remote sensing

Article

A Model Simplification Algorithm for 3D Reconstruction

Zhendong Liu [1], Chengcheng Zhang [1,*], Haolin Cai [2], Wenhu Qv [1] and Shuaizhe Zhang [2]

[1] Chinese Academy of Surveying and Mapping, Beijing 100036, China
[2] College of Geodesy and Geomatics, Shandong University of Science and Technology, Qingdao 266590, China
* Correspondence: zhangcc@casm.ac.cn; Tel.: +86-010-6388-0555

Abstract: Mesh simplification is an effective way to solve the contradiction between 3D models and limited transmission bandwidth and smooth model rendering. The existing mesh simplification algorithms usually have problems of texture distortion, deformation of different degrees, and no texture simplification. In this paper, a model simplification algorithm suitable for 3D reconstruction is proposed by taking full advantage of the recovered 3D scene structure and calibrated images. First, the reference 3D model scene is constructed on the basis of the original mesh; second, the images are collected on the basis of the reference 3D model scene; then, the mesh and texture are simplified by using the reference image set combined with the QEM algorithm. Lastly, the 3D model data of a town in Tengzhou are used for experimental verification. The results show that the algorithm proposed in this paper basically has no texture distortion and deformation problems in texture simplification and can effectively reduce the amount of texture data, with good feasibility.

Keywords: mesh simplification; texture simplification; reconstruction pipeline; camera poses; calibrated images

Citation: Liu, Z.; Zhang, C.; Cai, H.; Qv, W.; Zhang, S. A Model Simplification Algorithm for 3D Reconstruction. *Remote Sens.* **2022**, *14*, 4216. https://doi.org/10.3390/rs14174216

Academic Editor: Mohammad Awrangjeb

Received: 14 July 2022
Accepted: 25 August 2022
Published: 26 August 2022

1. Introduction

Oblique aerial photography obtains rich high-resolution textures of the top surface and side view of buildings by synchronously acquiring images from one vertical, four oblique, and five different angles. Traditional aerial photography only obtains information from the vertical direction and can be used for the production of 4D products, but it cannot meet the modeling needs of urban areas. Oblique aerial photography can not only truly reflect the situation of ground objects and obtain their texture information with high accuracy but also generate a real 3D urban model through advanced positioning (such as structure form motion [1]), multisource data fusion (such as LIDAR point cloud data [2]), modeling, and other technologies. Oblique aerial photography has been widely used in large-scale mapping, 3D model texture acquisition, and rapid 3D modeling of digital cities [3–5]. However, with the enlargement of the reconstruction area and the continuous improvement in the modeling accuracy, the contradiction between the rapidly growing 3D model data size and the limited transmission bandwidth and smooth model rendering has been intensified. Mesh simplification is an effective solution [6].

The mesh simplification technology transforms the high-resolution and accurate original 3D model into a coarser approximate mesh model [7]. At present, many researchers have made progress in this field, and the algorithms can be roughly divided into the following two categories: geometry-driven simplification algorithms and appearance attribute-driven simplification algorithms [8]. Common geometry-driven simplification algorithms include the algorithm based on vertex clustering [9], the algorithm based on vertex extraction [10], and the algorithm based on edge folding [11,12]. This type of algorithm only relies on the geometry in the cost metric and strives to ensure geometric fidelity on the premise of reducing vertices while ignoring appearance attributes such as color and texture. Simplification is not ideal when faced with 3D models with appearance attributes such as textures. The appearance attribute-driven simplification algorithm considers not only geometry but also appearance attributes, especially textures, to better ensure the simplified appearance. Among them, Garland and Heckbert extended their original algorithm by incorporating appearance attributes into the quadratic error metric (QEM) [7,13]. Sporysz et al. [14] performed Canny edge detection on texture maps and considered the ratio of the area of the simplified area to the total area of the 3D mesh, effectively preserving the edge information in the mesh. She et al. [15] segmented the surface mesh according to the topology and appearance and derived an error metric considering geometry and texture, which minimized the texture distortion. However, the above methods only take model simplification as the postprocessing step of 3D reconstruction and do not consider close integration with the 3D reconstruction pipeline. Moreover, the model after simplified processing using the above method has a certain degree of distortion in texture, and the texture is not simplified, resulting in a large amount of model data.

Therefore, a mesh and texture high-fidelity 3D model simplification algorithm must be explored. In this paper, on the basis of the current mainstream QEM algorithm, a model simplification algorithm for 3D reconstruction is proposed with view pose and calibrated images in the process of 3D reconstruction. The algorithm first constructs the energy function and selects the best view for each facet, completes the automatic texture mapping of the original 3D mesh, and constructs the reference 3D model scene with texture. Then, the reference 3D model scene is back-projected by the view pose to generate a reference image set, which is used to remove the inconsistency of the time dimension in the real image. Lastly, the QEM algorithm is used to simplify the mesh, and the reference image set is used as the data source of texture mapping to automatically simplify the texture of the simplified mesh and map.

The main contributions of this paper are as follows:

(1) The process of 3D model simplification and reconstruction of oblique photography is comprehensively regarded rather than a postprocessing step. In the process of model simplification, the scene structure recovered in the reconstruction process (internal and external parameters of view) and the calibrated image information are fully utilized;
(2) A mesh simplification method that considers texture fidelity is proposed. On the basis of the reference 3D model scene, this method uses the projection raster principle and texture reconstruction method to perform texture remapping of the simplified mesh and avoids texture deformation and distortion;
(3) A texture content simplification method is proposed, and the texture simplification parameters are adaptively calculated according to the QEM mesh simplification parameters to downsample the reference image and use it as the data source of the texture reconstruction method to achieve the purpose of texture simplification.

The components of this paper are as follows: the existing appearance attribute-driven simplification algorithms and their shortcomings are introduced in Section 2; the mesh texture simplification algorithm for 3D reconstruction is elaborated in Section 3; the experiments and results analysis are presented in Section 4; the discussion and conclusions of this algorithm are introduced in Section 5.

2. Related Works

The existing mesh simplification algorithms can be roughly divided into two categories: geometry-driven and appearance attribute-driven. The algorithms in this paper belong to the second category; hence, our discussion of earlier work focuses on appearance attribute-driven simplification algorithms.

2.1. Existing Appearance Attribute-Driven Simplification Algorithms

To better maintain the appearance of the simplified model, appearance attribute-driven simplification algorithms simultaneously consider the geometry and appearance attributes, particularly texture, during the simplification process. Most algorithms use iterative edge collapse to simplify the mesh and determine the texture coordinates of each replaced vertex [15] to reduce the degree of texture distortion. Garland and Heckbert [13] extended their original algorithm by merging attributes into a quadratic error measure (QEM) [7]. Cohen et al. [16] further improved their work by introducing a texture deviation measure and locally finding the texture coordinates of new vertices as a function of this measure. On the basis of the perception model, Williams et al. [17] prioritized edge dimension reduction by considering texture deviation, lighting contrast, dynamic lighting, and other factors.

Other algorithms use texture images or rendered images during mesh simplification. Lindstrom and Turk (2000) proposed an image-driven simplification method that compares the rendered images of the pre- and post-simplification models from multiple viewpoints, calculates the root-mean-square error (RMSE) of the image pixels, and then sorts the edge collapse operation according to the RMSE. Qu and Meyer (2008) analyzed the perceptual properties of surface signals (e.g., texture, color, and light) and used the results to calculate the importance values for each vertex of the model and integrate them into the QEM to guide the simplification process [18]. In addition, Pascual et al. (2008) and González et al. (2013) proposed a simplified algorithm based on viewpoint entropy and mutual information, which could also reduce the texture distortion of the simplified model [19,20].

In the model simplification with texture, other appearance-preserving strategies are also proposed. García and Patow [21] proposed a texture technique named inverse geometric texture (IGT), which defines texture coordinates for all vertices in the simplified model to preserve the texture details in the high-resolution reference model. Chen and Chuang [22] and Coll and Paradinas [23] modified the texture image to minimize the texture distortion that is caused by dimension reduction of each edge. Notably, the methods of García and Patow [21], Chen and Chuang [22], and Coll and Paradinas [23] are not applicable to embedded level of detail (LOD) construction because the texture coordinates of inherited vertices or the content of texture images are constantly changing during the iterative simplification process.

In terms of alleviating the texture distortion and deformation of simplified models, the latest and most effective methods in existing research are derived from She (2019) [15], who proposed a new simplified method for complex 3D building models, achieving a good balance between geometric fidelity and texture maintenance. The basic principle of this method is as follows:

Step 1: Mesh segmentation. On the basis of breadth first search (BFS), the surface mesh is divided into multiple regions according to the geometric similarity and texture features of the mesh. On the basis of the segmentation results, each edge of the model is assigned a weight so that more simplification operations can be carried out in the same region.

Step 2: Cost calculation and half-edge collapse. The mesh is traversed to calculate the initial cost of all edges, and the edge with the lowest cost is selected as the starting point of half-edge collapse simplification. The texture information in the model is fully considered in the simplification.

Step 3: Texture coordinate update. The texture coordinates are adjusted, and the cost of adjacent edges is updated. This method terminates until the simplification rate exceeds a specified threshold (the simplification rate is defined as the number of triangles that are removed divided by the number of triangles that are in the original model).

2.2. Deficiencies of Existing Methods

The existing algorithms are well developed for use in mesh simplification. However, in texture simplification, the current algorithms only consider the calculations and updates of the texture coordinates that correspond to the mesh after simplification, which can only alleviate texture distortion and deformation to a certain extent but not completely solve the problem. In addition, the algorithm does not simplify the texture content, resulting in a large amount of model data after simplification. Two aspects are described in detail below.

1. Texture distortion and deformation after mesh simplification

The mesh simplification algorithm performs mesh vertex deletion, edge collapse, and face merging on the adjacent facets of 3D space according to certain merging rules (such as the quadratic metric formula). However, because the texture patches corresponding to the merged facets are not necessarily continuous in the 2D texture space, this discrete distribution phenomenon leads to the failure of correctly calculating the texture coordinates of the newly generated triangular facets. This is explained in combination with Figures 1 and 2.

