ISPRS Journal of Photogrammetry and Remote Sensing 173 (2021) 135–150

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Feature-preserving 3D mesh simplification for urban buildings

Minglei Li [a,*], Liangliang Nan [b]

[a] College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[b] Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, the Netherlands

ARTICLE INFO

Keywords:
Urban modelling
Mesh simplification
Mesh filtering
Edge collapse

ABSTRACT

The goal of urban building mesh simplification is to generate a compact representation of a building from a given mesh. Local smoothness and sharp contours of urban buildings are important features for converting unstructured data into solid models, which should be preserved during the simplification. In this paper, we present a general method to filter and simplify 3D building mesh models, capable of preserving piecewise planar structures and sharp features. Given a building mesh model, a mesh filtering technique is firstly designed to yield piecewise planar regions and extract crease contours. The planar regions are used to constrain the simplification of the mesh. Mesh decimation is achieved through a series of edge collapse operations, which uses regional structural constraints and local geometric error metrics to handle planar and non-planar areas respectively. The proposed method preserves the mesh structure with meaningful levels of detail while reducing the number of faces. The effectiveness of this method is evaluated on various building models generated from different observation scales, and the performance is validated by extensive comparisons to state-of-the-art techniques.

1. Introduction

There is an explosive growth of 3D scanning techniques, which have the capability of observing urban scenes from various platforms. 3D models of mesh type are always favored by many applications. A mesh model generated from point clouds by a surface reconstruction algorithm (Carr et al., 2001; Alexa et al., 2003; Shen et al., 2004; Guennebaud and Gross, 2007; Kazhdan et al., 2006) or from a photogrammetry pipeline (Furukawa and Ponce, 2010; Wu et al., 2011; Agisoft, 2020; Pixel4D, 2020) typically contains hundreds of millions of faces, which represents a huge burden on the visualization, storage, and data transfer of real-world applications. In some large-scale application scenarios, such as communication shielding analysis, augmented reality navigation, disaster assessment simulation, etc., the texture information and fine details of the model are not necessary. However, low memory footprint, fast data transfer, and efficient physical calculation analysis are always preferred in those applications. Therefore, it has become an urgent task to reduce the complexity of the models and meanwhile preserve the structure of scenes.

Extracting buildings from large scenes is crucial for many applications and great techniques have been developed for the extraction of single buildings (Lafarge and Mallet, 2012; Weinmann et al., 2015a; Niemeyer et al., 2016; Landrieu and Simonovsky, 2018). In this context, our method does not commit to segmenting the urban scene. Instead, our objective is to reduce the geometric complexity of single buildings and meanwhile preserve their structure based on a mesh simplification approach.

An efficient mesh simplification method should maintain the main structures of the mesh, although small-scale details might be lost. How to define the structure of a mesh is still an open question. However, it is common sense that the structure of an object is related to the graph of sharp features, smooth parts, or planar regions of the object (Botsch and Kobbelt, 2001; Mitra et al., 2013; Salinas et al., 2015). Since planar shapes are the most common shapes in urban buildings, our simplification method is intended to preserve these planar features while reducing the model sizes. For non-planar regions, our method seeks for piecewise planar approximations of the original geometry. Another problem is that geometric defects, such as noises and uneven density, are present in complex meshes. To eliminate these defects, mesh preprocessing plays an important role in the success of the final simplification. To reduce the amount of data, mesh decimation is one of the most versatile algorithms, which merges vertices by the edge collapse operations that minimize predefined geometric error metrics (Garland and Heckbert, 1997; Lindstrom and Turk, 1998). However, the existing error metrics are sensitive to noise and vertex density, so erroneous approximations may accumulate during iterations.

* Corresponding author.
E-mail addresses: minglei_li@nuaa.edu.cn (M. Li), liangliang.nan@gmail.com (L. Nan).

https://doi.org/10.1016/j.isprsjprs.2021.01.006
Received 4 April 2020; Received in revised form 7 October 2020; Accepted 4 January 2021
Available online 18 January 2021

The goal of this work is to build a fully automated pipeline that slims down an urban building mesh model while preserving its structure. The method is not aimed at dealing with an entire urban scene at once, but to process single buildings in the scene one by one. Our pipeline consists of three major modules, namely mesh filtering, structure extraction, and mesh decimation. The simplification can filter out small scale details, and the results can form multi-resolution hierarchies for efficient geometry processing as well as level-of-detail (LOD) generation.

