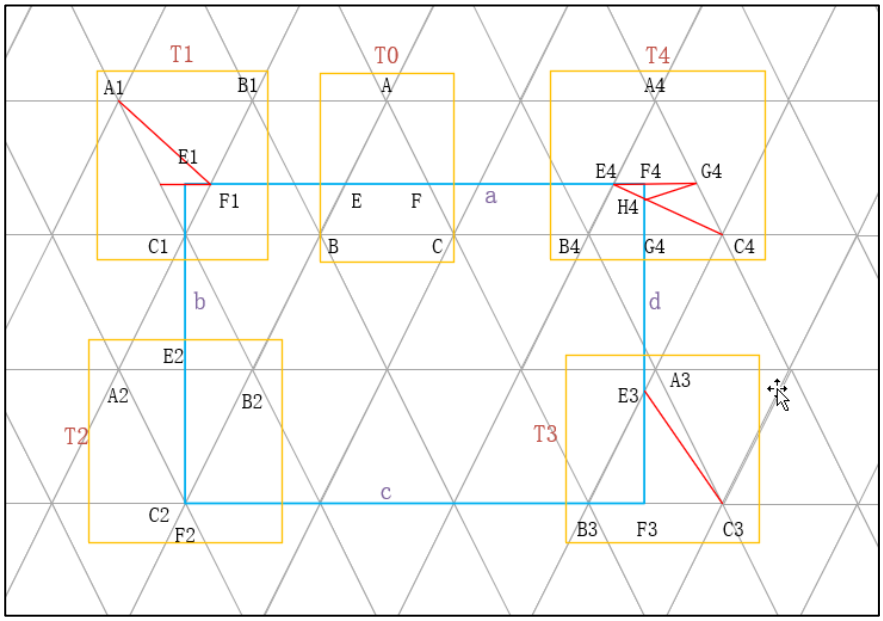


Figure 6. Triangle mesh cutting.

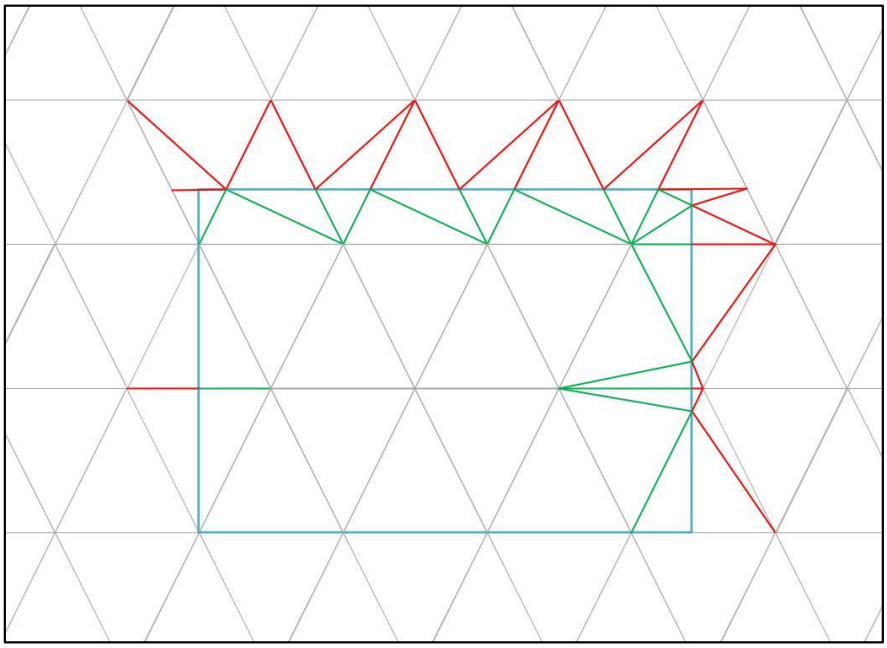


Figure 7. Triangle mesh reconstruction.

Step 7. The cut and reconstructed triangle mesh is restored to the 3D space, and the newly generated point elevation values are obtained by means of spatial linear interpolation. Taking point *E*3 in T3 of Figure 6 as an example, the elevation of point *E*3 is calculated by Equation (3) as:

$$Z_{E3} = Z_{A3} + \frac{dis(A3, E3)}{dis(A3, B3)} \times (Z_{B3} - Z_{A3}) \tag{3}$$

where Z is the elevation coordinate value of the point, and dis is the distance between two points in 3D space.

Calculate texture coordinates. The triangle mesh after cutting and reconstruction also destroys the original texture information. Therefore, it is necessary to calculate the corresponding texture coordinates for the intersection points added after triangle cutting.