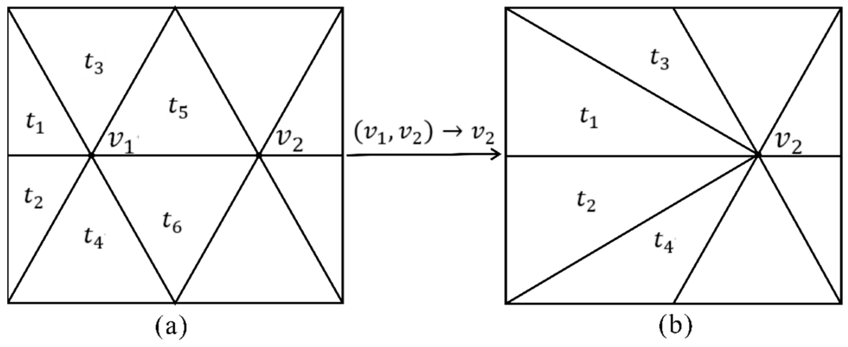

Figure 1. Schematic diagram of the mesh simplification, in which vertex v_1 is deleted, vertex v_2 replaces v_1 as the vertex of triangular patches t_1–t_4, and triangular patch t_5 is deleted. (**a**) Unsimplified mesh. (**b**) Simplified mesh.

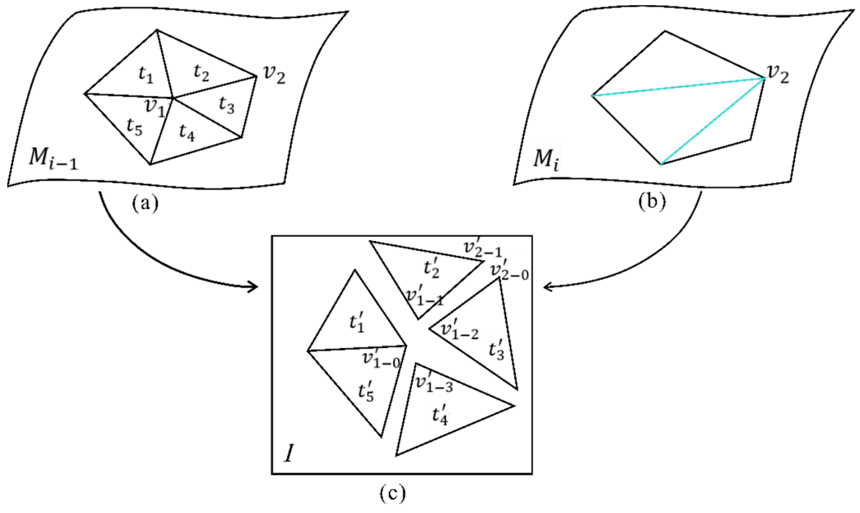

Figure 2. Mapping relationship between meshes and textures. (**a**) $M_{\mathrm{i}\ -1}$ is the original mesh in 3D space, and t_1–t_5 are five adjacent facets. (**b**) M_i is a simplified mesh after the half-edge collapse $(v_1, v_2) \rightarrow v_2$. (**c**) I represents the 2D texture space, $t'_1\ \sim -$ is the patch corresponding to t_1–t_5 in the texture space, and v'_{1-0}–v'_{1-3} and v'_{2-0}–v'_{2-1} are the corresponding points of v_1 and v_2 in I, respectively.

According to the existing algorithm [15], the original mesh $M_{(i-1)}$ is merged into a simplified mesh M_i, and the texture distortion and deformation of the model are taken into account during the merging process. The error metric formula constructed in Step 2 of the algorithm tends to merge consecutive texture patches, e.g., t'_1 and t'_5 in Figure 2c. For discrete texture patches, e.g., t'_2, t'_3, and t'_4 in Figure 2c, the algorithm collects multiple line segment groups before and after collapse. If there is no intersection between the line segment pairs, the calculation of the texture error formula fails and cannot handle such situations. Thus, this algorithm has certain limitations.

At the same time, the method of adjusting the texture coordinates in Step 3 of the existing algorithm is to search the nearest texture coordinate value of the texture space for the approximate replacement according to the texture coordinate mapping table established by the mesh vertices before and after simplification. The texture coordinate tex_2 corresponding to the vertex v_2 of facet t_3 in Figure 1b is replaced by the texture coordinate tex_2 corresponding to the vertex v_2 of facet t_5 in Figure 1a. Notably, the two tex_2 here may not be the same coordinate in the texture space. Therefore, the texture patch of facet t_3 in Figure 1b is likely to be distorted or deformed.

2. Texture content simplification

Since the existing algorithms directly take the original texture as the texture of the simplified mesh, only the texture coordinates of the relevant vertices are adjusted and updated. Generally, in 3D models composed of mesh and texture, the texture accounts for a large proportion of the entire model data. Therefore, the data size of the simplified model is still large.

3. Methodology

On the basis of the most widely used and most effective QEM algorithm, which is combined with the scene structure (internal and external parameters of the view) recovered in the 3D reconstruction pipeline and the calibrated images, a model (mesh & texture) simplification algorithm for 3D reconstruction is proposed in this paper. The core content includes the following three parts:

(1) The reference 3D model scene construction: The original fine 3D mesh was reconstructed to create the reference 3D model scene using the information of the scene structure that was recovered in the 3D reconstruction and calibrated image.
(2) Image acquisition of the reference 3D model scene: According to the relative pose relationship in 3D space between the reference 3D model scene and the internal and external parameters of the view, the rasterization calculation from the 3D mesh to the 2D image was carried out using the principle of back-projection, and the reference image set was collected.
(3) Mesh and texture simplification: The mesh was simplified using the QEM, the reference image set was used as the data source, and the texture was remapped and simplified using a texture reconstruction algorithm. The flowchart of this method is shown in Figure 3.

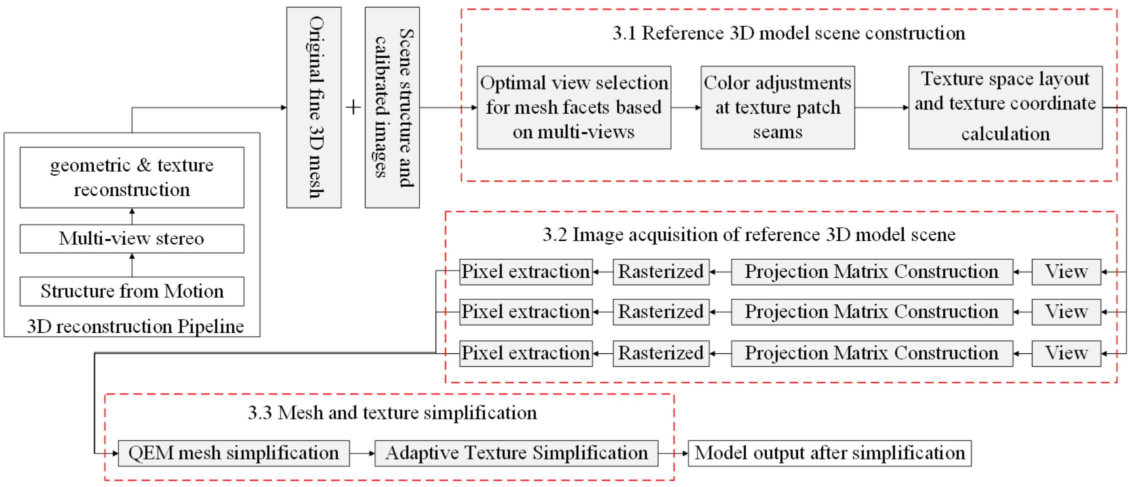
3.1 Reference 3D model scene construction
Optimal view selection for mesh facets based on multi-views
Color adjustments at texture patch seams
Texture space layout and texture coordinate calculation
Original fine 3D mesh
+
Scene structure and calibrated images
geometric & texture reconstruction
Multi-view stereo
Structure from Motion
3D reconstruction Pipeline
3.2 Image acquisition of reference 3D model scene
Pixel extraction
Rasterized
Projection Matrix Construction
View
Pixel extraction
Rasterized
Projection Matrix Construction
View
Pixel extraction
Rasterized
Projection Matrix Construction
View
3.3 Mesh and texture simplification
QEM mesh simplification
Adaptive Texture Simplification
Model output after simplification

Figure 3. Flowchart of this method.

3.1. *Reference 3D Model Scene Construction*

According to the texture reconstruction method proposed by Waechter et al. (2014) and Bi et al. (2017) [24,25], the original fine 3D mesh is automatically textured to complete the reconstruction of the reference 3D model scene. The detailed steps are described below.

1. Optimal view selection for mesh facets based on multi-view images

In essence, this step is composed of calculating a label for each facet to record the most suitable image number for the texture mapping of this facet. The Waechter method (2014) uses Soble gradient integration in the data item to solve the problem of image defocusing and uses the Potts model in the smoothing item to smooth the view selection of the neighborhood triangulation. The energy function formula is as follows:

$$E(l) = \sum_{F_i \in Faces} E_{data}(f_i, l_i) + \sum_{(F_i,F_j) \in Edges} E_{smooth}(f_i, f_i, l_i, l_j), \tag{1}$$

where $E_{data}(f_i, l_i) = \sum_{j=1}^{N} Grad_{ij}$; $Grad_{ij}$ is the Soble gradient integration of the triangular mesh f_i in the labeled image l_i, which represents the probability that the node f_i selects a certain label image l_i. $E_{smooth}(f_i, f_j, l_i, l_j) = \begin{cases} 0 \; l_i = l_j \\ \infty \; l_i \neq l_j \end{cases}$ means that, when adjacent nodes f_i and f_j select the same label image, the value of the smooth term is 0; otherwise, it is infinite. The smooth term E_{smooth} minimizes the visibility of seams (i.e., edges between textures in different images). $E(l)$ is minimized by graph cuts and alpha expansion [25].

