

Article

Improved Edge Folding Algorithm for 3D Building Models Taking into Account the Visual Features

Haoyuan Bai, Tao Shen *, Liang Huo *, Xiaoyu Wang and Xinyu Liu

School of Geomatics and Urban Spatial Information, Beijing University of Civil Engineering and Architecture, Beijing 100044, China; hybai9627@163.com (H.B.); wangxiaoyu.bucea@gmail.com (X.W.); xinyu0904@163.com (X.L.)

* Correspondence: shentao@bucea.edu.cn (T.S.); huoliang@bucea.edu.cn (L.H.)

Abstract: Simplifying 3D building models, effectively reducing model complexity and improving mapping efficiency, is an important part of 3D GIS. In order to address the problem that the simplification basis considered by most 3D building model data compression algorithms is mainly based on geometric features and fails to retain the visual features of the model, this paper proposed a half-folding simplification algorithm for 3D building models that took the visual features into account. The algorithm incorporated regional boundary constraints in the pre-processing stage to focus on preserving the boundary features of the model, and incorporated visual constraints, such as the vertex importance, edge length, and texture, in the calculation of the edge folding cost to construct a new error metric model. The simplification order was changed by redefining the edge folding cost, and a trade-off between compression rate and retention rate was made by using half-edge folding to maximise the retention of the visual features of the building. The experimental results showed that the algorithm could effectively reduce the number of elements drawn by the system and achieve a high level of detail information retention in the visual feature region of the model with a good visual representation.

Keywords: visual features; 3D building model; data compression; error metric model; QEM algorithm improvement



Citation: Bai, H.; Shen, T.; Huo, L.; Wang, X.; Liu, X. Improved Edge Folding Algorithm for 3D Building Models Taking into Account the Visual Features. *Buildings* **2023**, *13*, 2739. <https://doi.org/10.3390/buildings13112739>

Academic Editor: Haifeng Liao

Received: 25 September 2023

Revised: 26 October 2023

Accepted: 27 October 2023

Published: 30 October 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

In recent years, 3D urban models have been widely used as critical spatial data infrastructures and the main parts of 3D urban models are 3D building models [1]. Complex building models often contain a large amount of detail and geometric information, making their storage, transmission, and rendering difficult and expensive [2]. The real-time drawing of large-scale 3D scenes requires high-quality rendering and must also meet the operational requirements of real-time user interaction. In order to obtain detailed models that are suitable for use in a wide range of scenes, different levels of simplification of the high-resolution mesh model are required [3].

The LOD (level of detail) technique means that, according to the position and importance of the nodes of the object model in the display environment, it decides the resource allocation for object rendering. It reduces the number of faces and the level of detail of the non-important objects to obtain efficient rendering operations, and it is an effective method for reducing the complexity of 3D data. In order to conform to the hierarchical nature of the cognitive process, LOD models need to be quantitatively optimised for models of varying complexity, adaptively retaining the important features and reducing unnecessary redundancy, which is an important technical component of the Web3D presentation [4]. Mesh simplification methods, i.e., streamlining minor details while retaining important geometric and visual features of the model, are used to reduce the amount of model data.

First proposed by Clark [5] in the 1970s, 3D mesh model simplification has been extensively studied since the 1990s. Currently, a variety of mesh simplification methods

have been proposed based on different algorithmic ideas. The common simplification mechanisms include vertex clustering, incremental simplification, sampling, and adaptive subdivision. The simplification algorithms basically use some combination or variation of these four mechanisms. The edge folding algorithm belongs to one of the incremental simplifications, which is essentially vertex deletion. The algorithm has excellent characteristics such as a good quality of simplification, a good robustness, the ability to make substantial simplifications to generate models of different accuracies, and an ease of seamless switching between different levels of detail [6].

2. Related Work

As early as 1993, Hoppe [7] used the edge folding method to simplify the mesh model. The algorithm determined the order of edge folding and the location of new vertices by optimising a global energy equation and generated simplified models of different complexities by adding details one by one. The results generated by this algorithm built a good quality model, but its computational process was non-linear and difficult to implement and use. In 1997, Garland and Heckbert [8] proposed a quadratic error measure simplification algorithm based on this, which used the sum of the squared distances from the new vertex to the first-order neighbourhood triangle as the error in the error measure. The calculation process was simple and effectively limited the local curvature and volume changes while maintaining the appearance features. However, it did not maintain the detail features of the model surface well. Lindstrom [9] weighted the distance from the model's vertex to the planes adjacent to it using the area of those adjacent planes to optimise the square of the volume and to obtain the location of the new vertex.

Normal meshes were introduced by Guskov et al. [10]. The normal grid was a multi-resolution grid where each sub-grid was noted as a normal offset to the coarse grid. The normal grid then used the Garland and Heckbert simplification algorithm to generate the grid hierarchy. Kim et al. [11] proposed a simplification algorithm based on discrete curvature parametrization. The method used a butterfly subdivision template to calculate the positions of the new vertices after edge folding and used the value of the change in the discrete curvature due to edge folding as the error. Silva and Gomes [12] proposed a simplification algorithm based on the coplanarity of the faces around collapsed edges. An edge collapsed into a vertex if the angle between any two faces was less than a fixed value. This algorithm simplified quickly, but sometimes produced meshes with long triangles, i.e., relatively low-quality meshes. Qu et al. [13] proposed a method for perception-driven simplification based on using a 2D image perception analysis and then established a link between the geometric vertices and the 2D screen projections. The method made use of the more established perceptual evaluation model for 2D images by finding the perceptual degree of a vertex in image space using the grid vertices as the units of the perceptual information transfer.

Wang X et al. [14] proposed a tetrahedron simplification method based on triangle extraction. Zhou Yuanfeng et al. [15] proposed a feature-preserving mesh simplification algorithm based on the volumetric measures, which combined the square volume error of the triangles, both the shape factor and normal constraints. The simplification algorithm proposed by Menzel N et al. [16] was implemented using triangle extraction, which took into account human eye vision, introduced a perception-based metric, and proposed an efficient simplification algorithm for human eye vision systems. Li Fengxia [17] analysed the advantages and disadvantages of the QEM algorithm and proposed a resolution-based model simplification algorithm to preserve the model structure as much as possible under the same resolution conditions.

Yi Bing et al. [18] divided the model grid vertices according to Medioni's model tensor voting theory and weighted the divided vertices using the edge folding cost, according to the set weights. Zhang Shixue et al. [19] used the distance of the inter-model deformation to generate multiple resolution models, which were generated with the appropriate resolution as needed. Yan Dong et al. [20] calculated the vertex importance of each vertex and selected

the order of edge folding using the vertex importance as a way to preserve the detailed features of the model. Ozaki et al. [21] proposed a simplification algorithm for complex mesh models based on the quadratic error algorithm, which clustered the point cloud data using machine learning to obtain the details of the resulting model sampling. Its simplification efficiency was high and suitable for 3D complex grid models with a large data volume, but it was easy to lose the detail features of the sharp regions of the model.

Dassi et al. [22] introduced the displacement change of the original model and the resultant model in the folding cost to improve the overall simplification effect of the grid model. However, the algorithm had a limited effect on the retention of the local detail features of the model. Duan Liming et al. [23] proposed a mesh subdivision method that added the volume error and triangle spreading to the error measure, but its effect on the simplification of the boundary triangles was limited. Wuli W et al. [24] addressed the problem that most of the existing mesh simplification algorithms could not simultaneously take into account the details, approximate simplicity ratio, and mesh quality, and proposed a dynamic error control and particle swarm optimization for the adaptive simplification optimization of triangular meshes. These were divided into two stages: mesh simplification and mesh optimization. She et al. [25] introduced a new error metric that took into account both geometric and texture errors, thus preserving the geometric features and reducing the texture distortion. Using these methods, it was possible to set different simplification rates to generate multiple models with different levels of detail, and then implement discrete LOD strategies. However, it was difficult to generate correlations between the models with different levels of detail to implement a progressive transfer strategy. Chen Bo et al. [26] proposed a new 3D building simplification method that considered both the model grid and the building structure. The method divided the 3D building into a primary structure and a secondary structure. It then used StructureTree (a multi-way tree) to organise these structures. The structures were organised according to the dependencies between the building structures. When simplifying the building, whether or not to simplify the mesh or remove the secondary structures in the StructureTree leaf nodes depended on the volume change caused by the edge collapse and the visual saliency of the removed structures. This approach achieved a better quality of simplification.