In this paper, linear interpolation is used to calculate the texture coordinates, as shown in Equation (4):

$$\begin{cases} u_{new} = u_1 + \frac{(u_2 - u_1) \times d}{D} \\ v_{new} = v_1 + \frac{(v_2 - v_1) \times d}{D} \end{cases} \tag{4}$$

where u_{new} and v_{new} are the texture coordinates of the new intersection point, u_1 and v_1, u_2 and v_2 are the texture coordinates of the vertices at both ends of the side where the new intersection points are located, respectively, d is the distance between the new intersection point and the vertex on one end of the side, and D is the length of the side. Taking T3 in Figure 6 as an example, the texture coordinate of $E3$ is (u_{new}, v_{new}), the side where $E3$ is located is $A3B3$, (u_1, v_1) and (u_2, v_2) are the texture coordinates of $A3$ and $B3$, respectively, d is the length of $A3E3$, and D is the length of $A3B3$.

2.2.3. 3D Model Integration

After cutting the original terrain model and replacement model, the AABB collision detection method is used to detect the intersection file of the two models, the thread pool is used to assign processing tasks, the two models are integrated using their coordinates and then saved to the original terrain model coordinate system, and finally, the 3D model integration is completed. The specific steps are as follows.

Step 1. Retrieve intersecting files using collision detection. Obtain the minimum bounding box Box_A of the replacement model, use the database to obtain the bounding information of each Tile of the original terrain model and use the AABB collision detection method to initially screen out the intersecting Tile files with Box_A. Then, read the information of the model bounding box corresponding to each screened Tile file, and continue to use the AABB collision detection method to screen out the model files that intersect the minimum bounding box Box_B of the original terrain model and the minimum bounding box Box_A of the replacement model. The AABB collision detection method is shown in Figure 8 and Equation (5) [34].

$$\begin{aligned} CrossFlag = \quad & ((A_{min} \cdot x \geq B_{min} \cdot x \&\& A_{min} \cdot x \leq B_{max} \cdot x) \parallel (B_{min} \cdot x \\ & \geq A_{min} \cdot x \&\& B_{min} \cdot x \leq A_{max} \cdot x)) \&\& ((A_{min} \cdot y \\ & \geq B_{min} \cdot y \&\& A_{min} \cdot y \leq B_{max} \cdot y) \parallel (B_{min} \cdot y \\ & \geq A_{min} \cdot y \&\& B_{min} \cdot y \leq A_{max} \cdot y)) \&\& ((A_{min} \cdot z \\ & \geq B_{min} \cdot z \&\& A_{min} \cdot z \leq B_{max} \cdot z) \parallel (B_{min} \cdot z \\ & \geq A_{min} \cdot z \&\& B_{min} \cdot z \leq A_{max} \cdot z)) \end{aligned} \tag{5}$$

where A_{min} and A_{max} are the minimum and maximum point coordinates of Box_A, respectively, and B_{min} and B_{max} are the minimum and maximum point coordinates of Box_B, respectively. When $CrossFlag$ is true, these two boxes intersect; when $CrossFlag$ is false, these two boxes do not intersect.

Step 2. Assign thread tasks based on the screened model files. First, obtain the number of cores of the computer. The number of CPU cores used in this paper was six, so the number of threads to initialize the thread pool was six. Then, the number of tasks was initialized according to the number of OSGB model files screened out. Initialize each thread with each model file and its corresponding bounding box information and boundary line information.

Step 3. Three-dimensional model integration. Obtain the cut and reconstructed original terrain model and replacement model, integrate the triangle meshes of the two models and their corresponding texture information in the same coordinate system, and save to the same rendering node. Finally, the local model file is replaced with the node information after writing, and the conversion from computer memory to local is completed. Finally, the model integration is realized.