As a data item, E_{data} considers the following two aspects: summation of all pixels in the view gradient magnitude map projected by the facet F_i and image consistency detection. Among them, the formula for the sum of the pixels in the gradient magnitude projected by the facet F_i is $-\int_{\phi(F_i,l_i)} \| \nabla(I_{l_i}(p)) \|_2 dp$, and the gradient magnitude is $\| \nabla(I_{l_i}) \|_2$. Image consistency detection mainly uses a modified mean-shift algorithm to try to remove images containing occluders such as pedestrians.

As a smoothing term, E_{smooth} is the difference between the textures on the left and right sides of the gap. According to the Potts model, a smoothing term is proposed: $E_{smooth} = [l_i \neq l_j]$ ($[\cdot]$ is the Iverson bracket). This model prefers to use compact patches over distant views and is very fast to compute.

The optimal view of each facet in the mesh is obtained by minimizing the energy function by graph cut [26]. A texture patch is created, and the spatially adjacent patches with the same optimal view are stored in the same texture patch; that is, $texturePatch = (\{Tri_j \ldots Tri_n\}, l_i)$. Here, $texturePatch$ refers to a texture patch, Tri_j represents a facet belonging to the current texture patch, and l_i represents the optimal view corresponding to the current texture patch.

2. Color adjustment between mesh facets

After choosing the optimal view for a facet, adjacent facets may choose different views as the source of the texture. Due to objective factors such as the shooting angle, light, and occlusion during image acquisition, the pixels near the seams of the adjacent facets are different, and adjusting the color between the adjacent facets is necessary. Color adjustment includes the following two parts: the global color adjustment and the local color adjustment of the Poisson editing.

The global color adjustment first looks for inconsistent color values at vertex projections and along all adjacent seam edges, and then uses a weighted average to alleviate the inconsistencies. From the global perspective, the algorithm calculates an accumulated correction for the color value of each vertex, and its formula is expressed in matrix form as follows:

$$\| \mathbf{Ag} - \mathbf{f} \|_2^2 \| + \mathbf{\Gamma g} \|_2^2 = \mathbf{g}^{\mathrm{T}} \left(\mathbf{A}^{\mathrm{T}}\mathbf{A} + \mathbf{\Gamma}^{\mathrm{T}}\mathbf{\Gamma} \right) \mathbf{g} - 2\mathbf{f}^{\mathrm{T}}\mathbf{Ag} + \mathbf{f}^{\mathrm{T}}\mathbf{f}. \tag{2}$$

The specific explanation of the matrix is not repeated here; these details are provided in [20].

After global adjustment, the algorithm still cannot eliminate all visible seam color differences. Therefore, the local adjustment of the Poisson editing was also performed in [20]. A 20 pixel wide border bar was set in the algorithm, and the outer edge (Figure 4, light blue) and inner edge (Figure 4, red) of this strip were used as boundary conditions of the Poisson equation; the mean value of the pixel color of the image assigned to the patch and the neighbor patch was set to a fixed value of the color value of each outer edge pixel.

The value of each inner edge pixel is fixed to its current color. If the patch is too small, the inner edge is omitted.

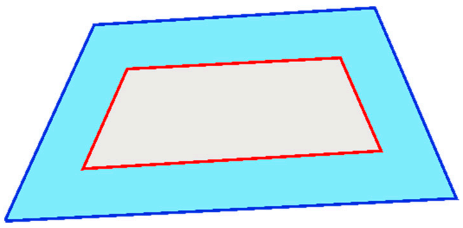

Figure 4. Texture patch with border bars (light blue for outer edge and red for inner edge).

3. Texture space layout and pixel extraction

After color adjustment, the proposed method calculates the layout of all texture patches in texture space and completes the extraction task of pixel values (generally RGB). The specific steps are described below.

Step 1: The view projection matrix (as shown in Equation (3)) is used to project all patches $\{Tri_j \ldots Tri_n\}$ stored in the texture patch *texturePatch* onto the view space (as shown in the upper right corner of Figure 5a), and the bounding box $box2D_{view}$ of *texturePatch* in the view space is calculated. All texture patches are processed sequentially according to the above method.

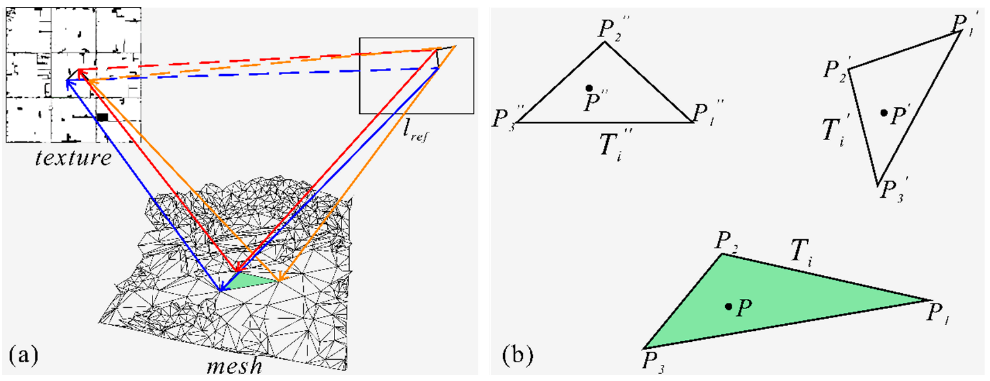

Figure 5. Schematic diagram of the transformation relationship among the 3D mesh, view, and texture. (**a**) Whole drawing (the bottom is the mesh, the top left is the texture, and the top right is the view reference image). (**b**) Enlarged drawing of three spatial relationships.

Step 2: A texture image without any pixel values is initialized to store the pixel values (generally RGB) of the reference 3D model scene. The space of this texture image is named the texture space. The 2D box packing algorithm in [27] is used to arrange the shape of all texture patches in texture space according to $box2D_{view}$, and the relative offset relationship $transform''(x'', y'')$ of each $box2D_{view}$ in texture space is calculated.

Step 3: The coordinate values $t'(x', y') = \{p_1'(x_1', y_1'), p_2'(x_2', y_2'), p_3'(x_3', y_3')\}$ of the three vertices contained in each facet in the texture patch in view space are calculated, and then the calculation formula of the texture coordinate values of the three vertices is equal to $\{p_1''(x_1'', y_1''), p_2''(x_2'', y_2''), p_3''(x_3'', y_3'')\} = transform''(x'', y'') + t'$. According to the coordinate value $t'(x', y')$ of the three vertices contained in the facet in the view space, pixel data are extracted from the image of the best view l_i of the texture patch to fill the texture image of the reference 3D model scene.

3.2. Image Acquisition of the Reference 3D Model Scene

To facilitate the description of the algorithm in this paper, the following provisions are made: the image set generated by "photographing" the reference 3D model scene with the help of the recovered view poses (the camera frustum represented by the red point and the yellow wire frame in Figure 6) is called the reference image set $\mathbf{IR} = \{imageRef_0 \ldots\ldots imageRef_i\}$; the image set generated by "photographing" the real objective world by UAVs (unmanned aerial vehicles), mobile terminals, and other devices is called the real image set $\mathbf{I} = \{image_0 \ldots\ldots image_i\}$.

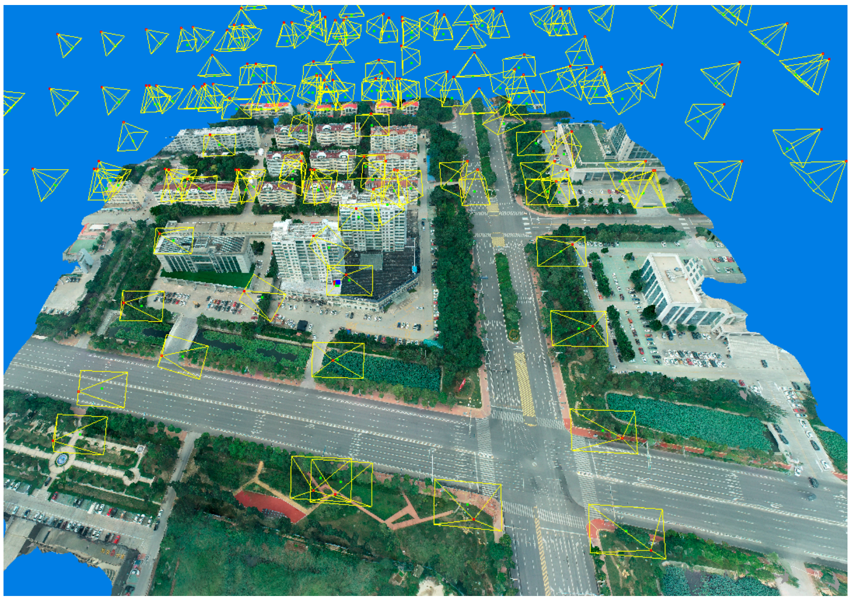

Figure 6. The 3D model and scene structure.

If the real image set $\mathbf{I} = \{image_0 \ldots\ldots image_i\}$ is directly used as the data source of the simplified mesh texture reconstruction in Step 3, the texture pixel content of the original mesh and the simplified mesh will be inconsistent. The specific reasons are provided in the discussion below.

(1) The faces of the original mesh and the simplified mesh have different shapes and sizes in the same space, leading to different views as the optimal views of the original mesh and the simplified mesh in the first stage of texture reconstruction, as shown in Figures 7–9. The red box in Figures 7 and 8 represents the same spatial range. The texture reconstruction algorithm transforms each facet into a node of the graph and forms an energy function based on the visibility of the facet, gradient amplitude, image consistency detection, moving object elimination, and other factors. Then, global graph cutting on the graph is used to obtain the optimal view of the facet. That is, the optimal view IDs of the original mesh in Figure 7b are 48, 87, 115, and 88, while the optimal view ID of the simplified mesh in Figure 8b is only 41.