In summary, many excellent algorithms have emerged from domestic and international research. However, the majority of them failed to retain the visual characteristics of the model well and challenges still remain for achieving a balance between a high efficiency and a high accuracy. Based on the above research, this paper proposed a quadratic error metric data compression algorithm that took into account the visual features. The algorithm first performed a pre-processing operation on the model to classify the vertices and edges and initially determine the importance of the edges. Then, it incorporated visual constraints, such as the vertex importance, edge length, and texture, into the calculation of the folding cost of the edges to construct a new error metric model, which in turn changed the simplified order. A new error metric model was constructed to change the simplified order of edge folding and the position of the new vertices, and to maintain the geometric constraints of the model as much as possible by using half-edge folding to improve the rendering efficiency while taking into account the user's visual characteristics.

3. Materials and Methods

3.1. Technical Route

The technical route is shown in Figure 1. For an original 3D building model, all its vertices and edges were first classified. The importance of each edge was initially determined, then the error cost of edge folding was derived by combining each constraint factor. The type of edge was considered while using the half-edge folding operation to determine the position of the new vertex, and the affected edge folding cost was updated in a timely manner during the folding process. By continuously performing the simplification operation, the quality of the simplified results was evaluated from the visual and quantitative

perspectives, respectively, and the improved algorithm was summarised and presented in a prospective manner.

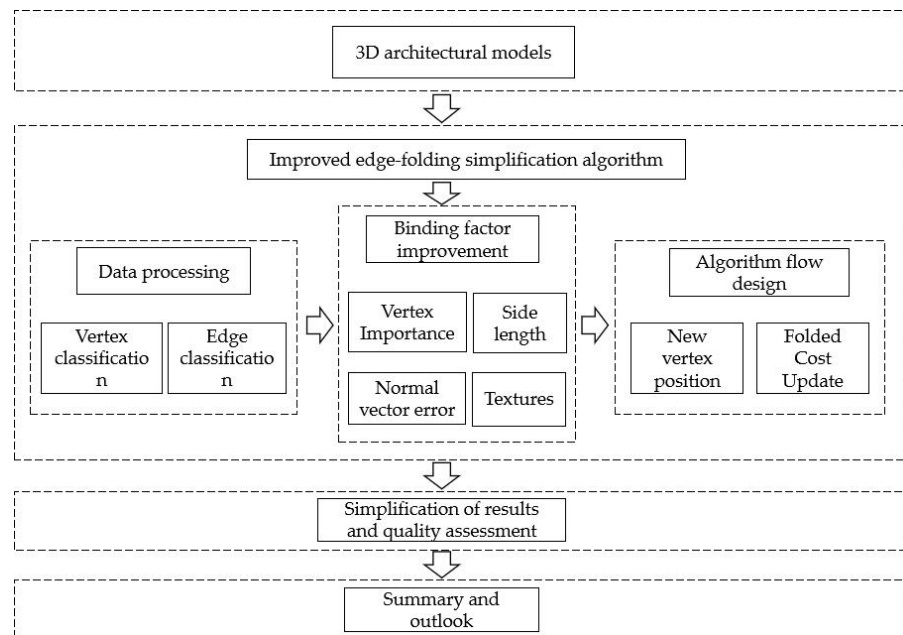


Figure 1. A technology roadmap for the improved edge folding algorithm for the 3D building models, taking into account the visual features.

3.2. QEM Simplification Algorithm

The introduction of the quadratic error metrics algorithm dates back to 1998, when it was first proposed by Garland and Heckbert in their paper ‘Surface Simplification Using Quadric Error Metrics’. This QEM simplification algorithm, which calculates the edge weights based on the point-to-surface distance, is simple to compute and can not only effectively reduce the number of faces in the model but can also keep the shape and topology of the model unchanged, generating a fast mesh model with a good simplification effect, which can be used in applications requiring high real-time performance [27].

The QEM algorithm uses the distance from a vertex to its first-order neighbourhood triangle face piece as the error measure for the edge folding simplification. The algorithm assigns an error measure matrix to all the vertices in the model, solves the error matrix and shrinkage cost of the new vertices resulting from edge folding from this error measure matrix, and determines the simplification of the edges according to the size of the shrinkage cost. the simplification is achieved by iterating over the edges with the lowest shrinkage cost in the folding model.

In a three-dimensional space, a plane can be represented as $ax + by + cz + d = 0$, where $a^2 + b^2 + c^2 = 1$, d is an arbitrary constant. Let $p = [a \ b \ c \ d]^T$ represent the plane and the coordinates of the new vertex V be $[x \ y \ z \ 1]^T$, such that the square of the distance from the new vertex V to the plane P is as follows.

$$D_p^2(V) = \left(\frac{|ax + by + cz + d|}{\sqrt{a^2 + b^2 + c^2}} \right)^2 = (p^T V)^2 = (V^T p)^2 = V^T (pp^T) V = V^T K_p V \quad (1)$$

where K_p is a fourth-order symmetric matrix, known as the quadratic error measure matrix of the triangle, which has the following definition.

$$K_p = pp^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \quad (2)$$

Let $Q(v_i) = \sum_{p \in P(v_i)} K_p$ be the quadratic error measure matrix of the vertices and $P(V_i)$ be a set of triangles. Each triangle in this set contains vertices V_i . By Garland's definition, when folding an edge (V_i, V_j) to a new vertex V , the edge folding cost is as follows.

$$\Delta(V) = \sum_{p \in P(V_i) \cup P(V_j)} D_p^2(V) = V^T \sum_{p \in P(V_i) \cup P(V_j)} K_p V = V^T Q(V) V \quad (3)$$

So, $\Delta V = V^T Q V$. Expanding the quadratic term of the folding error, the expression is as follows.

$$\Delta(V) = q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + 2q_{34}z + q_{44} \quad (4)$$

The size of the contraction cost depends on the location of the new vertex V . The greater the distance between the new vertex V and the triangle face piece associated with the edge (V_i, V_j) , the greater the contraction cost. An optimal position of the new vertex V exists such that ΔV reaches a local minimum. Taking the partial derivatives of x , y and z in Equation (4) and making them zero is equivalent to solving Equation (5).

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (5)$$

If the coefficient matrix is invertible, the position of V can be found using Equation (5). Otherwise, the point that minimises the contraction cost ΔV is chosen as the position of the new vertex among the midpoint and the two endpoints of the edge (V_i, V_j) .

3.3. Constraint Factor Addition

The QEM was calculated by finding the QEM weights of the two endpoints and the midpoint of the endpoints if the new vertex could not be solved directly, then comparing the magnitudes, and finally taking the vertex with the smallest edge weight as the shrinkage retention vertex. This was more computationally intensive each time. Half-edge shrinkage can avoid this process and make the computation less intensive, so this paper used the half-edge shrinkage strategy. However, since not adding new vertices would make the simplified model less effective, some constraints needed to be added to the selection of the retained vertices for each side shrinkage operation in order to improve the simplification effect to a certain extent. In this paper, we added the vertex importance, edge length, normal vector error, and texture to form a new error matrix model, which could effectively enhance the realism and visualization of the model.

3.3.1. Vertex Importance

In general, the more convex the vertex, the more important the vertex is and the more important it should be retained in the model simplification process. In this paper, we focused on the approximate curvature of the vertex and the average area of the local area to determine the vertex importance.