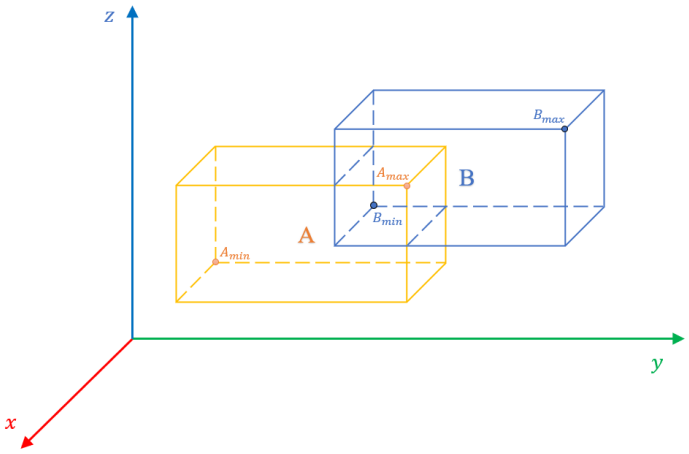
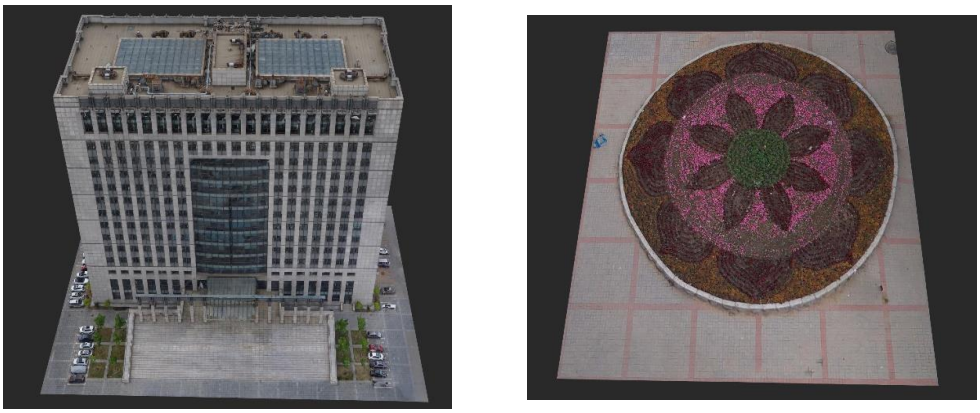


Figure 8. AABB collision detection.

3. Experimental Results and Discussion

Two experimental 3D replacement models were selected for the experiment, which were separately loaded and displayed, as shown in Figure 9. Figure 9a is a building, and Figure 9b is a circular garden. The two experimental models were compared to verify the effect of large-scene model integration and small-scene model integration, respectively. And the coordinate range of these two experimental models is shown in Table 1.



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

Figure 9. 3D replacement models: (**a**) 3D model of building; (**b**) 3D model of circular garden.

Table 1. Properties of the experimental 3D models.

Experimental 3D Models	Coordinate Range (X)	Coordinate Range (Y)	Coordinate Range (Z)
Building	15.4097~127.8028	−244.6184~−166.8273	40.7358~110.0085
Circular Garden	14.9737~29.4751	−339.6959~−324.4265	41.1104~47.5641

The 3D replacement model was loaded into the scene before model integration, and the size, angle and position of the model adjusted with the 3D dragger in OpenSceneGraph to make the model adapt to the scene. Figures 10 and 11 show the replacement models after loading into the scene and adjusting the size and position.



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

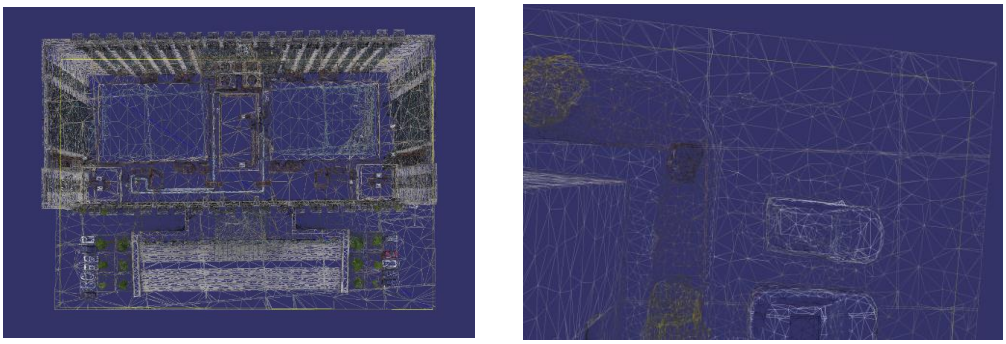
Figure 10. Model display before integration of the building: (**a**) model before integration; (**b**) details of the model before integration.