The main contributions of this paper are as follows:

(1) A mesh filtering method that can progressively enhance piecewise smoothness and sharp contour features. This pre-processing method considerably improves the generality of the technique to different data sources. Based on this, a simple region growing algorithm can be exploited to extract planar shapes from the mesh. The structure of a mesh is represented by a topological graph, which is assembled by a set of planar regions and intra-contour edges detected after mesh filtering.
(2) A refined mesh decimation method based on a hierarchy strategy. Compared with traditional mesh decimation methods, we design error metrics for the edge collapse operators in planar and non-planar regions using different strategies. Planes detected in the previous step are used as a constraint to avoid erroneous accumulation during the edge collapse iterations. Therefore, the simplified model keeps the original plane structures as much as possible.

2. Related work

As many mesh modelling algorithms are in conjunction with point cloud processing, we discuss related techniques of model filtering and contour extraction in the fields of both mesh processing and point cloud processing.

2.1. Meshing and filtering

Various surface reconstruction algorithms have been designed to generate mesh models from points based on different mathematic theories, such as radial basis functions (Carr et al., 2001), moving least squares (Alexa et al., 2003, Shen et al., 2004, Guennebaud and Gross, 2007), and Poisson equation (Kazhdan et al., 2006). The surface definition determines which form of projection procedures is used or which formulation of implicit functions is taken. Even though both implicit surface reconstruction and projection-based reconstruction can preserve local smoothness of models, they tend to lose sharp features. These methods are concerned with the fidelity of the model rather than the compactness. The output models of these techniques can be the input data of our algorithm.

Model fitting with parameterized geometric primitives is another way for surface reconstruction, such as the methods based on Hough Transform (Overby et al., 2004; Vosselman et al., 2004) and random sample consensus (RANSAC) (Schnabel et al., 2007; Henn et al., 2013). Planar structures are widely used in many urban building modelling methods (Lafarge and Mallet, 2012; Xiong et al., 2015; Li et al., 2016, 2019), as they assume that the input mesh exhibits a large amount of near-planar parts. Chauve et al. (2010) used planes to partition the 3D space into a polyhedral cell complex, and the reconstructed surface is defined as the interface of a volumetric cell assignment. Nan et al. (2017) proposed using a combination of planar patches to generate a manifold polygonal surface model. As a constraint in their objective function, an edge is shared by two adjacent planar regions. However, the selection of planar patches highly depends on planar shape extraction, which is still sensitive to noise and sampling anisotropy, resulting in spurious hypotheses.

Given a mesh model, regularization-based methods are usually designed using some vertex moving criteria to adjust the mesh surface satisfying some requirements. Anisotropy parameterization-free projections (Lipman et al., 2007, Huang et al., 2009) use local approximate strategies to resample vertices to regularize meshes. He et al. (2013) introduced a face edge-based Laplacian operator, which calculates gradient information between neighbouring faces. Then they use an ℓ_0-based optimization to minimize the sum of the values of Laplacian over all triangle face edges. This method is effective for smoothing piecewise flat areas and preserving the sharp features. Inspired by the bilateral filtering used in image smoothing (Tomasi and Manduchi, 1998), Solomon et al. (2014) performed mesh denoising by filtering the face normals, using a general formulation of the bilateral filter. The challenge for regularizing meshes on the sharp areas is that the neighborhood employed for curvature estimation would enclose vertices belonging to different surface patches across the sharp feature.

The normals provide convenient descriptors for detailed geometric features of a mesh model. Early works on normal estimation usually need sufficiently big neighborhoods to average out the effects of noise. The scheme is based on an isotropic neighborhood radius, which implies that they easily smear sharp features during the smoothing process. A significant inaccuracy remains the determination of surface normals close to non-differentiable surface edges. We take triangle faces of meshes as operation units and select an adaptive anisotropic neighborhood for each unit only enclosing neighbouring faces located on the same surface patch.

2.2. Contour extraction

The detection of sharp creases is an ill-posed problem. For point cloud data, the covariance tensor of the neighborhood of a point is useful for describing the local dimensionality. In previous work (Yang et al., 2013; Weinmann et al., 2015a, 2015b; Wei et al., 2015), point features are designed based on different arithmetic combinations of the eigenvalues of the covariance tensor, and these features are further used to classify corners, edges, and planes. For instance, Hackel et al. (2016) used these eigenvalues to compose features and extract all features from multi-scale representations. Then, they train a random forest-based binary classifier to classify each point into contour versus non-contour. The contour points construct an over-complete graph of candidate contours, then an optimal set of contours is extracted using a further binary classification in a higher-order Markov random field.