1. Approximate vertex curvature

For most 3D mesh models, different regions have different resolutions and, therefore, different curvatures. For example, key regions of the model such as sharp edges, inflection points, and creases have a large curvature, while other regions have a relatively small curvature [28]. The curvature in this context refers to the approximate curvature, and since the surface of the model is a second-order integrable surface, the surface of the model can be considered approximately smooth. The vertex curvature reflects the degrees of detail that are evident in the feature regions of the model. When determining the order in which the edges should be folded, lowering the priority of edges with a greater vertex curvature

can better preserve the geometric features of the model. The vertex curvature is defined as follows. Take any triangular face piece T and denote its three vertices as V_i , V_j , V_k , then the unit normal vector n_i of the triangular face piece T can be expressed as follows.

$$n_i = \frac{(V_j - V_i) \times (V_k - V_i)}{\|(V_j - V_i) \times (V_k - V_i)\|} \quad (6)$$

By weighting the area S_i of the normal vectors of the first-order neighbourhood triangle face piece of the vertex V_i , the vertex normal vector n_{v_i} can be solved as follows.

$$n_{v_i} = \frac{\sum_{i=1}^k S_i n_i}{\|\sum_{i=1}^k S_i n_i\|} \quad (7)$$

Once the normal vector of a vertex V_i is obtained, the curvature C_{v_i} of that vertex can be solved as follows.

$$C_{v_i} = \frac{\sum_k \alpha(n_{v_i}, n_i)}{k} \quad (8)$$

where $\alpha(n_{v_i}, n_i)$ is the angle between the vertex normal vector and k is the associated triangle face piece.

2. Average area of the local regions

A mesh model often has a dense, small or large number of triangles in the detail region and collapsing this region can result in a loss of detail in the model. In addition, when the local area of a vertex is the same, but the surrounding triangles are not the same in density, it is difficult to distinguish between them using the normal local area metric. If the average local area of the vertex is large, then the shapes of the triangles around the vertex are large and the area should be relatively flat. If the average local area of the vertex is small, then the shapes of the triangles around the vertex are small and the area is relatively curved. Suppose that the local area of the vertex V_0 consists of $\triangle V_0 V_1 V_2$, $\triangle V_0 V_2 V_3$, $\triangle V_0 V_3 V_4 \dots \triangle V_0 V_n V_1$, then the average area of the local region of the vertex V_0 can be expressed as follows.

$$\overline{S_{V_0}} = \frac{\sum_{i=1}^n S_{t_i}}{n} \quad (9)$$

where S_{t_i} is the area of the i th triangle in the neighbouring triangle of the vertex V_0 .

3.3.2. Edge Lengths

For most mesh models, the feature part tends to be in a small area of the model and the rest of the area is relatively flat. While the previous method maintains the detailed features of the model, it also causes the simplified resulting mesh to spend too many triangular slices to describe the feature area, resulting in sparser triangular slices in other areas, which may lead to an overall increase in the error between the simplified mesh and the original mesh. To avoid this, the distribution of the triangles in the simplified model needs to be adjusted. The longer the length of the edge connected by two vertices in the mesh model, the greater the effect of simplifying that edge on the error of the model. Therefore, the edge length is added as a constraint to the cost of folding the edge as well.

Let the coordinates of the two endpoints V_i and V_j of the folded edge be (x_i, y_i, z_i) and (x_j, y_j, z_j) , respectively. The edge length l_{ij} can then be found by calculating the following equation.

$$l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad (10)$$

The formula for the distance between two points in a three-dimensional space is derived from the formula for the distance between two points in a plane coordinate system combined with the Pythagorean theorem and the diagonal formula of the rectangle. This has been repeatedly shown to be correct, so we used it directly.

3.3.3. Normal Vector Errors

The normal vector is a vector perpendicular to the surface of the model, which describes the orientation and shape of the surface. Keeping the normal vector of the model constant during the model simplification process avoids changes in the direction of the surface normals, and thus maintains the geometry of the model. The shape and position of the first-order neighbourhood triangle changes when a side is collapsed, as shown in Figure 2. The maximum angular change in the normal vector of the first-order neighbourhood triangle that occurs when the side L_i is collapsed to a point V_{i0} is called the normal vector error, denoted as α_i , and expressed as follows.

$$\alpha_i = \max \left\{ \arccos(n_{i0} \cdot n'_{i0}), \arccos(n_{i1} \cdot n'_{i1}), \dots, \arccos(n_{in} \cdot n'_{in}) \right\} \quad (11)$$

where $n_{i0}, n_{i1}, \dots, n_{in}$ denotes the unit normal vector of its first-order neighbourhood triangle before the edge L_i is folded and denotes the unit normal vector of its first-order neighbourhood triangle after the edge is folded.

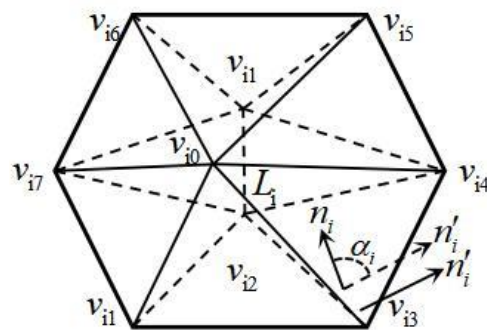


Figure 2. Triangular face piece flip.

3.3.4. Textures

In practical applications, the texture of a model is essential and is an important part of the visual experience. In order to better describe the importance of the local texture of the model, in the 3D architectural model simplification, this paper took the texture into account as a constraint in the error metric model. The error in the texture information for vertex V_0 can be calculated using Equation (12).

$$t_{V_0} = \frac{\sum_{i=1}^n \frac{S_{texV_0,i}}{S_{V_0,i}}}{n} \quad (12)$$

where n is the number of triangular faces in the first-order neighbourhood of vertex V_0 , $S_{V_0,i}$ denotes the area of the i th triangle in the first-order neighbourhood of vertex V_0 , and $S_{texV_0,i}$ denotes the texture area of the first triangle in the first-order neighbourhood of vertex V_0 . Assuming that (u_1, v_1) , (u_2, v_2) , and (u_3, v_3) denote the three texture coordinates of the triangular face piece, the texture area of the face can be calculated using Equation (13). However, this calculation is an estimate as the actual texture area may be affected by a variety of factors, such as texture repetition, stretching, etc.

$$S_{texv_0,i} = \frac{1}{2} \begin{vmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \end{vmatrix} \quad (13)$$

The formula expresses the importance of the texture in a given triangular surface, considering that if the texture area of that region is significant as a proportion of the geometric area, the simplification of that region will result in more texture loss. At the same time, the texture coordinates will keep the same update operation as the vertex coordinates during the simplification process.

3.4. Error Metric Model

When calculating the edge folding cost, the vertex approximate curvature, local area average, edge length, normal vector error, and texture are added to the error cost Δv , and the new obtained cost is taken as the final cost of edge folding. Considering that the spatial and textural coordinates of the vertices are needed in the calculation of the constraint factor, the error measure is divided into geometric and textural errors, and a reasonable weight is given to the combined measure. The error calculation formula can be expressed as follows.

$$Q = W_1 * Q_{geo} + W_2 * Q_{tex} \quad (14)$$

where Q_{geo} denotes the geometric error; Q_{tex} denotes the texture error; and W_1 and W_2 denote the weight coefficients of the geometric and texture errors in the error metric, respectively. After several tests, the best simplified results were obtained when W_1 was taken as 1 and W_2 was taken as 0.1. Note that in order to compare the geometric and texture errors at the same scale, we normalised the position coordinates of all the vertices in the model to be in the range [0, 1]. Since the range of texture coordinates is typically [0,1], the vast majority of models do not require additional processing of the texture coordinates. When folding an edge (V_i, V_j) , the folding cost can be calculated using Equation (15).

$$\Delta V = V^T (W_1 * Q_{geo,ij} + W_2 * Q_{tex,ij}) V = V^T Q V \quad (15)$$

where

$$V = [V_x \quad V_y \quad V_z \quad 1]^T$$

$$Q_{geo,ij} = l_{ij} \sum_{p \in P} K_p \times \left(\frac{C_{V_i}}{S_{V_i} \alpha_i} + \frac{C_{V_j}}{S_{V_j} \alpha_j} \right)$$

$$Q_{tex,ij} = \sum_{p \in P_i} K_p (t_{V_i} + t_{V_j})$$

$$Q = W_1 * Q_{geo,ij} + W_2 * Q_{tex,ij}$$

is the new vertex quadratic error matrix.