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

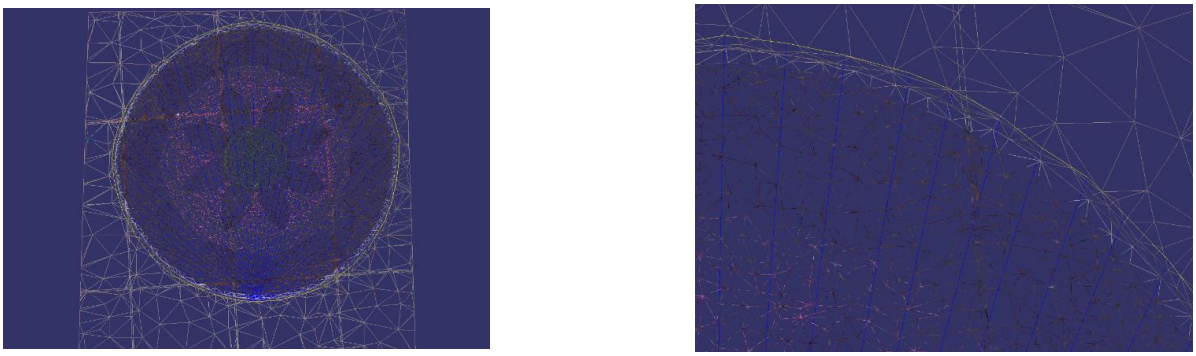
Figure 11. Model display before integration of the circular garden: (**a**) model before integration; (**b**) details of the model before integration.

Then, the boundary line was drawn on the 3D replacement model. In order to facilitate understanding, the mesh mode of the 3D model is added here for display. Figure 12 shows the boundary line of the building. Since the overall shape of the building model was relatively regular, the boundary line was drawn using a rectangle for easy cutting. Figure 13 shows the boundary line of a circular garden. As the model of the garden itself is similar to a circle, the method of polygons approximating circles was used to draw the boundary line.



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

Figure 12. Drawing the boundary line of the building: (**a**) overview of the boundary line; (**b**) details of the boundary line.



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

Figure 13. Drawing the boundary line of the circular garden: (**a**) overview of the boundary line; (**b**) details of the boundary line.

The model integration operation was started after drawing the boundary line. The test environment was on a computer with Windows11 64-bit operating system, 2.50 GHz CPU, 16 GB of RAM. The compilation environment was Microsoft Visual Studio 2017, MFC, the programming language was C++, and the 3D rendering engine was OpenSceneGraph (OSG).

Figures 14 and 15 show the effect of the model integration. As the overall difference between the models before and after integration was not significant, the mesh modes before and after model integration were used for comparison. Taking Figure 14 as an example, it can be seen that the building model before integration in Figure 14a is superimposed on the original terrain model and does not have any impact on the triangle meshes of this area. However, after the building model integration in Figure 14b, it can be seen that the triangle meshes of the original terrain model inside the boundary line have been removed and replaced by the triangle meshes of the building model inside the boundary line, and the triangle meshes at the boundary line have been cut better without gaps. Figure 14c,d show the details of the model after integration.

In the process of the experiment, because the original terrain model is a model of the whole area reconstructed by oblique photography, the files were stored in Tile form and the number of files was relatively large; the number of files of the replacement model was limited compared to it. If all the Tile files of the original terrain model are processed, this will increase the pressure on the computer threads to deal with the files, thus leading to a low processing speed. Therefore, before each model integration, we retrieved the relevant Tile files of the original terrain model through the spatial location of the replacement model in the original terrain model, and only processed these Tile files to shorten the processing time. The general processing time was about 30–60 s.

After completing the triangle mesh cutting, the replacement model needs to be written into the original terrain model file, and the data table created to load the original terrain model file needs to be modified. The corresponding information of the replacement model was added, the memory-to-local conversion was completed in order to achieve the integration of the data, and then the cache was cleared to realize the update of the scene. It can be seen from Figures 14 and 15 that the model integration effect of a single model is good, but it is impossible to modify only one area in general urban planning. Figure 16 shows the planning and updating of an area. Figure 16a shows the regional planning based on the original terrain model, which is divided into four building areas that need to be changed and one green area that needs to be changed. Figure 16b shows the effect after model integration.