Combining 2D imagery information with 3D data is an alternative way to extract 3D line segments. The multi-view stereo (MVS) technique is one of the most popular ways to reconstruct 3D models from image sequences (Furukawa and Ponce, 2010; Wu et al., 2011). Consequently, some methods extract line features in images and establish line correspondences, then triangulate them to 3D lines (Chen and Wang, 2011; OK et al., 2012; Hofer et al., 2015). The 3D lines are reconstructed by applying the soft matching method both on 2D images and a point cloud. In contrast, Lin et al. (2015) projected a point cloud into a collection of 2D images and extract the 2D line-support regions on these images. Then back-project them into the original point cloud as 3D line-support regions. However, these methods are tailored only for straight-line segments, so they cannot handle some curve contours in models. The quality of projected images will be unstable when the 3D point clouds have a strongly varying point density.

The methods for contour detection in 2.5D range images are more efficient because the data have a regular neighborhood structure. A distance threshold can quickly determine the depth discontinuity, hence provides some contour pixel candidates (Bormann et al., 2015; Choi et al., 2013). From a single image, Wang et al. (2016) introduced an approach to regularize 2.5D surface normal and depth predicted at each pixel. They infer coarse depths, surface normals, and planar boundary by convolutional neural networks (CNN), then a dense conditional random field is used to regularize the normals and depths with planar boundary constraint.

After extracting a set of irregular contour points, some contour

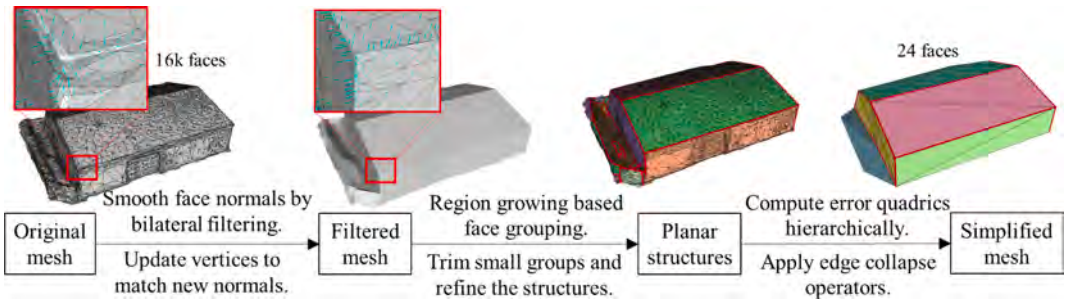


Fig. 1. The pipeline of our mesh simplification method.

refinement or regularization work is necessary to generate clear polygonal building contours (Rottensteiner et al., 2005; Xie and Feng, 2016; Chen et al., 2017; Li et al., 2019). Local linearity is one of the most widely used model assumptions. In general, it is still a challenge to obtain clean contours in curved areas using only geometric features.

2.3. Mesh simplification

A detailed model full of redundant triangle faces can be simplified by removing a large number of homogeneous faces. Edge collapse-based mesh decimation methods are very common in model simplification. The idea is that the greedy mesh decimation approaches use some local geometric error metrics to decide whether to delete the vertex or not. After removing vertices, all adjacent faces are re-triangulated (Garland and Heckbert, 1997). Lindstrom and Turk (1998) proposed a memory-efficient decimation method, which decomposes the error quadric matrix to a set of symmetric sub metrics. This method is more effective in terms of approximation error.

Considering the distinct characteristics of different regions of a model, Zelinka and Garland (2002) added boundary constrains on the decimation metrics. They define a volume in which the approximation must lie, and does not permit the simplification algorithm to work outside the volume. Similarly, Cohen et al. (2003) developed a piecewise linear mapping function based on known error bounds of the model for successive simplification operations. Marinov and Kobbelt (2005) proposed an integral error metric designed to derive multi-resolution meshes whose structures are aligned to some detected geometric elements. In (Salinas et al., 2015), the structure is abstracted by a set of planar proxies detected from a preprocessing step. From the set of detected proxies, they compute a proxy-based graph which constrains the mesh decimation process. Sometimes, the topology of the model cannot be maintained. This process can make drastic topological alterations to the model. BigSUR (Kelly et al., 2017) provides another way to get the simplified model. It fuses multi-source data, including coarse meshes of urban buildings, street images, and GIS footprints, to formulate a binary integer program that globally balances multiple sources of errors to produce compact facade structures. However, these multi-source data are not always available at the same time.