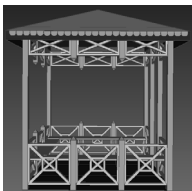
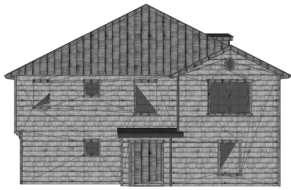
4. Simplification of the 3D Building Models Taking the Visual Features into Account

4.1. Overview

In order to verify the feasibility and effectiveness of the algorithms in this paper, the algorithms were verified using the Python programming language and the MeshLab platform to achieve a simplified display of the model. The machine tested was an Intel(R) Core(TM) 2 CPU 2.70 GHz with 16 GB of RAM and an operating system of Windows 11. The machine graphics card was an RTX 3050Ti.

The experimental data consisted of two models, a Chinese gazebo and a Chinese detached house. All the models were produced using 3ds Max with complex geometry and multi-chart textures. The basic information of these models is shown in Table 1. The experiments were simplified by collapsing the edges of these models at level three.

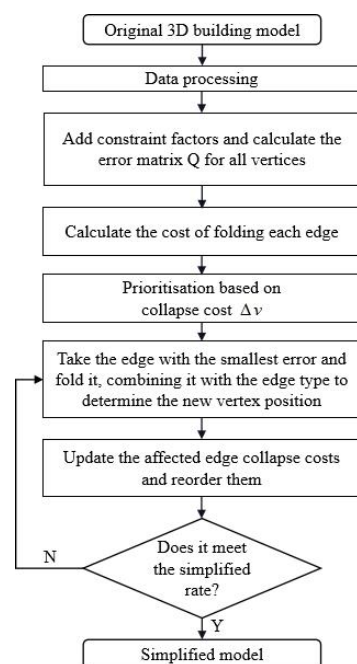
Table 1. Experimental data.

Model Name	Number of Vertices	Number of Faces	Thumbnails
Chinese Gazebo	2311	4002	
Chinese Detached House	3753	6484	

4.2. Algorithm Flow Design

For the 3D building models, newly generated vertices could destroy the geometric constraints, such as parallelism, coplanarity, and perpendicularity, making the quality of the simplified model degraded. On the contrary, since no new vertices were generated, the vertices of the simplified model were always a subset of the original model, and the geometric constraints of the model were better maintained. In addition, the half-folding did not need to calculate the position coordinates of the new vertices, which made it faster and more efficient when dealing with large data volumes for the 3D building models. Therefore, in this paper, the half-edge folding operation was chosen to simplify the model data.

The improved algorithm divided each edge into bounding and non-bounding edges during data pre-processing. When calculating the edge folding cost, it took into account the effects of the vertex approximate curvature, local area average, edge length, normal vector error, and texture, as well as the type of edge, and achieved model simplification by removing the edge length with the lowest folding cost. Figure 3 shows the specific flow design of the algorithm.

**Figure 3.** Algorithm flow chart of our method.

4.3. Data Processing

4.3.1. Vertex Classification

When we observe objects, we are more sensitive to key details such as angles, edges, and contours, and less perceptive of flat areas. The greater the change in undulation at a vertex, the greater the impact on the mesh model after removing that point, i.e., the greater the importance of the vertex. At the same simplification rate of the model, we first considered removing the points with a low vertex importance and keeping the points with a high vertex importance [29].

The vertices of the mesh model were classified according to their geometric importance and could be divided into the following four categories.

- Simple vertices: Vertices that were completely surrounded by their first-order neighbourhood triangle facets and where each edge using this vertex was used by only two triangle facets.
- Inner-edge vertices: Two neighbouring triangular face slices whose dihedral angles were not less than a set angle threshold and were treated as characteristic edges; a simple vertex was used by two characteristic edges.
- Corner vertices: Vertices that were used by three or more characteristic edges.
- Boundary vertices: Vertices located at the boundary of the mesh model.

4.3.2. Edge Classification

In 3D building models, the classification of boundary and non-boundary edges is usually judged based on the surface topology and geometry of the building model. In the modelling and analysis of building models, the delineation of boundary and non-boundary edges is important for the extraction of geometric information from buildings, the visualization and simulation of buildings, and the structural analysis of buildings [30]. The following are the definitions of the boundary and non-boundary edges.

- Boundary edge: A boundary edge is an edge that is adjacent to only one face on the surface of a building model. A boundary edge connects a face piece to the outside of the model or to an edge between the other face pieces. On the surface of an architectural model, a boundary edge is usually the outer contour boundary or demarcation line of a building.
- Non-boundary edge: A non-boundary edge is an edge that is adjacent to two or more faces on the surface of an architectural model. Non-boundary edges connect the edges between two or more face pieces. On the surface of an architectural model, non-boundary edges are usually the edges of parts of the building's interior such as walls, floor slabs, roofs, etc.

We further divided the non-boundary edges in the triangular mesh model into two categories based on the number of boundary vertices: transitional edges and simple edges. As shown in Figure 4, if only one of the two endpoints of an edge (v_1, v_3) was a boundary point, the edge was a transition edge. If none of the two endpoints of an edge (v_1, v_2) was a boundary point, the edge was a simple edge. In order to maintain the boundary characteristics of the model, we specified that when an edge was a boundary edge, no folding was performed on that edge. When an edge was a transition edge, the coordinates of the new vertex V were those of the boundary vertex. When an edge was a simple edge, the coordinates of the new vertex V were those of the more costly of the two endpoints to fold.

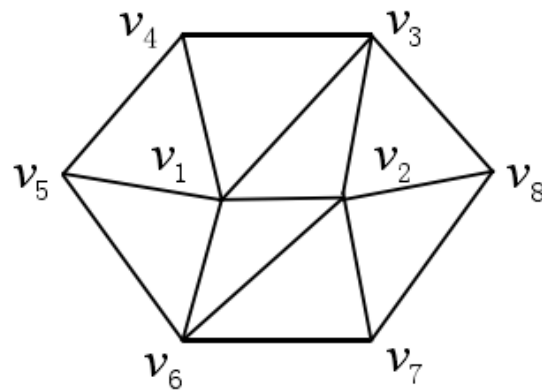


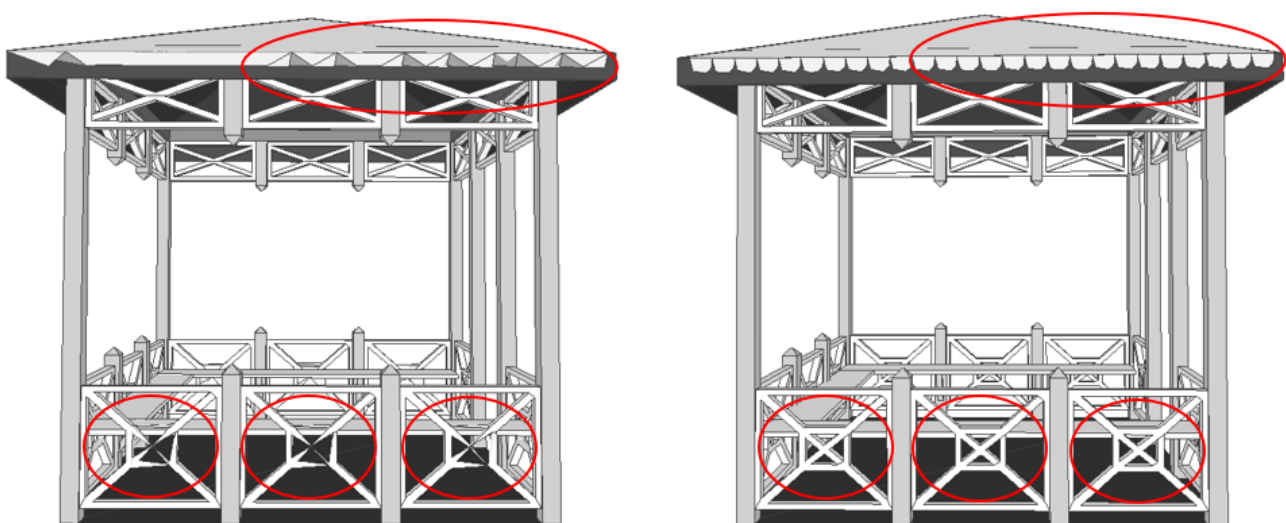
Figure 4. Grid model.