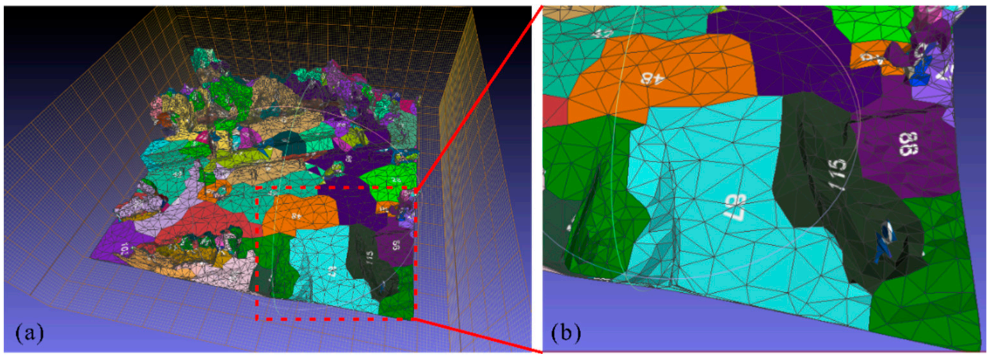

Figure 7. Best view IDs of the original mesh facets: (**a**) whole drawing; (**b**) partial enlarged drawing.

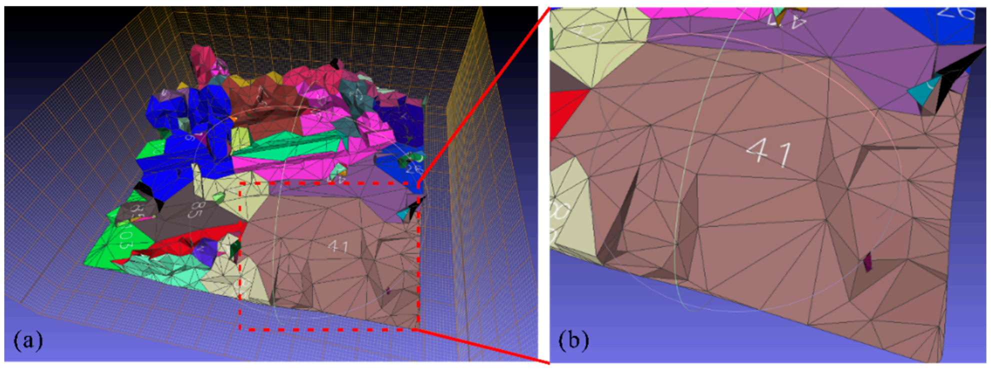

Figure 8. Best view ID of the simplified mesh facets: (**a**) whole drawing; (**b**) partial enlarged drawing.

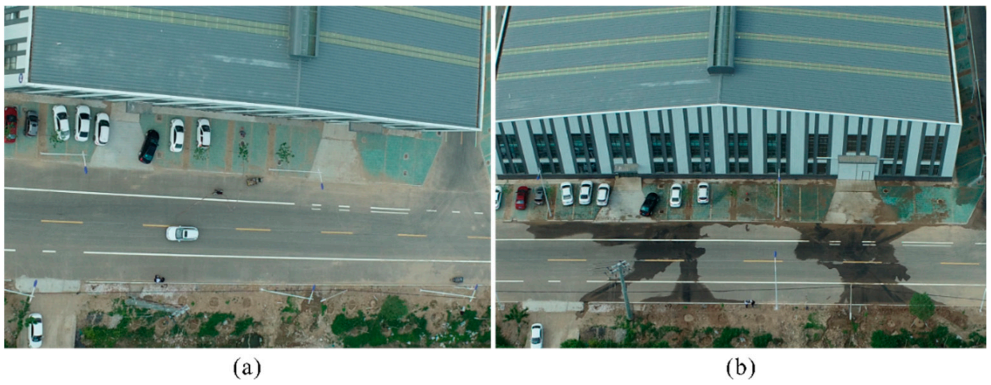

Figure 9. Images acquired at different times in the same area. (**a**) This image includes moving cars on the road and no water. (**b**) This image includes water on the road and no moving cars.

(2) Data collection by devices such as UAVs or mobile terminals is affected by the time dimension, and images collected at different times in the same area will contain different information such as vehicles, pedestrians, dynamic shadows, and changes in road wetting and dryness, as shown in Figure 9a,b. Therefore, the pixel content extracted from different images for the same facet may be different.

Considering the above two aspects and according to Reason (1) of the analysis, the original and simplified meshes in the red box will take different views as the optimal view of the facet. According to Reason (2), the pixel content of the facet extracted from different images is different because the image pixel content collected at different times in the same space is different. As a result, the textures have inconsistencies in pixel content.

To avoid the problem of inconsistent pixel content in the texture of the original and simplified meshes, the algorithm uses the solved camera internal parameters, view external parameters, calibrated images, and reference 3D model scenes to back-project the textures corresponding to all facets in the reference 3D model scene constructed in the previous section from the texture space to the view space and grids them. A reference image is generated for each view to form the reference image set. Combined with Figure 5 and Equations (3) and (4), the specific steps are explained below.

Step 1: According to the internal and external parameters of the view in the scene and the image resolution, the projection matrix of view l is constructed (Equation (3), where M_1 is the internal parameter matrix, and M_2 is the external parameter matrix), and the 3D points in the world coordinate system are transformed into pixels in the view image coordinate system. Then, an octree index is built for the reference 3D model scene to rapidly search the visible facets of view l.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} f/dx & 0 & \mu_0 & 0 \\ 0 & -f/dy & v_0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R_c & T_c \\ O_{1\times 3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \frac{1}{Z_c} M_1 M_2 \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{3}$$

Step 2: Combined with the camera projection matrix of view l, a depth map l_{depth} and a reference image l_{ref} with the same resolution $Re(width,\ height)$ as the view image are created. l_{depth} is in the camera coordinate system of view l, and the depth value is initialized to 0.

Step 3: The visible faces of view l are traversed. The face T_i is projected from 3D space to the 2D reference image l_{ref} using a camera projection matrix and rasterized into triangle T_i'.

Step 4: The occupied pixels of T_i' are counted, and the color values are extracted from the texture of the reference 3D model scene pixel by pixel to construct the data content of the reference image l_{ref}.

The view space pixel $p'(x', y')$ in Figure 5 is taken as an example for illustration, and the calculation process of the pixel color value is described in detail, where $p(x, y)$ and $p''\left(x_0'', y_0''\right)$ are the mappings of Point $p'(x', y')$ in the 3D space and the texture space, respectively.

First, the depth value $d'(x', y')$ of $p(x, y)$ in the view space is calculated and compared with the depth value $d_{depth}(x', y')$ in the depth map l_{depth}. If $d'(x', y')$ is less than $d_{depth}(x', y')$, then $d'(x', y')$ is stored in the depth map l_{depth}. Then, the texture coordinate $p''\left(x_0'', y_0''\right)$ is calculated using the back-projection between T_i' and T_i, T_i and T_i'' (the inverse process of Equation (3)) and the linear relationship between the texture coordinates $\left\{p_1''\left(x_1'', y_1''\right),\ p_2''\left(x_2'', y_2''\right),\ p_3''\left(x_3'', y_3''\right)\right\}$ of the three vertices of the facet, which is described by Equation (4). Lastly, the pixel color value (usually RGB) is extracted on the basis of the texture coordinate $p''\left(x_0'', y_0''\right)$ and filled to the pixel coordinate $p'(x', y')$ in the reference image l_{ref}.

$$\begin{cases} D = P'' - P_3'' \\ d_{00} = P_1''\left(x_1''\right) - P_3''\left(x_3''\right) \\ d_{01} = P_2''\left(x_2''\right) - P_3''\left(x_3''\right) \\ d_{10} = P_1''\left(y_1''\right) - P_3''\left(y_3''\right) \\ d_{11} = P_2''\left(y_2''\right) - P_3''\left(y_3''\right) \\ Det = d_{00} \times d_{11} - d_{10} \times d_{01} \end{cases} \Rightarrow \begin{cases} x_0'' = \frac{d_{11} * D(x'') - d_{01} * D(y'')}{Det} \\ y_0'' = \frac{d_{00} * D(y'') - d_{10} * D(x'')}{Det} \end{cases} \tag{4}$$

Step 5: All views are traversed and Steps 1–4 are repeated until the acquisition task of the reference image set is completed.

3.3. Mesh and Texture Simplification

The simplification of the 3D model can be divided into the following two aspects: mesh simplification and texture simplification. The mesh simplification is very mature at present. Similar to [15], the algorithm in this paper adopts the QEM algorithm, which is the most widely used and has the best effect. However, the texture simplification still has the problem of texture distortion and deformation, and the amount of model data is still large due to the unsimplified texture. The method proposed in this paper can effectively solve the above problems. The specific steps are described below.

Step 1: The simplification parameters α and $\alpha \in (0.0,\ 1.0]$ in the QEM algorithm are determined according to the actual requirements. A smaller α indicates a greater simplification degree. Vertex deletion and mergence, edge collapse, and facet deletion and mergence on the original fine 3D mesh are performed using the QEM algorithm.

Step 2: The texture simplification parameter β, i.e., the resolution sampling level of the reference image, is determined. The value of β can be independently determined or estimated according to the mesh simplification parameter α. The calculation formula proposed in this paper is $\beta = ceil(1/\alpha)$. Theoretically, the range of the β value is $\beta \in [1.0, +\infty)$; however, in practical applications, it is meaningless for the resolution level of the image to be infinite. Therefore, the commonly used range limit is set to $\beta \in [1.0,\ 20)$. In addition, the calculation formula of β can also be customized.

Step 3: The internal parameter matrix M_{s1}, external parameter matrix M_{s2}, and reference image resolution Re_s involved in the texture reconstruction algorithm are determined according to the β value. The calculation formula of the internal parameter matrix M_{s1} is as follows:

$$M_{s1} = M_1/(\beta \times Max(width,\ height)), \tag{5}$$

where M_1 is the camera internal parameter matrix used in Section 3.2, and *width* and *height* represent the original width and height of the view image, respectively. As the poses of the view are fixed, the external parameter matrix is unchanged, i.e., $M_{s2} = M_2$.