In this paper, the structure of a mesh is abstracted by a set of piecewise planar regions and their intra-contours, which are obtained after mesh filtering. This method is particularly useful for simplifying single buildings with sharp regions among facades and roofs. The detection of contours is performed at a spatial scale induced by a tolerance threshold of face-orientation discontinuity. To maintain the structure of the model and to avoid the error accumulation, we propose a refined mesh decimation method, which defines a hierarchical error metric to preserve the stability of the detected planar shapes during the subsequent edge collapse iterations.

3. Methodology

The input data to the algorithm is a triangulated mesh model that could be derived from point cloud by any surface reconstruction or modelling technique, such as dense photogrammetry or implicit surface reconstruction algorithms (Carr et al., 2001; Alexa et al., 2003; Shen et al., 2004; Guennebaud and Gross, 2007; Kazhdan et al., 2006; Furukawa and Ponce, 2010; Wu et al., 2011; Agisoft, 2020; Pixel4D, 2020). The proposed algorithm tackles the simplification problem as a three-phase task: (1) mesh filtering that suppresses noises and enhances piecewise smoothness; (2) planar region and feature extraction; (3) iterative mesh decimation based on a hierarchical error metric. The pipeline of the proposed method is illustrated in Fig. 1.

3.1. Mesh filtering

In the mesh filtering step, we decouple face normals and face vertex

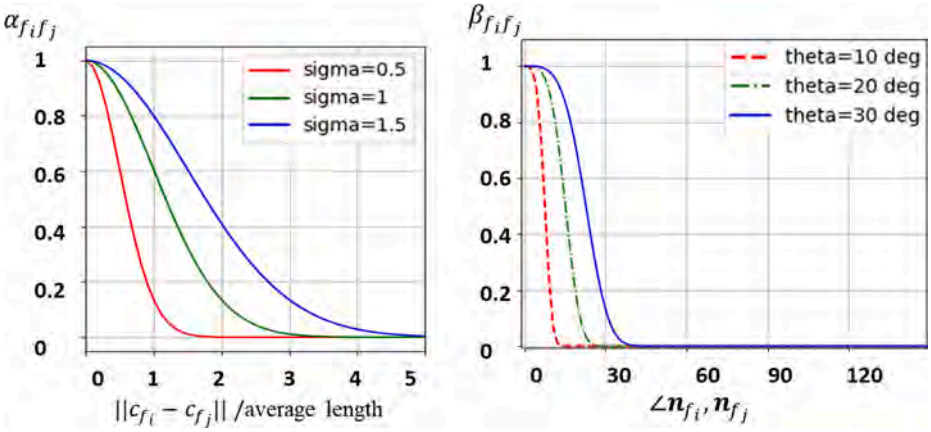


Fig. 2. The effect of the two weight terms. Left: $\alpha_{f_i f_j}$ decreases when the distances of faces increase. The horizontal axis and sigma values represent the ratio to the average edge length. Right: $\beta_{f_i f_j}$ decreases when the angle between normals increases. The horizontal axis indicates the angle between two neighbouring faces.

positions and refine them separately. Specifically, we use a bilateral filtering algorithm to smooth the normals of triangle faces, and then use the new normals to update mesh vertices. The mesh filtering program iterates between normal filtering and vertex updating. In the following, we first introduce the normal filtering method.

Face normal filtering. Given a mesh $\mathcal{M}$, we denote as $\boldsymbol{V}(\mathcal{M}) = \{v_i; i = 1, \cdots, n\}$ the set of vertices and $\boldsymbol{E}(\mathcal{M}) = \{e_{i,j} = v_j - v_i; v_i, v_j \in \boldsymbol{V}\}$ the set of edges. $e_{i,j}$ represents an edge that links two vertices v_i and v_j. As our basic operation targets are triangle faces, we define the initial normal of each face as the unit vector perpendicular to the face plane. Given a triangle face f_i, its initial normal is calculated by:

$$n_{f_i} = \frac{(v_2 - v_1) \times (v_3 - v_1)}{\|(v_2 - v_1) \times (v_3 - v_1)\|}, \tag{1}$$

where v_1, v_2, and v_3 are the vertices of f_i.