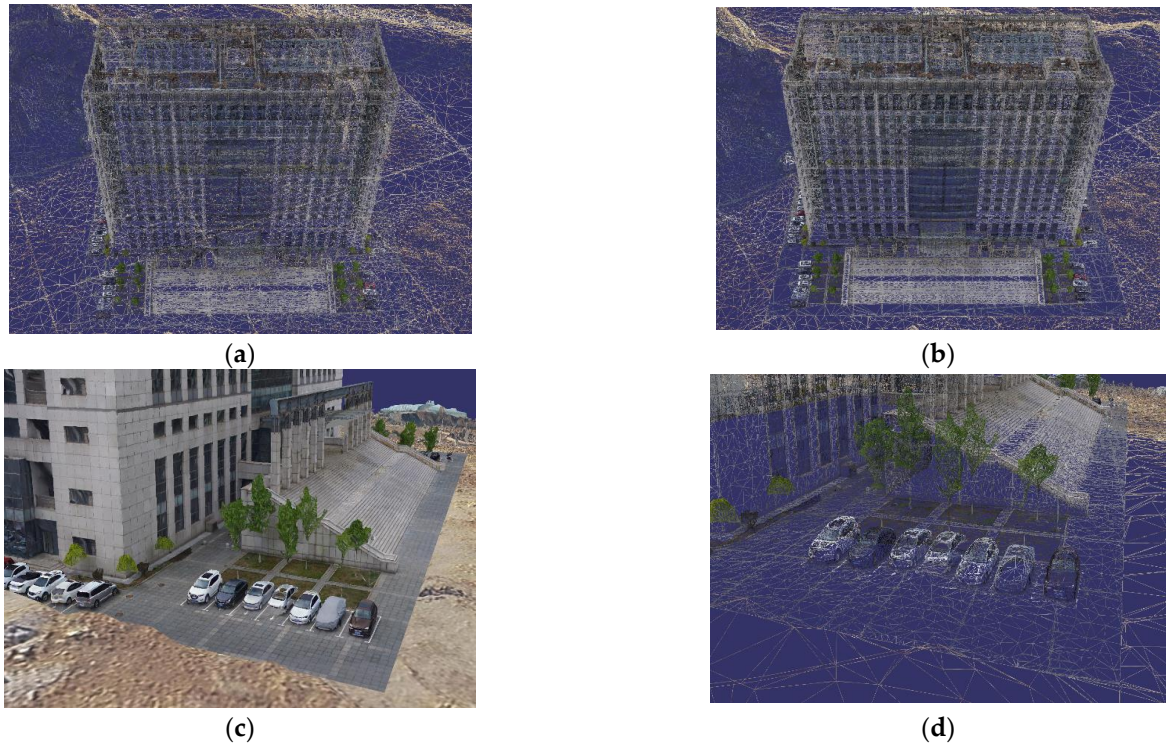


Figure 14. Model integration of the building: (**a**) mesh mode before integration; (**b**) mesh mode after integration; (**c**) details of the model after integration; (**d**) details of the mesh after integration.

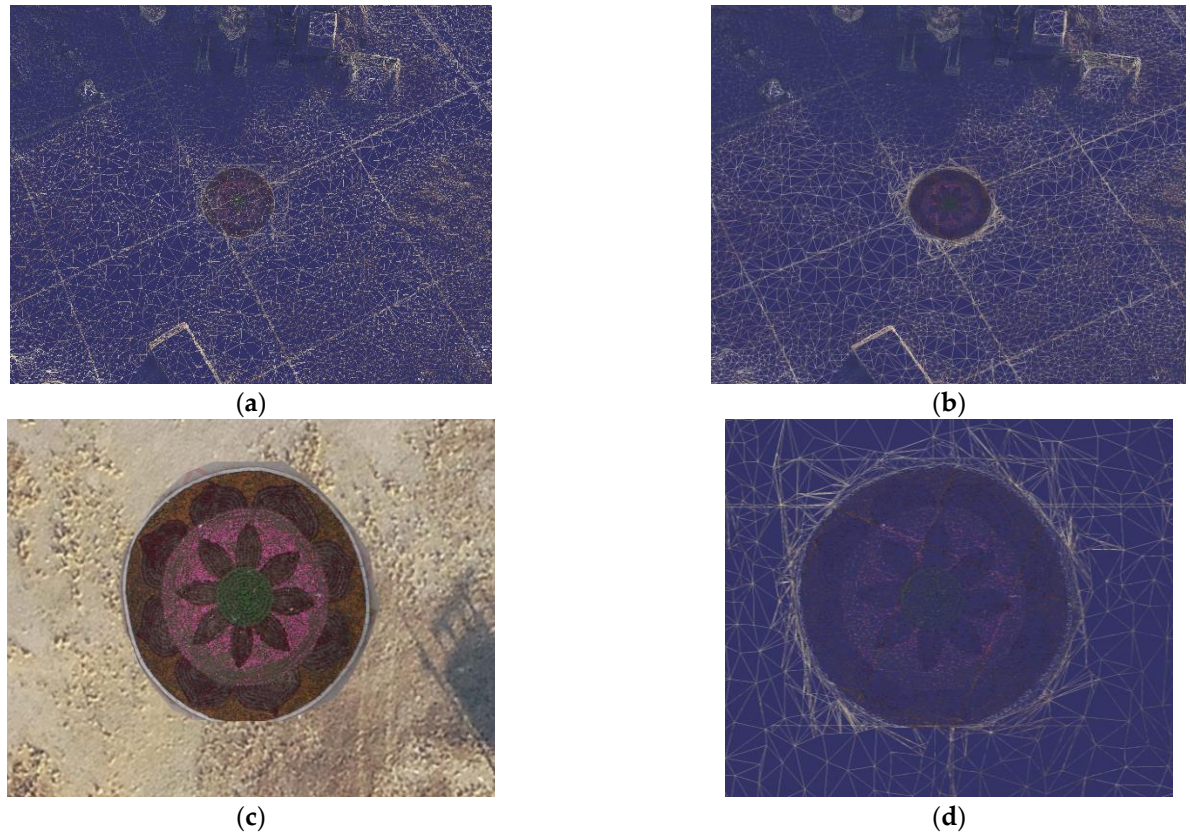
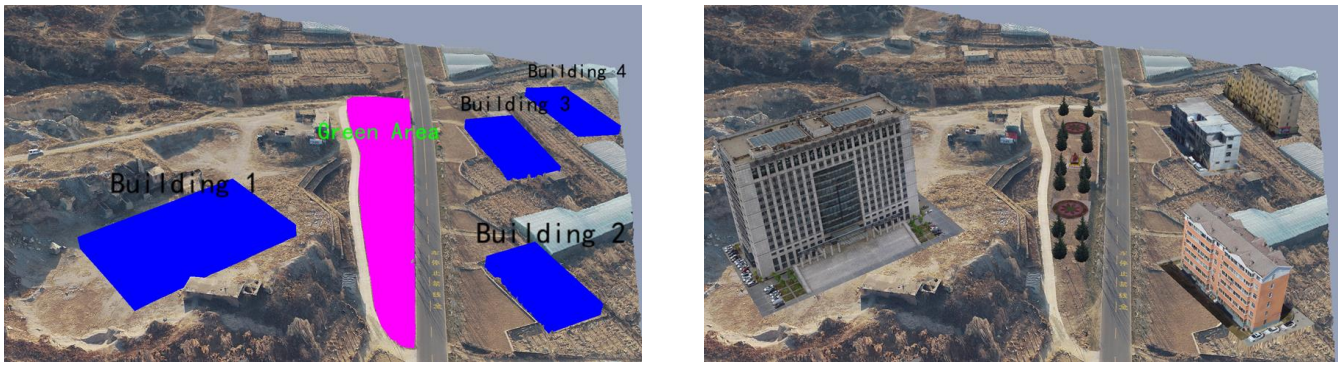