The calculation formula of the reference image resolution Re_s is

$$Re_s(width_s,\ height_s) = Re(width,\ height)/\beta. \tag{6}$$

The reference image is sampled using Re_s to obtain $\mathbf{IR_s} = \{imageRef_{0s} \ldots\ldots imageRef_{is}\}$.

Step 4: Taking the parameters, the image data, and the simplified mesh as input, the simplified mesh calculated in the above steps is textured according to the texture reconstruction algorithm steps described in Section 3.1.

4. Experiment and Results

4.1. Experimental Data and Environment

The method proposed in this paper was embedded into the IMS software, which is a reality modeling software that was independently developed by the authors at the Chinese Academy of Surveying and Mapping. Three-dimensional mesh model data reconstructed by oblique photography in the town of Tengzhou city, Shandong Province, China, were used for experimental verification. The detailed parameters of the experimental data are shown in Table 1, and the reconstruction scope is shown in Figure 10. The experimental data cover the main types of 3D models such as buildings, vegetation, and roads. The selection of this area is of general significance for experimental verification. The experimental running environment is a workstation with the Windows 10 64 bit operating system, an Intel Core (TM) I9-10900X CPU with a dominant frequency of 3.70 GHz, and 128 GB of memory.

Table 1. Experimental data information.

Data Type	Reconstruction Scope (km^2)	Ground Resolution (m)	Number of Images	Image Resolution	
				Vertical View	Oblique View
3D mesh	10.62	0.03	34364	7952 × 5304	7952 × 5304

4.2. Experimental Result of Texture Quality

Two types of 3D models with complex structures and rich textures in different situations (building area and factory area) were selected for this experiment. The building area contained main elements such as building facades, windows and attachments, and the factory area contained the main elements such as oil tanks, road traffic signs, and pipelines. A comparative experiment was designed to compare the algorithm in this paper with the traditional algorithm [15] in the same environment.

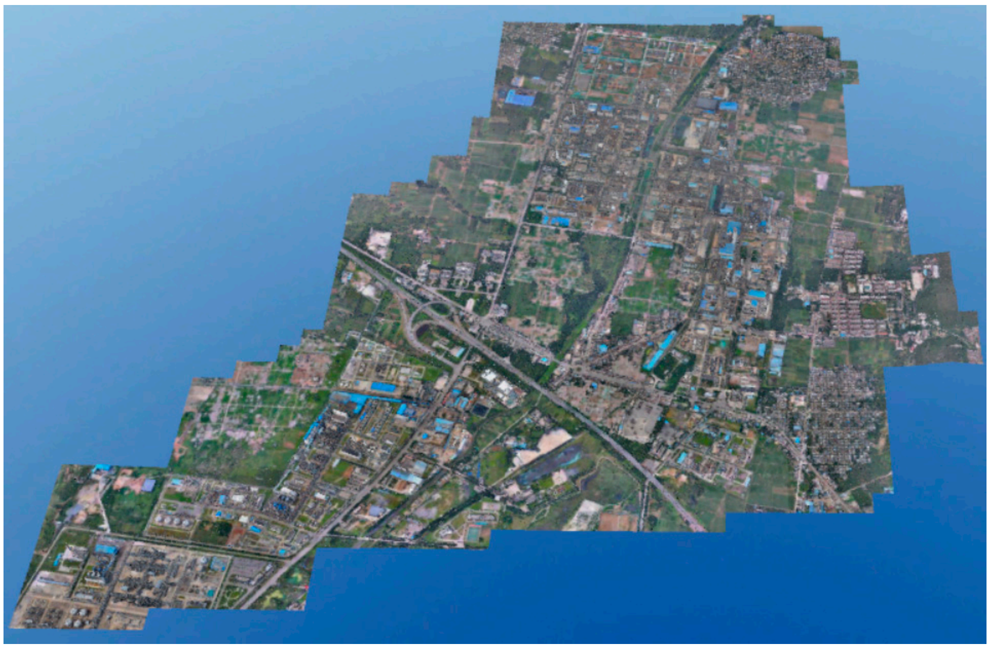

Figure 10. Experimental data.

There are two parameters in this algorithm: the mesh simplification parameter α and the texture simplification parameter β. To verify the effect of the two parameters on the quality of texture simplification, experiments were performed from two aspects: (1) α is changed and β is unchanged; (2) α is changed and β is changed.

1. α is changed, β is unchanged

The two types of 3D models were only mesh simplified without texture simplification, i.e., $\beta = 1.0$. Taking the original mesh ($\alpha = 1.00,\ \beta = 1.00$) as the true value and different mesh simplification parameters $\alpha = 0.50,\ 0.25,\ 0.125$, the texture distortion and deformation degree of the proposed algorithm and the traditional algorithm were compared. The experimental comparison results of the building area and the factory area are shown in Figures 11 and 12, respectively.

Experimental comparison with different grid simplification parameters: As seen from Figure 11a,c,e,g, smaller mesh simplification parameters of the traditional algorithm led to greater distortion and deformation of the texture. When the mesh simplification parameters were large, the degree of texture distortion and deformation was small. However, with the decrease in the mesh simplification parameters, the texture distortion and deformation were obvious, particularly at the oil tanks, road traffic signs, and pipelines in the factory area, as shown in Figure 12c,e,g. However, as seen from Figure 11a,b,d,f, under different mesh simplification parameters, the texture results of the proposed algorithm in the building area and factory area were not distorted and deformed, which was close to the original mesh.

Experimental comparison of the same mesh simplification parameters: Comparing the experimental results of the six groups of meshes with the same simplification parameters, such as Figure 11b,c, Figure 11d,e, Figure 11f,g, Figure 12b,c, Figure 12d,e, and Figure 12f,g, the number of triangular facets of the proposed algorithm was equal to that of the traditional algorithm, and the texture distortion and deformation of the proposed algorithm were less than that of the traditional algorithm, verifying the superiority of the proposed algorithm in the texture quality. In addition, the proposed algorithm had no distortion or deformation in the building facades, attachments, road traffic signs, pipelines, oil tanks, and other areas of the two models. Moreover, this algorithm can be applied to models containing different elements and has strong universality.

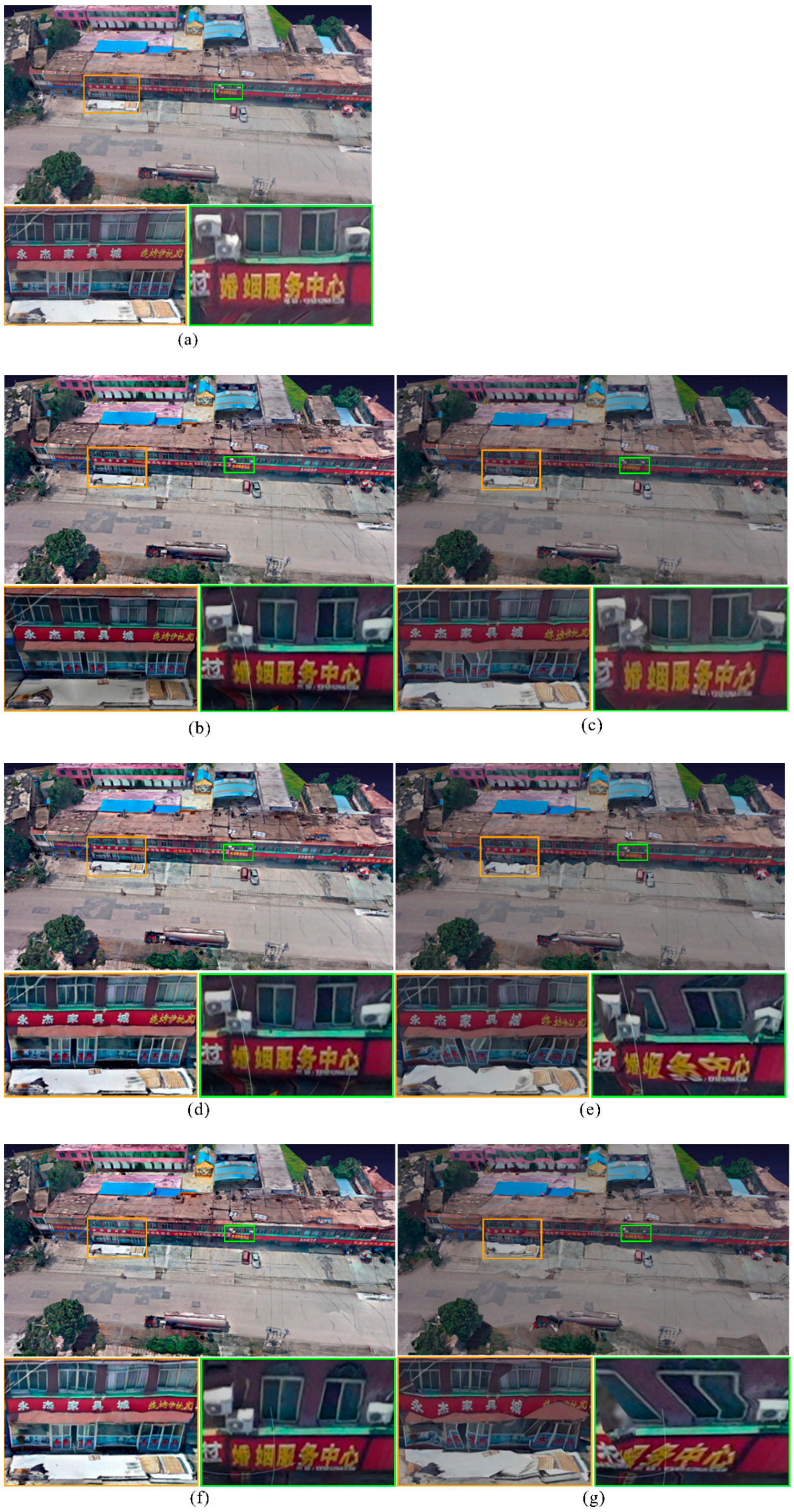

Figure 11. Texture quality comparison of the building area models. The orange and green framework in the upper overall image represent the corresponding local enlarged drawing below the overall image. (**a**) Original mesh (true value): $\alpha = 1.0$, $\beta = 1.0$, number of facets = 109,304. (**b**) The proposed algorithm: $\alpha = 0.50$, $\beta = 1.0$, number of facets = 54,651. (**c**) The traditional algorithm: $\alpha = 0.50$, $\beta = 1.0$, number of facets = 54,651. (**d**) The proposed algorithm: $\alpha = 0.25$, $\beta = 1.0$, number of facets = 27,326. (**e**) The traditional algorithm: $\alpha = 0.25$, $\beta = 1.0$, number of facets = 27,326. (**f**) The proposed algorithm: $\alpha = 0.125$, $\beta = 1.0$, number of facets = 13,662. (**g**) The traditional algorithm: $\alpha = 0.125$, $\beta = 1.0$, number of facets = 13,662.