After initializing every face normal, a bilateral filtering algorithm is applied to refine the normal orientations. It involves two types of weights: (1) spatial distance-based weights $\alpha_{f_i f_j}$; and (2) normal proximity-based weights $\beta_{f_i f_j}$. The spatial distance of two faces, f_i and f_j, is calculated by the Euclidean distance between their centroids c_{f_i} and c_{f_j}. We denote $\boldsymbol{n}_{f_i}$ and $\boldsymbol{n}_{f_j}$ as the normal of the two faces. Both weight functions are defined in similar forms of the Gaussian function, as follows:

$$\alpha_{f_i f_j} = exp\left(-\frac{\|c_{f_i} - c_{f_j}\|^2}{2\sigma_{dist}^2}\right), \beta_{f_i f_j} = exp\left(-\frac{\|1 - \boldsymbol{n}_{f_i} \cdot \boldsymbol{n}_{f_j}\|^2}{(1 - cos\theta)^2}\right), \tag{2}$$

where σ_{dist} is the variance parameter of distance proximity; θ is a user-specified angle threshold that a smaller value yields a better differentiation. Both $\alpha_{f_i f_j}$ and $\beta_{f_i f_j}$ are non-negative functions. The values of $\alpha_{f_i f_j}$ decrease rapidly when the distance of neighbouring faces increases; $\beta_{f_i f_j}$ provides less influence when two normals are more different and vice versa. So, a small value of weight inhibits the two neighbouring faces from influencing each other. To better understand the effect of the two weights, we depict the curves of weight values to the similarities, as shown in Fig. 2. In our work, σ_{dist} is estimated by the average edge length of the mesh edges, and we set $\theta = 20°$ by default.

When extracting the neighborhood of a face, we should avoid including the faces across the crease areas. We use an adaptive topological neighborhood querying scheme. Each neighborhood set consists of multiple faces with similar normal directions. Let $\mathbf{N}_{f_i}$ denote the neighborhood set of f_i on condition that each element satisfies two criteria: (1) it shares at least one vertex with the query face f_i; (2) the dot product of neighbour's normal and normal of f_i is greater than $\cos\theta$. Besides, the query face itself is also included in $\mathbf{N}_{f_i}$. A filtered normal $\hat{n}'_{f_i}$ for face f_i is computed as:

$$\hat{n}'_{f_i} = \frac{1}{w_i} \sum_{f_j \in \mathbf{N}_{f_i}(f)} A_{f_i} \cdot \alpha_{f_i f_j} \cdot \beta_{f_i f_j} \cdot n_{f_j}, \tag{3}$$

$$w_i = \left\| \sum_{f_j \in \mathbf{N}_{f_i}(f)} A \cdot \alpha_{f_i f_j} \cdot \beta_{f_i f_j} \cdot n_{f_j} \right\|,$$

where w_i is a normalization term ensuring the result is a unit vector. A_{f_i} is the area of the face f_i. Intuitively, $\hat{n}'_{f_i}$ is a weighted average of normals within the neighborhood $\mathbf{N}_{f_i}$. For meshes with uneven face sizes, the topological neighborhood can provide better querying results than the range-based neighbor querying, as it only includes neighbouring faces with similar orientations. As can be seen from Eq. (3), the new surface normal $\hat{n}'_{f_i}$ is a weighted average of its neighbors. Since all the weights