Assuming that the edge to be contracted was (v_1, v_2) , this edge was contracted to form a new vertex v_3 . If there were boundary vertices in v_1 and v_2 , then the position of the new vertex was taken to be that of the boundary vertex; otherwise it was the position of the more costly point to collapse in v_1 and v_2 . The boundary vertices were already marked during mesh pre-processing, and this strategy of choosing the position of the new vertex was one that affected the accuracy of the simplification, but it can be substantially more efficient. After generating the new vertex v_3 , the common triangles of v_1 and v_2 were first removed and all the remaining triangles were used as the neighbouring triangles of v_3 . The weights of the new vertex v_3 were then calculated and the weights of all the neighbouring vertices of v_3 were also recalculated.

By classifying the edges in the mesh model and adopting different methods of vertex determination for different edges, not only can the error metric of the model feature region be improved and the geometric features of the model be better preserved, but many operations can be avoided and the boundaries of the mesh model can be better maintained.

5. Results and Analysis

Both model data were simplified using the QEM algorithm and the optimised algorithm in this paper. The simplification results are shown in Figures 5 and 6. The statistics related to the simplification are shown in Tables 2 and 3 (both in msec). A comparison of the model simplification efficiency is shown in Figure 7. The error ratings and comparisons of the simplification results are shown in Tables 4 and 5, and Figures 8 and 9 (all in cm).



(a)

Figure 5. Cont.

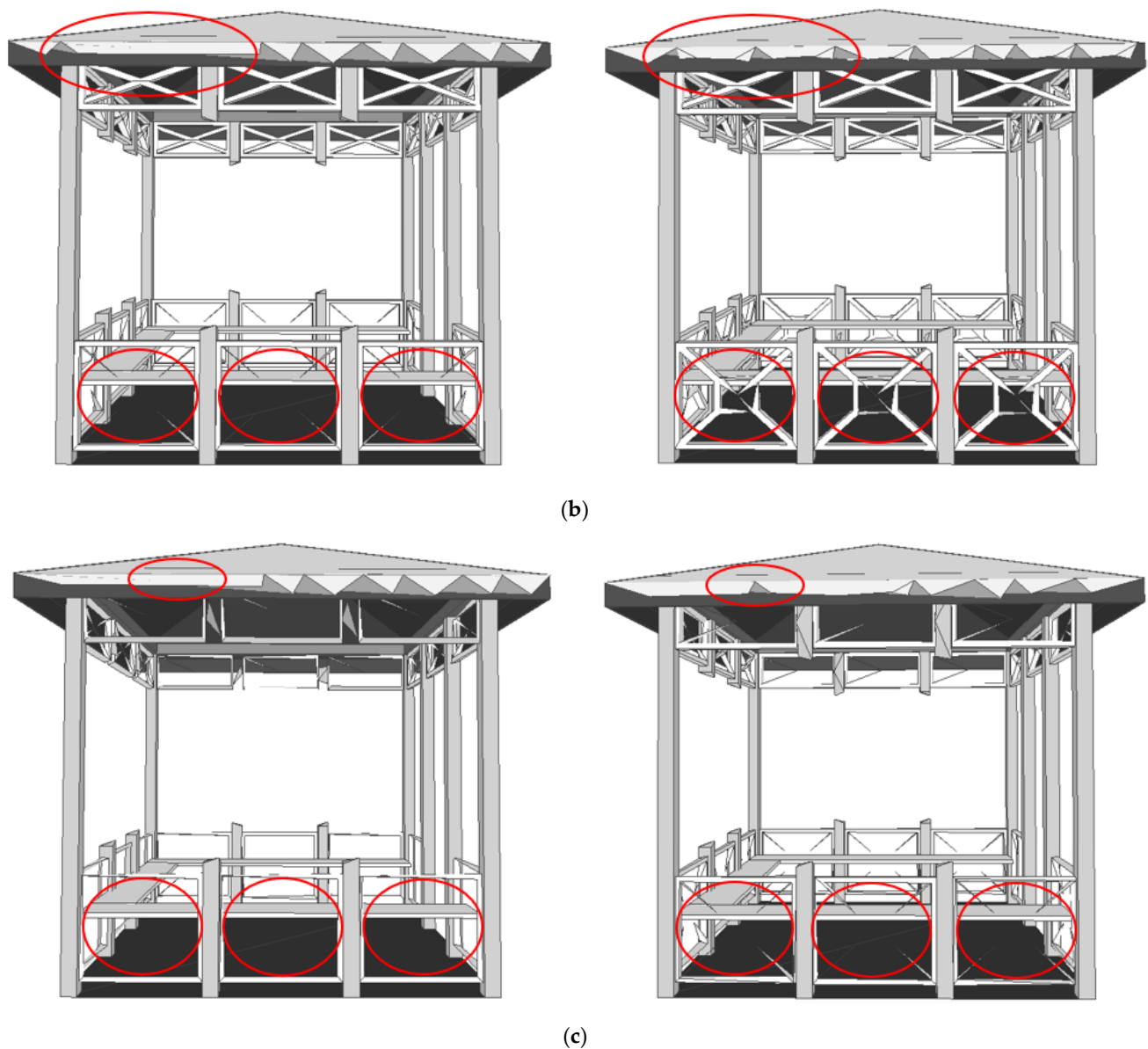


Figure 5. Simplified results of the Chinese gazebo model. (a) Simplification to 80%; (b) simplification to 50%; (c) simplification to 20%. The left side are all the simplified results using the QEM algorithm and the right side are all the simplified results using our method. The red circles mark the areas where there is a clear difference between the simplified results of the two algorithms.

Table 2. Simplified statistics of the Chinese gazebo model.