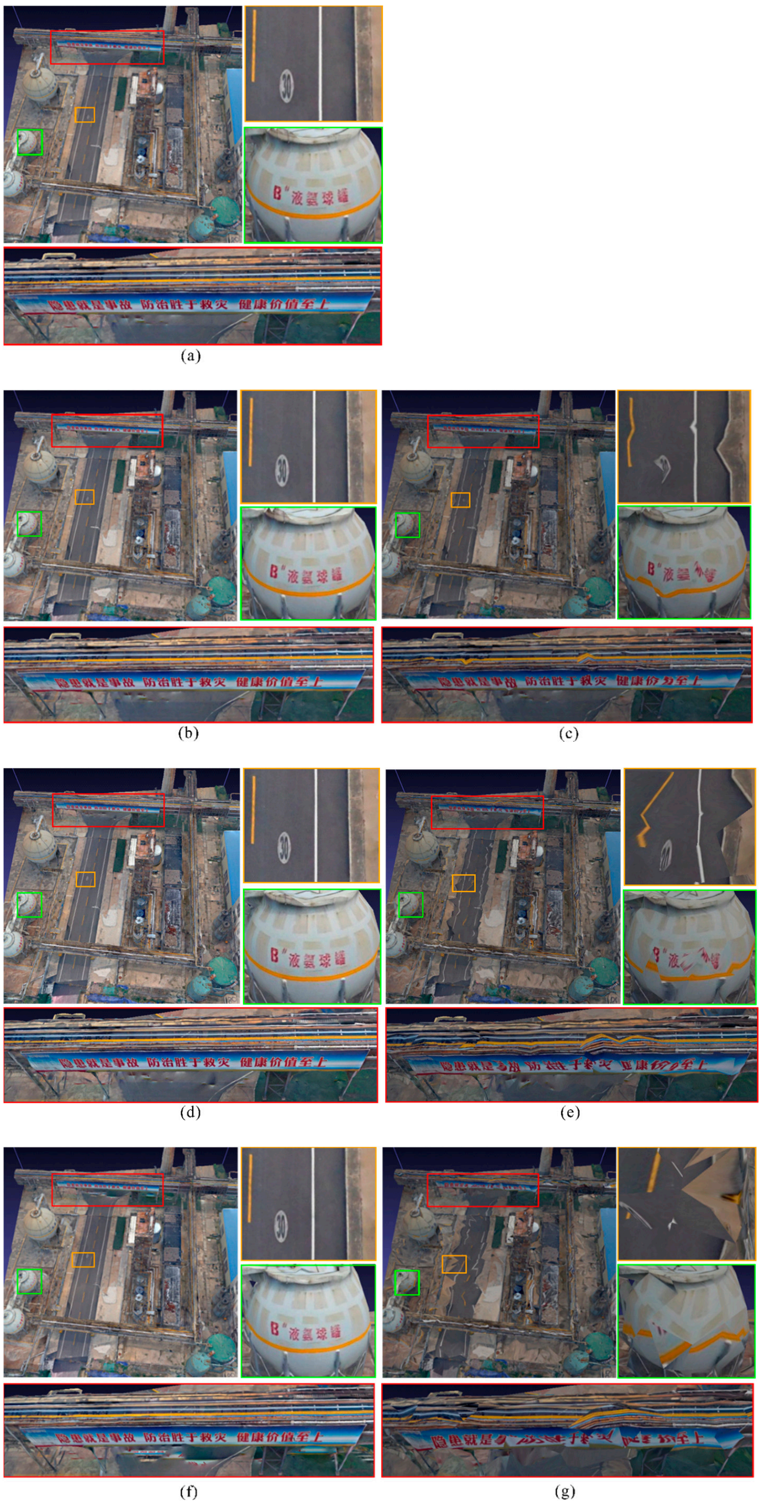

Figure 12. Texture quality comparison of the factory area models. The red, orange and green framework in the overall image represent the corresponding local enlarged drawing around the overall image. (**a**) Original mesh (true value): $\alpha = 1.0$, $\beta = 1.0$, number of facets = 125,288. (**b**) The proposed algorithm: $\alpha = 0.50$, $\beta = 1.0$, number of facets = 62,643. (**c**) The traditional algorithm: $\alpha = 0.50$, $\beta = 1.0$, number of facets = 62,643. (**d**) The proposed algorithm: $\alpha = 0.25$, $\beta = 1.0$, number of facets = 31,321. (**e**) The traditional algorithm: $\alpha = 0.25$, $\beta = 1.0$, number of facets = 31,321. (**f**) The proposed algorithm: $\alpha = 0.125$, $\beta = 1.0$, number of facets = 15,661. (**g**) The traditional algorithm: $\alpha = 0.125$, $\beta = 1.0$, number of facets = 15,661.

2. α is changed, β is changed,

The mesh simplification parameters in the experiment were set to $\alpha = \{1.0, 0.5, 0.25, 0.125\}$; in addition, according to the calculation formula $\beta = ceil(1/\alpha)$ in Section 3.3, $\beta = \{1.0, 2.0, 4.0, 8.0\}$ was obtained. The proposed algorithm was performed in the building area according to the above parameters, and the simplification results were obtained as shown in Figure 13.

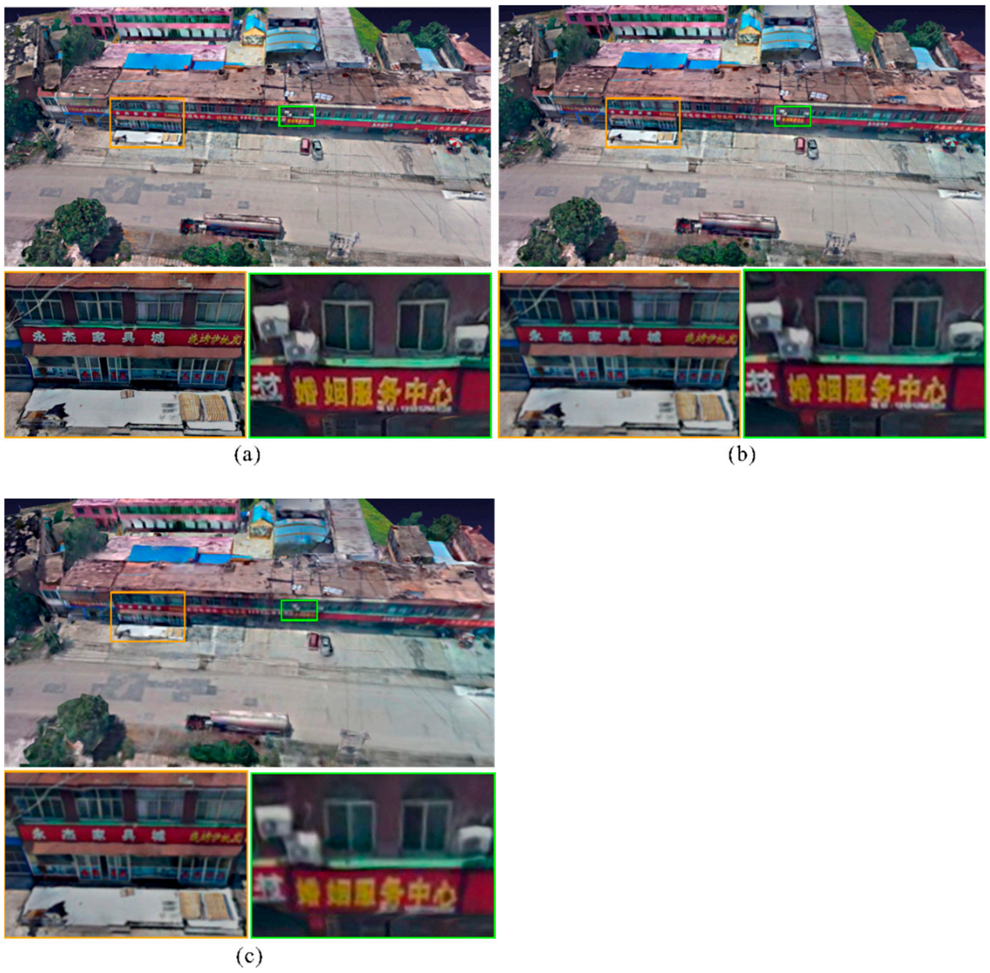

Figure 13. Texture quality comparison of the building area models. The orange and green framework in the upper overall image represent the corresponding local enlarged drawing below the overall image. (**a**) The proposed algorithm: $\alpha = 0.50$, $\beta = 2.0$, number of facets = 54,651. (**b**) The proposed algorithm: $\alpha = 0.25$, $\beta = 4.0$, number of facets = 27,326. (**c**) The proposed algorithm: $\alpha = 0.125$, $\beta = 8.0$, number of facets = 13,662.