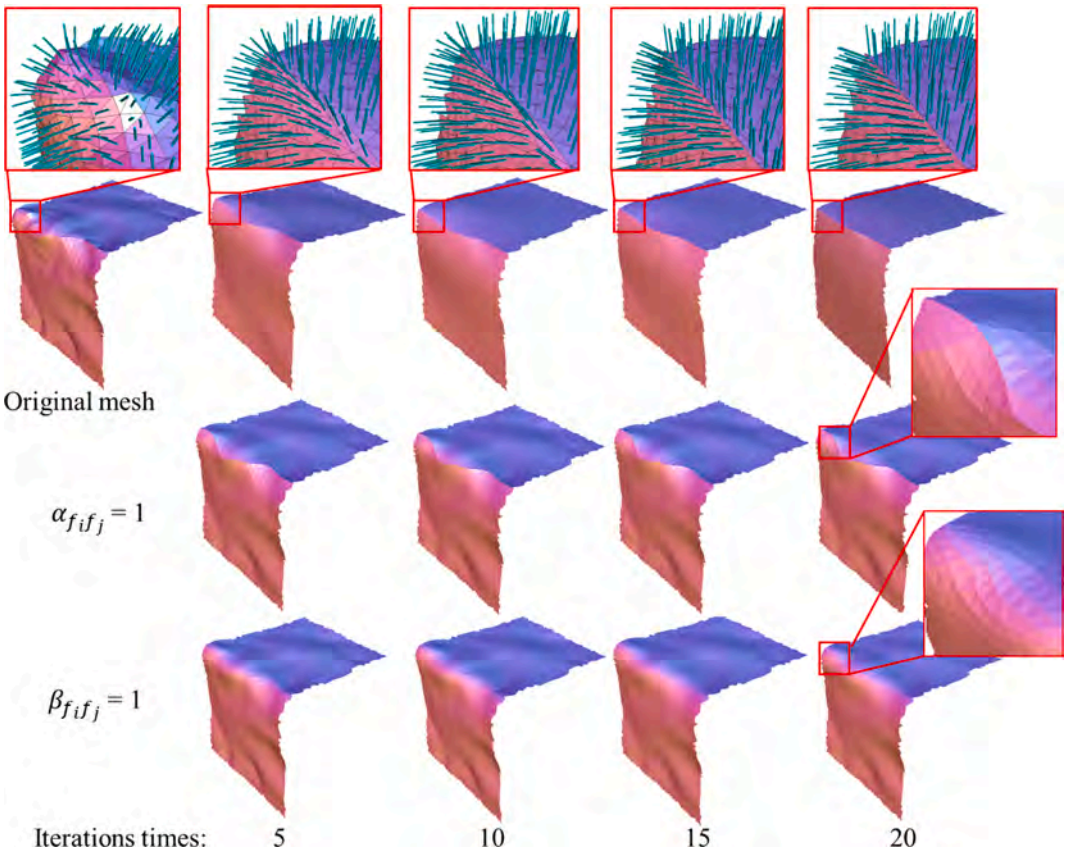


Fig. 3. The effect of the two weights on the filtering. The faces are color-coded based on face normal directions. The number below each column indicates the number of iterations.

are positive, Eq. (3) can be intuitively interpreted as a smoothing operation over the face normals, thus it inherently preserves the basic structure of the model.

Note that the initial normal n_{fj} can be extremely irregular if the initial mesh surface is highly crumpled. In this case, we use an isotropic geometrical neighborhood N_{f_i} at the beginning of the filtering. The isotropic geometrical neighborhood includes all neighbouring faces whose distances from the query face f_i are less than a range threshold. After a few iterations of the filtering process, the neighbouring normals will be homogeneous. Then, we use the aforementioned topological neighborhood query method for further filtering.

Vertex position update. After refining the face normals, the vertex positions need to be updated correspondingly. To calculate vertex displacements, we design a variation of the discrete anisotropic Laplacian operator inspired by HA (Huang and Ascher, 2008). In the original method from HA, the discrete anisotropic Laplacian (AL) operator is given by

$$\Delta_{v_i} = \frac{1}{w_{v_i}} \sum_{v_k \in \mathbf{N}_{v_i}(v)} exp\left(\frac{-{h_{i.k}}^2}{2\sigma_i^2}\right) h_{i,k} \cdot n_{v_i}, \tag{4}$$

$$w_{v_i} = \sum_{v_k \in \mathbf{N}_{v_i}(v)} exp\left(\frac{-{h_{i.k}}^2}{2\sigma_i^2}\right), h_{i,k} = e_{i.k} \cdot n_{v_i},$$

where $v_k \in \mathbf{N}_{v_i}(v)$ denotes the neighbors of a vertex v_i; $h_{i.k}$ denotes the projection of an edge $e_{i.k}$along the normal n_{v_i}; σ_i is the deviation of edge projections; and n_{v_i} is the normal of v_i. w_{v_i} is used to normalize the influences of edge projections. Eq. (4) averages the neighbouring edges with anisotropic weights.

We also use weighted averages but depart from HA in the sense that our implementation is based on face normals rather than vertex normals. As a non-boundary vertex has at least three faces in its one-ring neighbors referred to as face neighborhood $\mathbf{N}_{v_i}(f)$, they provide reliable guidance for the vertex updating. In our implementation, the connected segment between the face center c_{f_k} and the vertex v_i, i.e.$c_{f_k} - v_i$, is used to replace $e_{i.k}$ in the projection $h_{i.k}$. Besides, our experiments and analysis have shown that there is no obvious difference by using exponential kernel on $h_{i.k}$, so the projection value is normalized by dividing the number of neighbouring faces. Thus, our vertex displacement value Δ_{v_i} is calculated by

$$\Delta_{v_i} = \frac{1}{|\mathbf{N}_{v_i}(f)|_0} \sum_{f_k \in \mathbf{N}_{v_i}(f)} \left[\left(c_{f_k} - v_i\right) \cdot \hat{n}'_{f_k}\right] \cdot n_{f_k}, \tag{5}$$

where $|\mathbf{N}_{v_i}(f)|_0$ gives the number of neighbouring faces; $f_k \in N_{v_i}(f)$ denotes f_k one of the neighbouring faces of v_i, and $\hat{n}'_{f_k}$ is the filtered normal of f_k.