Simplification Level	Number of Faces	Simplification Rate	QEM Algorithm Time Consuming	Our Method Time Consuming
Level 1	4002	0	0	0
Level 2	3201	20%	47	85
Level 3	2001	50%	100	135
Level 4	800	80%	160	201

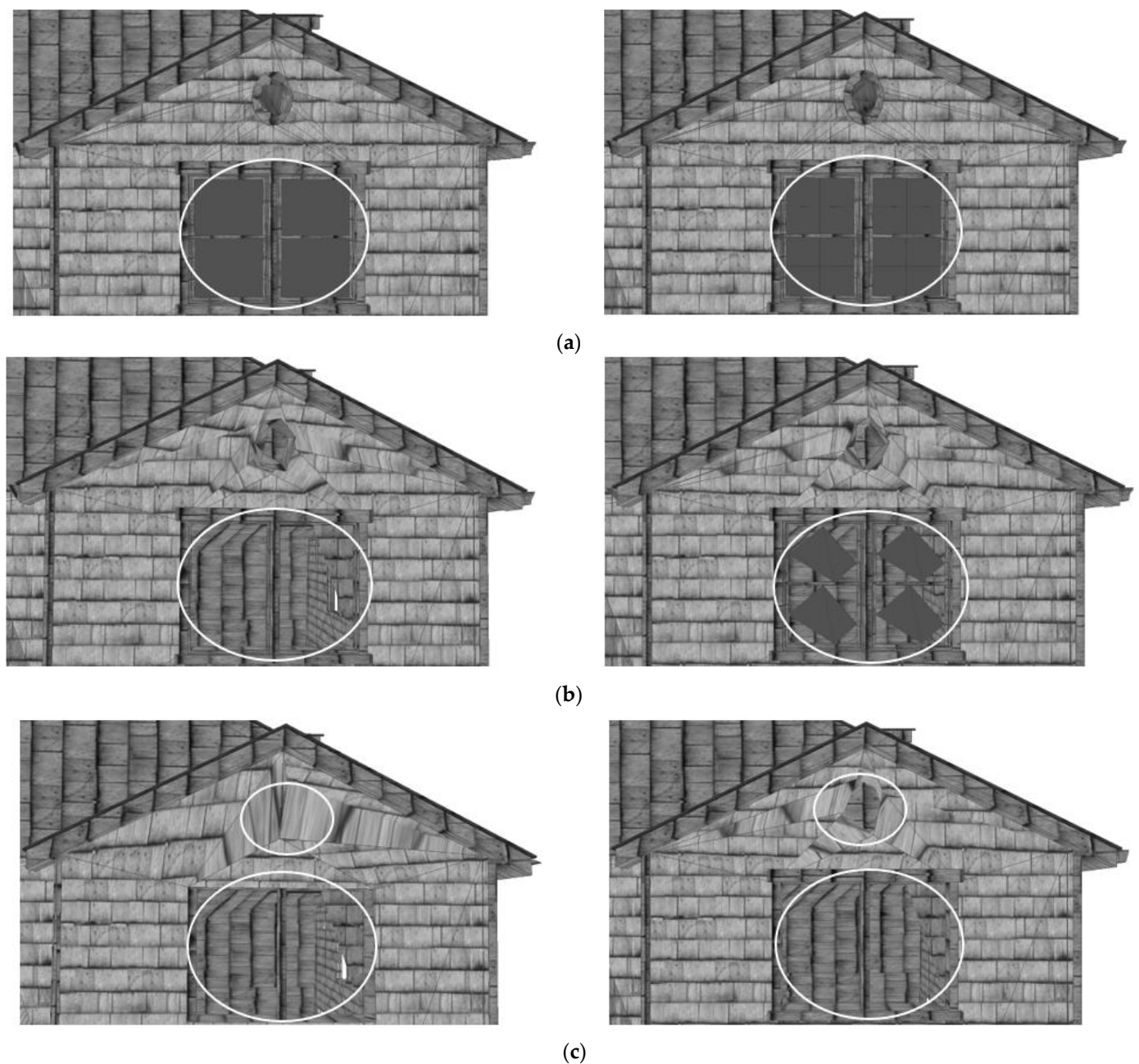
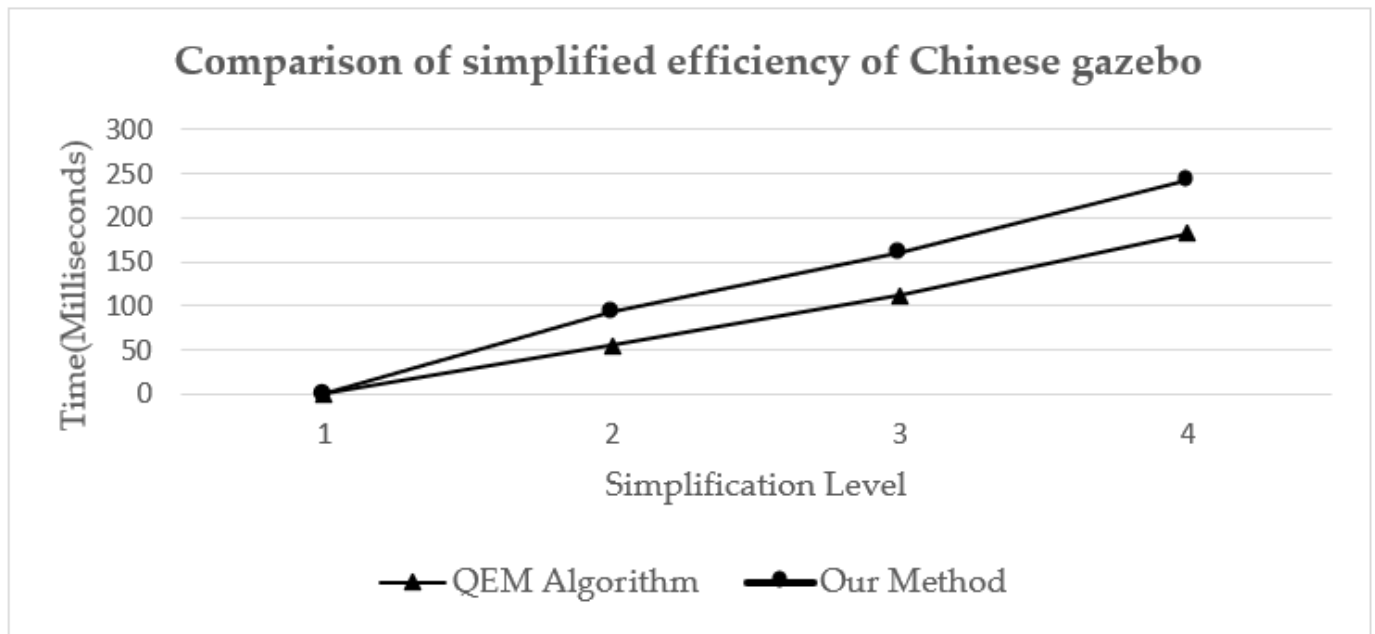


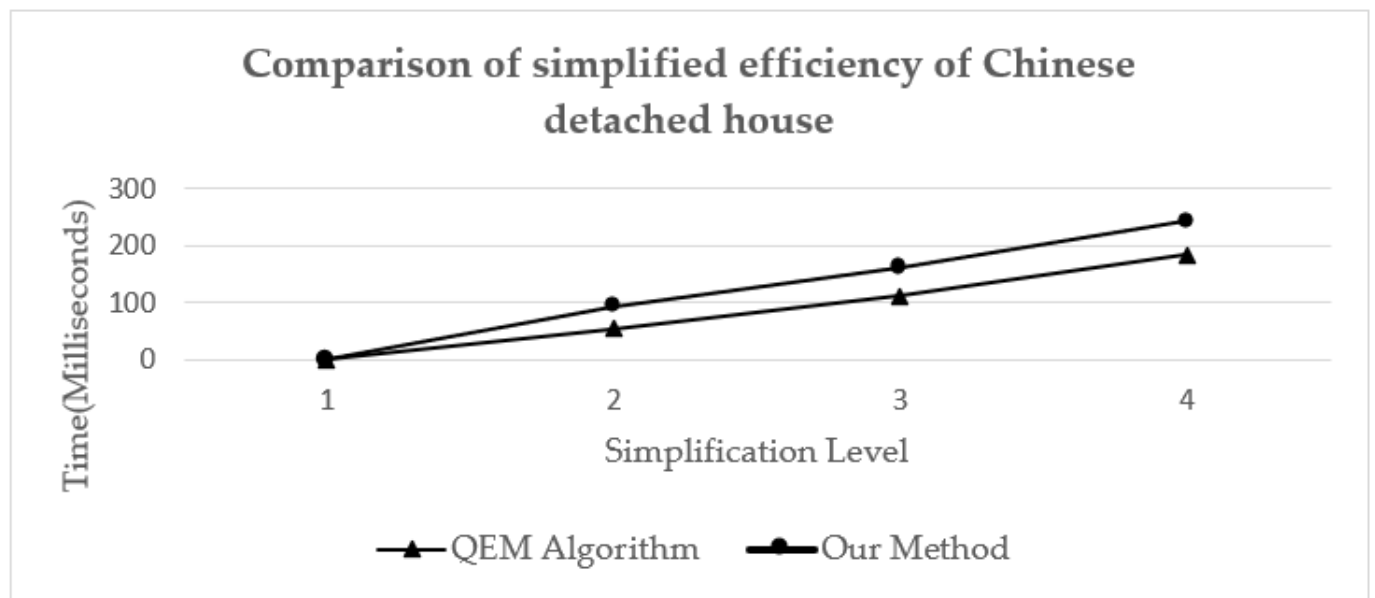
Figure 6. Simplified results of the Chinese single-family house model. (a) Simplification to 80%; (b) simplification to 50%; (c) simplification to 20%. The left side are all the simplified results using the QEM algorithm and the right side are all the simplified results using our method. The white circles mark the areas where there is a clear difference between the simplified results of the two algorithms.