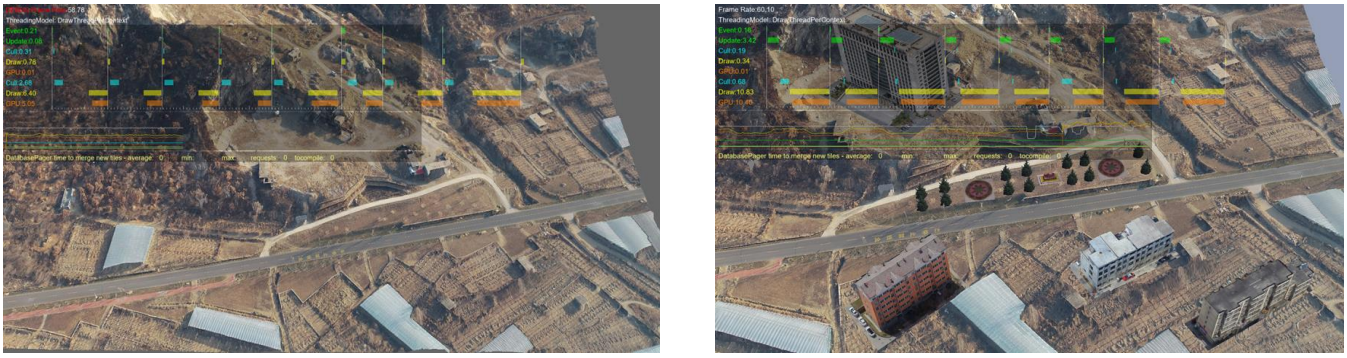
Figure 15. Model integration of circular garden: (**a**) mesh mode before integration; (**b**) mesh mode after integration; (**c**) details of the model after integration; (**d**) details of the mesh after integration.



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

Figure 16. Regional planning: (**a**) regional planning based on the original terrain model; (**b**) update effect after model integration.

The model integration in this paper was carried out in the display process, which can improve the efficiency of model updating to a certain extent. The average frame rate of browsing before integration was about 60 fps, and that of browsing after integration was between 45 and 60 fps, as shown in Figure 17. This fluency can be used by designers to browse, view, or design.