An iteration is finally accomplished by vertex updating: $v_i \leftarrow v_i + \Delta_{v_i}$. Then the updated vertices can generate new face normals for the next iteration. According to Eqs. (1), (3), and (5), we iterate the three steps of face normal initialization, normal filtering, and vertex updating until convergence is reached. The theoretical convergence criterion is that for every vertex the displacement is within a specified threshold.

The optimization result of the filtering operation should satisfy $\min\sum_{\forall n_{f_k}} \hat{n}_{f_k} \Pi_{f_k}$, where Π_{f_k} is the plane coplanar with a face f_k. However, the vertex update strategy approximates the optimization result. So, after each iteration, the vertex updating and normal smoothing affect each other and they are difficult to converge completely. Extensive experiments revealed that after 10 iterations, a plausible model flatness can be achieved. For efficiency, we set the number of iterations to the range [10, 20]. In practice, a large number will also work but it takes more time.

Fig. 3 illustrates the concept behind the proposed approach on a ridge area of a building model. The faces are color-coded based on their normal orientations. Due to the noise, the initial mesh surfaces around the ridge area are bumbled, and the orientation distribution of face normals is quite fuzzy. After our normal filtering and vertex updating (first row), these problems have been improved significantly. For this model, the change of the mesh became negligible and the contour between the two regions became clear after 15 iterations. The filtered mesh provides fundamental support for the later feature extraction.

To demonstrate the impact of both the normal similarity weight and spatial weight, we show the smoothing results by omitting one of the weights (i.e., setting its value to 1). As shown in the second row of Fig. 3, where only the normal similarity weight is used to filter the mesh, the face normals converge rapidly. Although there are sharp discontinuities in the ridge area, the contour has increased tortuosity. This is because using only the similar normals in the neighborhood strengthens the orientation of each other, which has an effect of sharpening the transition regions. The third row shows that using only the spatial weight degenerates the formula into a simple range-weighted mean filtering, which leads to over smoothing in the ridge area.

3.2. Region and contour extraction.

Planar region extraction. After filtering the 3D model, the topological relationship of its triangular faces does not change, and the characteristics of the smooth region and the edge region of the model are further enhanced. On this basis, a simple region growing method can get good plane segmentation results. To get piecewise planar regions, we segment triangle faces into some regions by traversing triangle faces through their common edges to compare their normal directions. Neighbouring triangles are merged into a group if their normals are close enough. A region stops expanding when no more faces satisfy the merging conditions, and then a new growth stage starts from the remaining faces. After the region growing, we obtain a set of coarse planar regions and every triangle face has been assigned to one of these regions. A detailed description of the region growing method is given in **Algorithm 1**.

To further analyze these regions, especially their spatial adjacent relationships, we build a topology graph $\boldsymbol{G} = (\boldsymbol{V_G}, \boldsymbol{E_G})$ without considering their geometric positions. It is similar to the graphs proposed in aerial building roof analysis tasks (Oude Elberink and Vosselman, 2009; Xiong et al., 2014). In the graph $\boldsymbol{G}$, each node denotes a region $v_i := R_i$ and the edge between two nodes means that two regions are adjacent. To build the set $\boldsymbol{E_G}$, we traverse all face edges. If two regions $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$ have faces that share a common edge, we consider the two regions adjacent and insert a graph edge $\boldsymbol{e}_{ij}$ in $\boldsymbol{E_G}$.

Using the topology graph, we further refine the planar regions and detect intersecting region contours. As the region growing algorithm only considers normal proximity, there are two types of regions that need to be refined: (1) some very small groups with few faces, and (2) some separated similar regions that should be connected.

Generally, very small groups will be considered invalid. If the area of a region $\boldsymbol{R}_i$ is smaller than a specified area threshold, the faces in $\boldsymbol{R}_i$ need to be redistributed to one of its adjacent regions $\{\boldsymbol{R}_j\}$. To find the target region $\boldsymbol{R}_{\text{target}}$, we compute the summation of the projected distances from face vertices $\{v\}_{\boldsymbol{R}_i}$ to the plane $\Pi_{\boldsymbol{R}_j}$ of the candidate region, according to Equation (6).

$$\boldsymbol{R}_{\text{target}} = \text{argmin}\left(\sum_{\{v\}_{\boldsymbol{R}_i}} \left| v, \Pi_{\boldsymbol{R}_j} \right|\right) \tag{6}$$

If the value of the smallest summation is smaller than a specific fitting threshold, all faces in $\boldsymbol{R}_i$ are regrouped into $\boldsymbol{R}_{\text{target}}$; otherwise, the faces remain ungrouped. Hence, we allow ungrouped faces, which are lying on highly curved regions or they are outliers.