Table 3. Simplified statistics for the Chinese detached house model.

Simplification Level	Number of Faces	Simplification Rate	QEM Algorithm Time Consuming	Our Method Time Consuming
Level 1	6484	0	0	0
Level 2	5186	20%	55	93
Level 3	3242	50%	112	160
Level 4	1295	80%	183	243



(a)



(b)

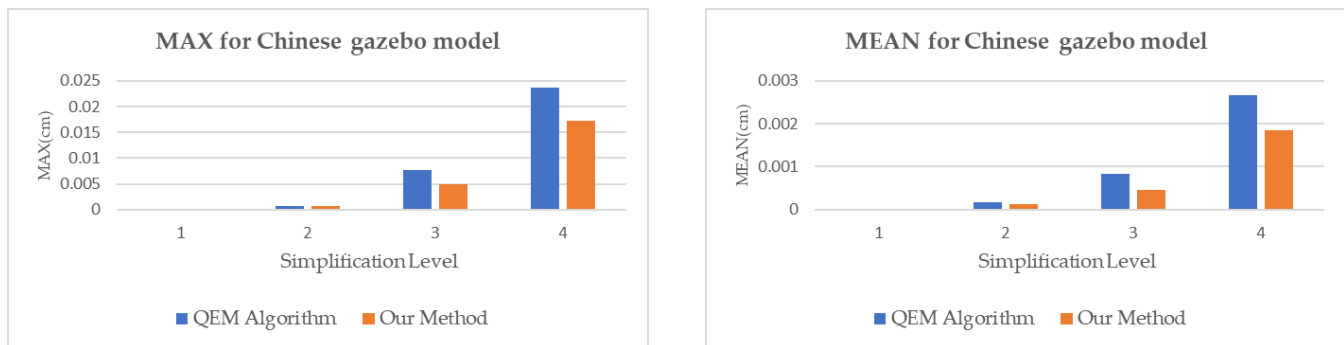
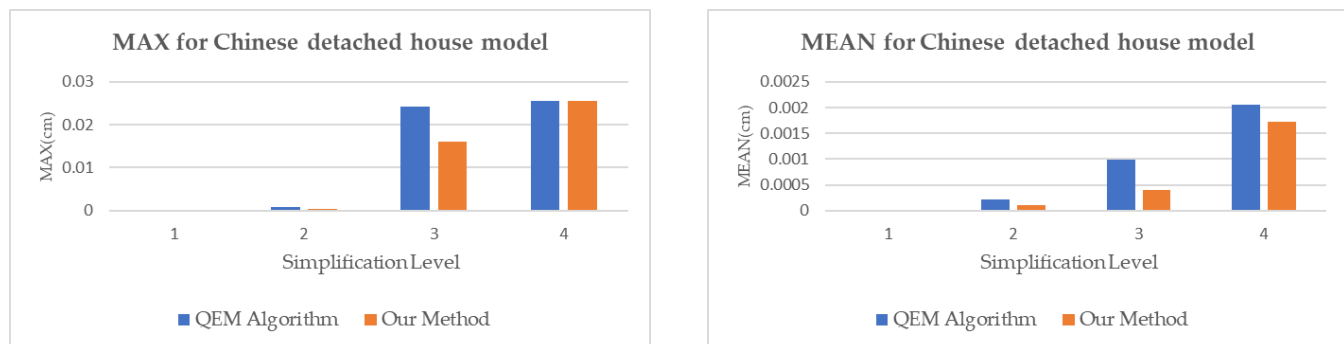
Figure 7. Comparison of the model simplification efficiency. (a) Comparison of the simplified efficiency of the Chinese gazebo model; (b) comparison of the simplified efficiency of the Chinese detached house model.

Table 4. Assessed values for the Hausdorff distance for the Chinese gazebo model.

Simplification Rate	Chinese Gazebo Model			
	QEM Algorithm		Our Method	
	MAX	MEAN	MAX	MEAN
20%	0.000759	0.000173	0.000759	0.000120
50%	0.007696	0.000838	0.004962	0.000462
80%	0.023669	0.002666	0.017283	0.001852

Table 5. Assessed values for the Hausdorff distance for the Chinese detached house model.

Simplification Rate	Chinese Detached House Model			
	QEM Algorithm		Our Method	
	MAX	MEAN	MAX	MEAN
20%	0.000867	0.000214	0.000481	0.000114
50%	0.024172	0.000991	0.016136	0.000395
80%	0.025508	0.002058	0.025494	0.001731

**Figure 8.** Simplified error comparison of the Chinese gazebo model.**Figure 9.** Simplified error comparison of the Chinese detached house model.

The aim of the 3D mesh model simplification was to reduce a complex mesh model of a triangular mesh to a simple mesh model of a triangular mesh and to maintain as much as possible the geometric characteristics of the original model. Therefore, the simplification results of different algorithms were assessed from two perspectives: a visual perspective and a quantitative perspective. The visual perspective was the human cognitive perspective, i.e., likeness, and the quantitative perspective was the quantification of geometric similarity using mathematical expressions [27].

Firstly, from the visual perspective, we compared the above simplified results with the local enlargement and found that the algorithm in this paper not only effectively reduced the amount of data in the grid model, but also retained more detailed features in the high feature areas, such as the windows, balustrades, and roofs. The texture distortion of the model was significantly reduced compared to the QEM algorithm in the same simplified ratio, which was more similar to the original model in terms of the visual effect.

According to the statistical table and comparison graphs, it can be seen that the algorithm in this paper was somewhat complicated in the process of calculating the folding cost. It slightly increased the running time of the simplification algorithm and maximised the grid description of the local detail features while ensuring a certain number of grids.

Again, from a quantitative perspective for error evaluation, this paper considered the Hausdorff distance proposed by P. Cignoni [31] in 1998 as an accuracy evaluation index

for the simplification effect, which took into account the error between the original to the simplified 3D mesh model along with the inverse error. The maximum geometric error represented the maximum value of the minimum distance between the corresponding points of the original and the simplified 3D grid model and is calculated as shown in Equation (16).

$$H(M, M') = \max(E(M, M'), E(M', M)) \quad (16)$$

where $E(M, M')$ represents the minimum distance from any one vertex in the original 3D mesh model M to all the vertices in the simplified 3D mesh model M' . The definition of $E(M', M)$ is symmetric to $E(M, M')$. Similarly, there was a minimum geometric error and an average geometric error. Since the minimum distance was necessarily zero, only the maximum and average values of the errors were counted in this article. In general, the Hausdorff distance of the simplified model should be as close as possible to the Hausdorff distance of the original model in order to ensure the quality of the simplification effect.

The comparison showed that at low simplification ratios, the difference between the simplification errors of this algorithm and the QEM algorithm was small, and the maximum and average values of the Hausdorff distances were also small. As the degree of simplification increased, especially when the simplification ratio reached 80%, it was obvious that the accuracy of the grid simplification algorithm of this paper was higher. This was because the simplification of flat areas had less influence on the Hausdorff distance, and the preservation of the detailed features made the simplification error lower. This shows that the algorithm in this paper has obvious advantages when performing 3D mesh simplification of urban scenes and can reflect the visual features of the model well.