As seen from Figure 13a–c, a larger texture simplification parameter led to a fuzzier texture. However, there was almost no difference in the texture distortion and deformation. By comparing Figures 11b and 13a, Figures 11d and 13b, and Figures 11f and 13c, when the mesh simplification parameters were the same and the texturing simplification parameters were different, the simplified texture in Figure 12 and the unsimplified texture in Figure 11 had almost no difference in the distortion and deformation. However, the simplified texture was less sharp than the unsimplified textures. This indicates that the size of the texture simplification parameter β did not affect the distortion and deformation of the texture but affected the clarity of the texture. In extreme cases, the texture content became difficult to distinguish beyond a certain texturing reduction parameter threshold.

4.3. Experimental Result of Texture Data Size

The proposed algorithm can not only simplify the mesh but also reduce the data size of the whole model through texture simplification. Multiple model tiles (with a size of 100.0 m $\times$ 100.0 m) were randomly selected from the four representative areas (residential areas, factory areas, vegetation areas, and road areas) as experimental data, as shown in Figure 14. The mesh and texture simplification parameters (α, β) were set as (1.0,1.0), (0.5, 2.0), (0.25, 4.0), and (0.125, 8.0), respectively. The data size of the proposed algorithm was compared with that of the traditional algorithm, as shown in Table 2, and the bar graph is shown in Figure 15.

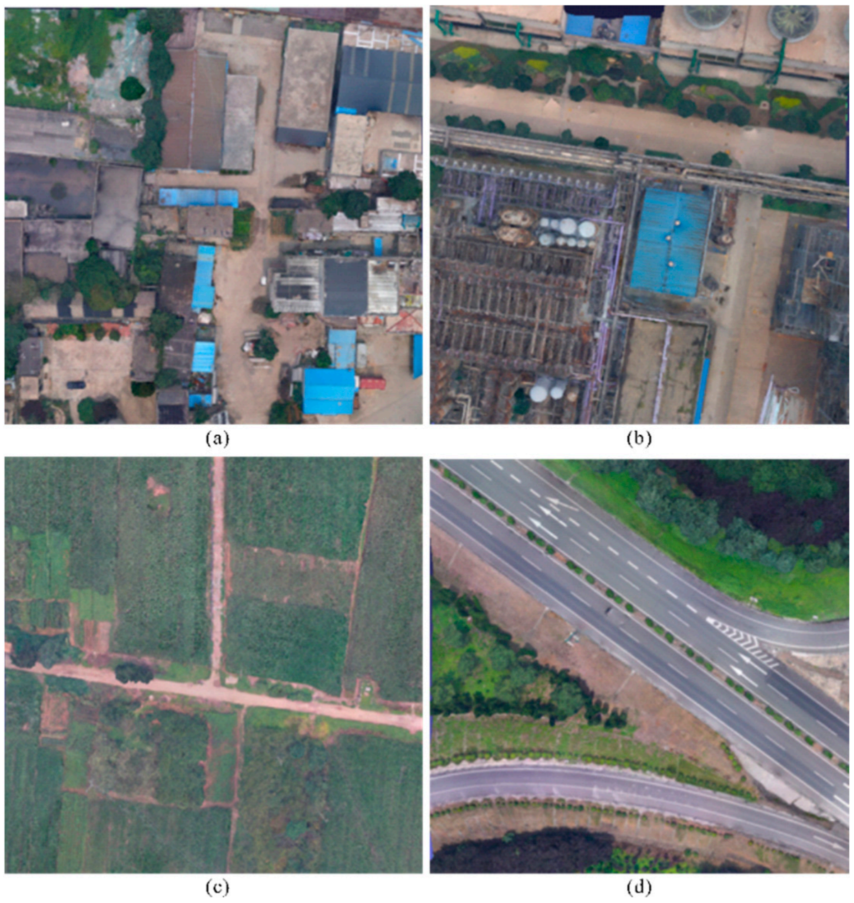

Figure 14. Four types of 3D models of different areas. (**a**) Residential areas. (**b**) Factory areas. (**c**) Vegetation areas. (**d**) Road areas.

As seen from Table 2, the data size of the 3D model can be reduced in the proposed algorithm through simplification in terms of mesh and texture, and the data size of the texture exponentially decreases with an increasing β value. For example, when $(\alpha, \beta) = (0.5,\ 2.0)$ and $(\alpha, \beta) = (0.125,\ 8.0)$, the model data size of the proposed algorithm is only 38.9% and 7.11% of that of the original 3D model. However, the traditional method only reduces the data size of the mesh, while the data size of the texture does not change. With the decrease in the α value and the increase in the β value, the gap between the proposed algorithm and the traditional algorithm in 3D model data size reduction is more significant. For example, when $(\alpha, \beta) = (0.125,\ 8.0)$, the model data size of the proposed algorithm is only 12.9% of that of the traditional algorithm. In summary, the proposed algorithm in this paper can effectively alleviate the contradiction between the rapidly increasing amount of 3D model data and the limited transmission bandwidth and smooth model rendering.

Table 2. Data size statistics. The data size (mesh, texture) after the proposed algorithm and the traditional algorithm are simplified under different simplification parameters are counted, and the unit is kb. (**a**) Residential areas. (**b**) Factory areas. (**c**) Vegetation areas (farmland for example). (**d**) Road areas (highways for example).

(**a**)				
Simplification parameters ($\alpha; \beta$)	(1.0; 1.0)	(0.5; 2.0)	(0.25; 4.0)	(0.125; 8.0)
The proposed algorithm (mesh; texture)	(13,329; 11,417)	(6578; 4021)	(3219; 1158)	(1589; 347)
Traditional algorithm (mesh; texture)	(13,329; 11,417)	(6578; 11,417)	(3219; 11,417)	(1589; 11,417)
(**b**)				
Simplification parameters ($\alpha; \beta$)	(1.0; 1.0)	(0.5; 2.0)	(0.25; 4.0)	(0.125; 8.0)
The proposed algorithm (mesh; texture)	(15,281; 12,561)	(7540; 4004)	(3674; 1204)	(1800; 373)
Traditional algorithm (mesh; texture)	(15,281; 12,561)	(7540; 12,561)	(3674; 12,561)	(1800; 12,561)
(**c**)				
Simplification parameters ($\alpha; \beta$)	(1.0; 1.0)	(0.5; 2.0)	(0.25; 4.0)	(0.125; 8.0)
The proposed algorithm (mesh; texture)	(12,679; 11,284)	(6251; 3234)	(3168; 792)	(1610; 203)
Traditional algorithm (mesh; texture)	(12,679; 11,284)	(6251; 11,284)	(3168; 11,284)	(1610; 203)
(**d**)				
Simplification parameters ($\alpha; \beta$)	(1.0; 1.0)	(0.5; 2.0)	(0.25; 4.0)	(0.125; 8.0)
The proposed algorithm (mesh; texture)	(12,549; 10,007)	(6186; 3339)	(3138; 785)	(1597; 234)
Traditional algorithm (mesh; texture)	(12,549; 10,007)	(6186; 10,007)	(3138; 10,007)	(1597; 10,007)

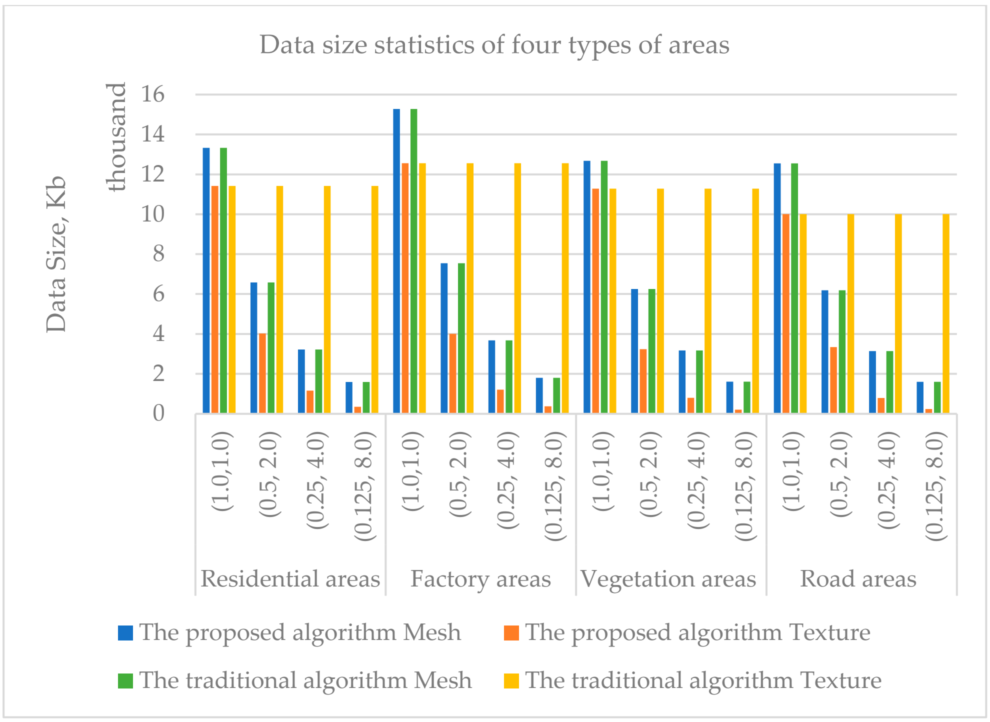

Figure 15. Histogram of data size statistics.

5. Discussion

In terms of mesh simplification, the algorithm proposed in this paper is the same as the traditional algorithm, which is based on the QEM algorithm and performs folding and merging operations on the triangular facets of the mesh. In terms of the texture simplification, the traditional algorithm only recalculates the texture coordinates, which leads to a certain degree of texture distortion and deformation. The proposed algorithm can effectively avoid texture distortion and deformation and can simplify the texture content. This paper discusses texture quality and texture data size.

5.1. Comparative Analysis of Texture Quality

1. Influence of the mesh simplification parameters α on the texture quality

The comparison results of Figures 11 and 12 show that, using different mesh simplification parameters α, the model simplification results of the proposed algorithm in this paper do not have texture distortion or deformation problems. For the traditional algorithm, as the mesh simplification parameter α decreases, the degree of distortion and deformation of the texture becomes increasingly obvious.