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

Figure 17. Average frame rate of browsing: (**a**) average frame rate before model integration; (**b**) average frame after model integration.

4. Conclusions

This paper proposes a 3D triangle mesh integration method for oblique photography model data. Firstly, one must load the replacement model to the replacement location, draw the boundary line on the model and project it. Then, it is necessary to project the triangle meshes that need to be cut, traverse the projected triangle meshes, and use the cutting line to cut and reconstruct the triangle meshes of the original terrain model and replacement model. At the same time, the texture image should be cut based on coordinate information, and the cut triangle meshes and texture image should be restored back to the initial 3D space to complete the cut and reconstruction of the 3D model. Afterwards, all pending files should be obtained through AABB collision detection, and threads should be allocated. The original terrain model and the replacement model were integrated into the same coordinate system, ultimately completing the integration of the oblique photography model. The experimental results show that this method effectively achieves the integration of 3D models in oblique photography, and there are no gaps in the 3D model after integration. This method can be widely used in fields such as 3D model repair in oblique photography.

Author Contributions: Conceptualization, D.C., M.S. and F.C.; data curation, D.C., M.S., B.M. and D.W.; formal analysis, F.C., Y.L. and B.M.; funding acquisition, D.C.; investigation, M.S. and F.C.; methodology, M.S.; project administration, D.C.; resources, B.M., D.W. and Y.S.; software, M.S. and F.C.; supervision, D.C. and B.M.; validation, F.C. and Y.L.; visualization, M.S.; writing—original draft, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted with support from the National Natural Science Foundation of China (Grant No. 41871310), and the Fundamental Research Funds for the Central Universities (Grant Nos. N17241004 and N2201007).

Data Availability Statement: The data presented in this study are available upon reasonable re-quest from the corresponding author.

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

References

1. Li, D.R.; Shao, Z.F.; Yang, X.M. Theory and practice from digital city to smart city. *Geospat. Inf.* **2011**, *9*, 2.
2. Song, Y.-S. Cultural assets reconstruction using efficient 3D-positioning method for tourism geographic information system. *J. Tour. Leis. Res.* **2010**, *22*, 97–111.
3. Leng, X.; Liu, D.; Luo, J.; Mei, Z. Research on a 3D geological disaster monitoring platform based on REST service. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 226. [CrossRef]
4. Singla, J.G.; Padia, K. A novel approach for generation and visualization of virtual 3D city model using open source libraries. *J. Indian Soc. Remote Sens.* **2021**, *49*, 1239–1244. [CrossRef]
5. Mademlis, I.; Mygdalis, V.; Nikolaidis, N.; Montagnuolo, M.; Negro, F.; Messina, A.; Pitas, I. High-level multiple-UAV cinematography tools for covering outdoor events. *IEEE Trans. Broadcast.* **2019**, *65*, 627–635. [CrossRef]
6. Zhang, Z.X.; Zhang, J.Q. Solutions and core techniques of city modeling. *World Sci. Technol. RD* **2003**, *25*, 7.
7. Wu, Y.Q.; Zhang, J.X. Application value of three-dimensional UAV real scene modeling technology in mine deep and peripheral geological situation exploration. *World Nonferrous Met.* **2022**, *14*, 113–115.
8. Murtiyoso, A.; Remondino, F.; Rupnik, E.; Nex, F.; Grussenmeyer, P. Oblique aerial photography tool for building inspection and damage assessment. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *40*, 309–313. [CrossRef]
9. Garland, M.; Willmott, A.; Heckbert, P.S. Hierarchical face clustering on polygonal surfaces. In Proceedings of the 2001 Symposium on Interactive 3D Graphics, Chapel Hill, NC, USA, 26–29 March 2001; Association for Computing Machinery: New York, NY, USA, 2001; pp. 49–58.
10. Sander, P.V.; Wood, Z.J.; Gortler, S.; Snyder, J.; Hoppe, H. Multi-chart geometry images. In Proceedings of the Eurographics Symposium on Geometry Processing, Aachen, Germany, 23–25 June 2003; 10p.
11. Lévy, B.; Petitjean, S.; Ray, N.; Maillot, J. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. TOG* **2002**, *21*, 362–371. [CrossRef]
12. Paulsen, R.R.; Bærentzen, J.A.; Larsen, R. Markov random field surface reconstruction. *IEEE Trans. Vis. Comput. Graph.* **2009**, *16*, 636–646. [CrossRef]
13. Wang, X.; Liu, X.; Lu, L.; Li, B.; Cao, J.; Yin, B.; Shi, X. Automatic hole-filling of CAD models with feature-preserving. *Comput. Graph.* **2012**, *36*, 101–110. [CrossRef]
14. Li, Z.; Meek, D.S.; Walton, D.J. Polynomial blending in a mesh hole-filling application. *Comput. Aided Des.* **2010**, *42*, 340–349. [CrossRef]
15. Liu, S.; Wang, C.C. Quasi-interpolation for surface reconstruction from scattered data with radial basis function. *Comput. Aided Geom. Des.* **2012**, *29*, 435–447. [CrossRef]
16. Lindstrom, P.; Koller, D.; Ribarsky, W.; Hodges, L.F.; Faust, N.; Turner, G.A. Real-time, continuous level of detail rendering of height fields. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 4–9 August 1996; pp. 109–118.
17. Hoppe, H. View-dependent refinement of progressive meshes. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 3–8 August 1997; pp. 189–198.
18. Ye, K.; Ma, X.Y.; Shi, L. A monomorphic method of oblique photography model based on 3D scene. In Proceedings of the 3rd International Conference on Green Materials and Environmental Engineering (GMEE), Beijing, China, 22–23 October 2017; pp. 93–97.
19. Schneider, M.; Klein, R. Efficient and accurate rendering of vector data on virtual landscapes. *J. WSCG* **2007**, *15*, 59–64.
20. Agugiaro, G.; Kolbe, T.H. A deterministic method to integrate triangular meshes of different resolution. *ISPRS J. PhotoGrammetry Remote Sens.* **2012**, *71*, 96–109. [CrossRef]
21. Xie, X.; Xu, W.; Zhu, Q.; Zhang, Y.; Du, Z. Integration method of TINs and GRIDs for multi-resolution surface modeling. *Geo-Spat. Inf. Sci.* **2013**, *16*, 61–68. [CrossRef]
22. Geng, Z.Y.; Ren, N.; Li, Y.C.; Xiao, J.C. Research on the fusion method of 3D models of oblique photography and large scene terrain. *Sci. Surv. Mapp.* **2016**, *41*, 108–113.