6. Conclusions

In this paper, a half-edge folding simplification algorithm for 3D architectural models that took into account visual features was proposed. The improvements of this algorithm were mainly reflected in the region boundary constraints in the pre-processing stage and in the optimization of the error metric model in the calculation of the folding cost. Compared with the traditional edge folding algorithm, the algorithm proposed in this paper had better visual fidelity and can provide a reliable basis for the compression processing of 3D building models.

In future work, we will explore more effective error metric methods and compression processing strategies to achieve better compression results. In particular, we will further explore texture compression improvements to improve the network transmission efficiency and real-time rendering performance of 3D architectural models by processing texture images in the layers and hierarchies without compromising the visual quality. At the same time, we will also explore data compression algorithms for 3D building models in different scenarios in conjunction with practical application requirements to provide better support for the practical application of 3D building models.

Author Contributions: Conceptualization, L.H. and T.S.; methodology, H.B., L.H., T.S., X.W. and X.L.; software, H.B.; validation, H.B., X.W. and X.L.; formal analysis, H.B.; resources, X.W. and X.L.; original draft preparation, H.B., L.H. and T.S.; writing—review and editing, L.H. and T.S.; supervision, L.H.; project administration, X.W. and X.L.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Project of the Beijing Institute of Surveying and Mapping; The Science and Technology Plan of Beijing; Research and Application Demonstration of Key Technologies for the Basic Maps of Intelligent Connected Vehicle (The project number: Z211100004221015).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experimental data used in this article are all from the open data website officially provided by Shen Zhen, China. Available online: <https://www.3d66.com/> (accessed on 6 May 2023).

Acknowledgments: The authors are grateful to Xincheng Yang for his contribution to the analysis and supervision of this article.

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

References

1. Xiang, H.; Huang, X.; Lan, F.; Yang, C.; Gao, Y.; Wu, W.; Zhang, F. A Shape-Preserving Simplification Method for Urban Building Models. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 562. [CrossRef]
2. Chen, J.; Li, J.; Li, M. Progressive Visualization of Complex 3D Models Over the Internet. *Trans. GIS* **2016**, *20*, 887–902. [CrossRef]
3. Sun, Y.; Ma, J.; She, J.; Zhao, Q.; He, L. View-Dependent Progressive Transmission Method for 3D Building Models. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 228. [CrossRef]
4. Zhao, J. Automatic Simplification Approach for the LoDs of Complex 3D Building Models. Ph.D. Thesis, Wuhan University, Wuhan, China, 2012.
5. Clark, J.H. Hierarchical geometric models for visible surface algorithms. *Commun. ACM* **1976**, *19*, 547–554. [CrossRef]
6. Li, S.; Huo, L.; Shen, T.; Zhu, J.; Li, P.; Liu, H. A simplified algorithm for edge folding of 3D building model considering Angle error. *J. Wuhan Univ.* **2020**, *45*, 1282–1287.
7. Hoppe, H.; Deroose, T.; Duchamp, T.; McDonald, J.; Stuetzle, W. Mesh Optimization. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993, Anaheim, CA, USA, 2–6 August 1993.
8. 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; Volume 1997, pp. 209–216.
9. Lindstrom, P.; Turk, G. Fast and Memory Efficient Polygonal Simplification. In Proceedings of the Visualization Conference, Research Triangle Park, NC, USA, 8–23 October 1998.
10. Guskov, I.; Vidimčec, K.; Sweldens, W.; Schröder, P. Normal meshes. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000; pp. 95–102.
11. Kim, S.J.; Kim, C.H.; Levin, D. Surface simplification using a discrete curvature norm. *Comput. Graph.* **2002**, *26*, 657–663.
12. Silva, F.G.M.; Gomes, A.J.P. Adjacency and incidence framework: A data structure for efficient and fast management of multiresolution meshes. In Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, Melbourne, VIC, Australia, 11–14 February 2003; pp. 159–166.
13. Qu, L.; Meyer, G.W. Perceptually driven interactive geometry remeshing. In Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, Redwood City, CA, USA, 14–17 March 2006; pp. 199–206.
14. Wang, X.M.; Wu, J.Y.; Wu, E.H. A Novel Technique Based on Triangle Decimation for Tetrahedral Simplification. *Acta Electron. Sin.* **2007**, *35*, 2343.
15. Zhou, Y.; Zhang, C.; He, P. Feature Preserving Mesh Simplification Algorithm Based on Square Volume Measure. *Chin. J. Comput.* **2009**, *32*, 203–212. [CrossRef]
16. Menzel, N.; Guthe, M. Towards Perceptual Simplification of Models with Arbitrary Materials. *J. Comput. Graph. Forum* **2010**, *29*, 2261–2270. [CrossRef]
17. Li, F.; Zhao, D.; Li, Z. Multi-resolution model generation algorithm based on appearance preservation. *J. Syst. Simul.* **2012**, *24*, 62–66.
18. Yi, B.; Liu, Z.; Tan, J. Hierarchical quadratic error simplification algorithm for mesh model with boundary feature preservation. *J. Comput.-Aided Des. Graph.* **2012**, *24*, 427–434.
19. Zhang, S.; Zhao, J. Multi-resolution animation Model Generation based on Deformation distance Analysis. *J. Comput. Res. Dev.* **2012**, *49*, 1432–1437.
20. Dong, Y.; Zhang, Z.; Yang, K. Research on simplification method of conformal mesh based on vertex importance. *J. Comput. Eng. Des.* **2013**, *34*, 1889–1895.
21. Ozaki, H.; Kyota, F.; Kanai, T. Out-of-core framework for QEM-based mesh simplification. In Proceedings of the 15th Eurographics Symposium on Parallel Graphics and Visualization, Cagliari, Italy, 25–26 May 2015.
22. Dassi, F.; Ettinger, B.; Perotto, S.; Sangalli, L.M. A mesh simplification strategy for a spatial regression analysis over the cortical surface of the brain. *Appl. Numer. Math.* **2015**, *90*, 111–131. [CrossRef]
23. Duan, L.; Shao, H.; Li, Z.; Zhang, G.; Yang, S. Simplification method for feature preserving of efficient triangular mesh model. *Guangxue Jingmi Gongcheng/Opt. Precis. Eng.* **2017**, *25*, 460–468.
24. Wang, W.L.; Duan, L.M.; Wang, H.Y.; Bai, Y. Simplification and optimization of triangular mesh model based on dynamic error control and PSO. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 63–71.
25. 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]
26. She, J.; Chen, B.; Tan, J.; Zhao, Q.; Ge, R. 3D building model simplification method considering both model mesh and building structure. *Trans. GIS* **2022**, *26*, 1182–1203. [CrossRef]

27. Sun, Z. Research on 3D Model Simplification Algorithm and Progressive Transmission Method. Master's Thesis, University of Electronic Science and Technology, Chengdu, China, 2016.
28. Jia, Q.; Liu, Y.; Gu, X. Edge collapse mesh simplification algorithm based on detail features preserving. *J. Comput. Inf. Syst.* **2014**, *10*, 2883–2890.
29. Zhang, R.; Hu, S. Research Status of Simplification Algorithm for Triangular Mesh Model. *Digit. Technol. Appl.* **2018**, *1*, 2.
30. Jia, Q. Research on Mesh Simplification Algorithm Based on Improved Quadratic Error Measure. Master's Thesis, Yanshan University, Qinhuangdao, China, 2014.
31. Cignoni, P.; Rocchini, C.; Scopigno, R. Metro: Measuring Error on Simplified Surfaces. In *Computer Graphics Forum*; Blackwell Publishers: Oxford, UK, 1998; Volume 17, pp. 167–174.

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.