This is because, with the reduction in mesh simplification parameters, the texture patches corresponding to the folded and merged facets are less likely to be continuous in the 2D texture space. This discrete distribution phenomenon leads to the inability to correctly calculate the texture coordinates of the newly generated triangular facets. Thus, the degree of texture distortion and deformation becomes more serious. The performance of the traditional algorithm is poor when dealing with such simplified textured 3D models. The algorithm in this paper solves this shortcoming of the traditional algorithm. Therefore, the simplified 3D model has almost no distortion or deformation in texture.

However, looking closely at the results of our algorithm presented in Figures 11 and 12, slight distortions and deformations of the texture can be found. This is mainly because the mesh simplification parameter α is smaller, and errors occur when the QEM algorithm simplifies the mesh, resulting in unevenness in the plane area after simplification, for example.

2. Influence of the texture simplification parameter β on the texture quality

Combining Figures 13 and 11b,d,f, the algorithm proposed in this paper can use the texture simplification parameter β to control the degree of texture simplification, and the texture simplification results are effective, without distortion and deformation. In addition, the following phenomenon can be found: with the increase in the texture simplification parameter, the simplified texture gradually becomes blurred. This is mainly because the texture simplification of the algorithm proposed in this paper performs appropriate downsampling processing to the images in the reference dataset, resulting in a reduction in texture clarity. However, the proposed algorithm can meet the normal requirements of model simplification under general cases.

In the next step, we can synthesize the reference image, retaining the main or important content in the image, which can be used as a source of texture data when the texture simplification parameter β is large.

5.2. Comparative Analysis of Texture Data Size

Objectively, in terms of the amount of 3D model data, textures account for a major part compared to meshes. Thus, research on texture simplification is carried out in this paper, which has important value and practical significance.

The comparative analysis of the statistical values in Table 2 confirms the effectiveness of the proposed algorithm in this paper in reducing the amount of data. This is mainly because the algorithm proposed in this paper uses the texture simplification parameters to downsample all the images in the reference image set and then uses the downsampled images as the data source for texture mapping of the mesh. Therefore, there is a considerable difference between the simplified texture data volume of the algorithm in this paper and the results of the traditional algorithm. It can be concluded that the 3D model simplified by the algorithm in this paper can effectively alleviate the contradiction between the rapidly growing 3D model data volume and the limited transmission bandwidth and smooth model rendering.

6. Conclusions

When the existing 3D mesh simplification algorithm performs mesh simplification, since only the texture coordinates are recalculated for the texture, there are different degrees of texture distortion and deformation, and the existing 3D mesh simplification algorithm does not simplify the texture. Therefore, a model (mesh and texture) simplification algorithm for 3D reconstruction was proposed in this paper. First, a reference 3D model scene was constructed on the basis of the original mesh and the recovered camera pose. Second, the reference image set was generated by image acquisition on the basis of the reference 3D model scene and the recovered camera pose. Lastly, the QEM algorithm is used to simplify the mesh, and the improved texture reconstruction algorithm is used to simplify the texture and avoid distortion. Through experimental verification and comparative analysis, the conclusions are as follows: (1) in the case of the mesh simplification only, the proposed algorithm basically has no texture distortion and deformation and is robust; (2) the proposed algorithm can support mesh and texture simplification together. With the increasing texture simplification parameters, the texture has almost no distortion and deformation, but only the clarity is reduced, which significantly reduces the amount of texture data. When the texture simplification parameter $\beta = 2.0$, the texture data size of the proposed algorithm is only 28.17% of that of the traditional algorithm. When the texture simplification parameter $\beta = 8.0$, the texture data size of the proposed algorithm is only 2.40% of that of the traditional algorithm.

The algorithm proposed in this paper still has some shortcomings, which will be further studied. The proposed algorithm uses all the images in the reference image set in the mesh and texture simplification steps. In the next step, the number of reference images used in the simplification can be optimized to improve the efficiency of the algorithm.

Author Contributions: Conceptualization, Z.L. and C.Z.; methodology, Z.L.; software, Z.L. and H.C.; experiment, Z.L., W.Q. and S.Z.; writing—original draft preparation, Z.L. and H.C.; supervision, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Basal Research Fund of CASM (Grant Nos. AR2120, AR2215).

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

References

1. Xu, B.; Zhang, L.; Liu, Y.; Ai, H.; Wang, B.; Sun, Y.; Fan, Z. Robust Hierarchical Structure from Motion for Large-Scale Unstructured Image Sets. *ISPRS J. Photogramm. Remote Sens.* **2021**, *181*, 367–384. [CrossRef]
2. Zolanvari, S.M.I.; Ruano, S.; Rana, A.; Cummins, A.; Da Silva, R.E.; Rahbar, M.; Smolic, A. DublinCity: Annotated LiDAR Point Cloud and Its Applications. *arXiv* **2019**, arXiv:1909.03613.
3. Vrubel, A.; Bellon, O.R.; Silva, L. A 3D Reconstruction Pipeline for Digital Preservation. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: New York, NY, USA, 2009; pp. 2687–2694.
4. Galliani, S.; Lasinger, K.; Schindler, K. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 873–881.
5. Mostegel, C.; Prettenthaler, R.; Fraundorfer, F.; Bischof, H. Scalable Surface Reconstruction from Point Clouds with Extreme Scale and Density Diversity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 904–913.
6. Zhang, C.; Zhang, H.; Guo, B.; Peng, Z. Structure-aware simplified algorithm of mesh model for urban scene. *Acta Geod. Et Cartogr. Sin.* **2020**, *49*, 334–342. [CrossRef]
7. Garland, M.; Heckbert, P.S. Surface Simplification Using Quadric Error Metrics. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 3–8 August 1997; pp. 209–216.
8. Xian, Y.; Fan, Y.; Huang, Y.; Wang, G.; Tu, C.; Meng, X.; Peng, J. Mesh Simplification with Appearance-Driven Optimizations. *IEEE Access* **2020**, *8*, 165769–165778. [CrossRef]
9. Rossignac, J.; Borrel, P. Multi-Resolution 3D Approximations for Rendering Complex Scenes. In *Modeling in Computer Graphics*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 455–465.
10. Schroeder, W.J.; Zarge, J.A.; Lorensen, W.E. Decimation of Triangle Meshes. In Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, Chicago, IL, USA, 26–31 July 1992; pp. 65–70.
11. Garland, M.; Zhou, Y. Quadric-Based Simplification in Any Dimension. *ACM Trans. Graph.* **2005**, *24*, 209–239. [CrossRef]
12. Pan, Z.; Pang, M. Survey for decimation of geometric meshes. *J. Jiangsu Univ. Nat. Sci. Ed.* **2005**, *1*, 67–71.

13. Garland, M.; Heckbert, P.S. Simplifying Surfaces with Color and Texture Using Quadric Error Metrics. In Proceedings of the Proceedings Visualization'98 (Cat. No. 98CB36276), Research Triangle Park, NC, USA, 18–23 October 1998; IEEE: New York, NY, USA, 1998; pp. 263–269.
14. Sporysz, P.; G\lomb, P. Improving the Quality of Mesh Simplification with Texture Saliency Measurement. In Proceedings of the 2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA), Paris, France, 14–17 October 2014; IEEE: New York, NY, USA, 2014; pp. 1–6.
15. She, J.; Gu, X.; Tan, J.; Tong, M.; Wang, C. An Appearance-Preserving Simplification Method for Complex 3D Building Models. *Trans. GIS* **2019**, *23*, 275–293. [CrossRef]
16. Cohen, J.; Olano, M.; Manocha, D. Appearance-Preserving Simplification. In Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, Orlando, FL, USA, 19–24 July 1998; pp. 115–122.
17. Williams, N.; Luebke, D.; Cohen, J.D.; Kelley, M.; Schubert, B. Perceptually Guided Simplification of Lit, Textured Meshes. In Proceedings of the 2003 Symposium on Interactive 3D Graphics, Monterey CA, USA, 27–30 April 2003; pp. 113–121.
18. Qu, L.; Meyer, G.W. Perceptually Guided Polygon Reduction. *IEEE Trans. Vis. Comput. Graph.* **2008**, *14*, 1015–1029. [CrossRef] [PubMed]
19. González, C.; Castelló, P.; Chover, M.; Sbert, M.; Feixas, M.; Gumbau, J. Simplification Method for Textured Polygonal Meshes Based on Structural Appearance. *Signal Image Video Processing* **2013**, *7*, 479–492. [CrossRef]
20. Castelló, P.; Sbert, M.; Chover, M.; Feixas, M. Viewpoint-Driven Simplification Using Mutual Information. *Comput. Graph.* **2008**, *32*, 451–463. [CrossRef]
21. Garcia, I.; Patow, G. Igt: Inverse Geometric Textures. In Proceedings of the ACM SIGGRAPH Asia 2008 Papers, Singapore, 9–13 December 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 1–9.
22. Chen, C.-C.; Chuang, J.-H. Texture Adaptation for Progressive Meshes. *Computer Graphics Forum.* **2006**, *25*, 343–350. [CrossRef]
23. Coll, N.; Paradinas, T. Accurate Simplification of Multi-Chart Textured Models. *Computer Graphics Forum.* **2010**, *29*, 1842–1853. [CrossRef]
24. Waechter, M.; Moehrle, N.; Goesele, M. Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 836–850.
25. Bi, S.; Kalantari, N.K.; Ramamoorthi, R. Patch-Based Optimization for Image-Based Texture Mapping. *ACM Trans. Graph.* **2017**, *36*, 1015–1029. [CrossRef]
26. Boykov, Y.; Veksler, O.; Zabih, R. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 1222–1239. [CrossRef]
27. Bansal, N.; Khan, A. Improved Approximation Algorithm for Two-Dimensional Bin Packing. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Portland, OR, USA, 5–7 January 2014; SIAM: Philadelphia, PA, USA, 2014; pp. 13–25.