23. Noardo, F. Multisource spatial data integration for use cases applications. *Trans. GIS* **2022**, *26*, 2874–2913. [CrossRef]
24. Livny, Y.; Kogan, Z.; El-Sana, J. Seamless patches for GPU-based terrain rendering. *Vis. Comput.* **2009**, *25*, 197–208. [CrossRef]
25. Jung, Y.; So, B.; Lee, K.; Hwang, S. Swept volume with self-intersection for five-axis ball-end milling. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2003**, *217*, 1173–1178. [CrossRef]
26. Temkin, I.; Myaskov, A.; Deryabin, S.; Konov, I.; Ivannikov, A. Design of a digital 3D model of transport–technological environment of open-pit mines based on the common use of telemetric and geospatial information. *Sensors* **2021**, *21*, 6277. [CrossRef] [PubMed]
27. Zhao, Z. Research on 3D digital map system and key technology. *Procedia Environ. Sci.* **2012**, *12*, 514–520.
28. Lee, G.; Lee, S.; Kwon, S. A study on compression of 3D model data and optimization of website. *J. Eng. Appl. Sci. JEAS* **2019**, *14*, 3934–3937. [CrossRef]
29. Yoon, G.-J.; Song, J.; Hong, Y.-J.; Yoon, S.M. Single image based three-dimensional scene reconstruction using semantic and geometric priors. *Neural Process. Lett.* **2022**, *54*, 3679–3694. [CrossRef]
30. Yuan, Z.; Li, Y.; Tang, S.; Li, M.; Guo, R.; Wang, W. A survey on indoor 3D modeling and applications via RGB-D devices. *Front. Inf. Technol. Electron. Eng.* **2021**, *22*, 815–826. [CrossRef]
31. Fu, Y.; Yan, Q.; Liao, J.; Zhou, H.; Tang, J.; Xiao, C. Seamless texture optimization for RGB-D reconstruction. *IEEE Trans. Vis. Comput. Graph.* **2023**, *29*, 1845–1859. [CrossRef]
32. Ruan, M.; Xie, M.; Liao, H.; Wang, F. Studying on conversion of oblique photogrammetry data based on OSG and ARX. *Proc. IOP Conf. Ser. Earth Environ. Sci.* **2019**, *310*, 032062.
33. Hughes, J.F.; Foley, J.D. *Computer Graphics: Principles and Practice*; Pearson Education: London, UK, 2014.
34. Ericson, C. *Real-Time Collision Detection*; CRC Press: Boca Raton, FL, USA, 2004.

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.