After eliminating very small regions, the fusion regions inherit their graph edges. Separated regions that should be together are mainly caused by these sandwiched small regions. After trimming small regions

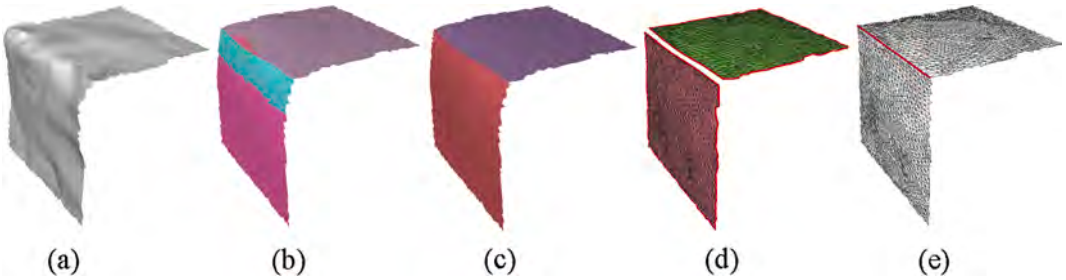


Fig. 4. An overview of the feature extraction process. (a) Initial mesh; (b) preliminary classification based on region growing; (c) the segmentation result after merging similar regions; (d) two refined planar region structures; and (e) the intra-contour edge depicted in red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in $\boldsymbol{G}$, we check and merge the regions that are near coplanar and adjacent. After all, a plane is fitted by least-squares (LS) fitting from all related face vertices for each planar region.

The pseudo-code of the planar structure extraction is given in **Algorithm 1**.

Algorithm 1. Planar structure extraction framework
Input: Original mesh
1) Initialize a group indicator for every face $g(f) = \mathrm{g} = -1$, and put all faces in a global set $\boldsymbol{A} = \{f\}$;
2) While $\boldsymbol{A} \neq \varnothing$ do:
3) f_i = A.front(), erase f_i from $\boldsymbol{A}$, and $g(f_i) = ++\mathrm{g}$;
4) Define a new group $\boldsymbol{R}_g = \varnothing$, and insert f_i into $\boldsymbol{R}_g$;
5) Find all neighbor faces $\mathbf{N}_{f_i}$ of f_i;
6) For each $f_k \in \mathbf{N}_{f_i}$ do:
7) If$\mathrm{g}(f_k) \neq -1$, erase f_k from $\mathbf{N}_{f_i}$, continue;
8) Else if $\hat{\boldsymbol{n}}'_{f_k} \cdot \hat{\boldsymbol{n}}'_{f_i} > \sin_0(\theta)$,set $g(f_k) = \mathrm{g}$, insert $f_k$into $\boldsymbol{R}_g$,find neighbors $\mathbf{N}_{f_k}$ of f_k, erase f_k from $\boldsymbol{A}$, and concat $\mathbf{N}_{f_i}$ with $\mathbf{N}_{f_k}$.Go to 6).
9) Mark $\boldsymbol{R}_g$ as a planar region, then go to 2);
10) End loop 2). Get coarse regions $\{\boldsymbol{R}_g\}$.
11) Construct a coarse region-based graph $\boldsymbol{G} = (\boldsymbol{V}_G, \boldsymbol{E}_G)$;
12) For each node/region $v_i := R_i$ in $\boldsymbol{V}_G$, do:
13) If area($\boldsymbol{R}_i$) > area threshold, continue;
14) Else, find target region $\boldsymbol{R}_{\text{target}}$ according to Equation (6)If the smallest sum < fitting threshold,merge $\boldsymbol{R}_i$ into $\boldsymbol{R}_{\text{target}}$;Else mark faces in $\boldsymbol{R}_i$ as ungrouped.Go to 12).
15) Fitting planes by LS for every region.
Output: Refined planar regions with supporting face indices.

Contour feature extraction. The contour features of a mesh are coincident with some triangle face edges, along which the orientations of the joint faces show significant discontinuities. Similar to the process searching graph edges, we traverse all face edges and check whether the adjacent faces of an edge belong to different regions. Specifically, an edge is a contour edge if its two adjacent faces belong to different regions. An intersection line l can be calculated from two corresponding region planes for the contour edge. Then, the two vertices of the contour edge are projected to line l.

Fig. 4 illustrates the feature extraction procedure using the mesh in Fig. 3. The region growing method classified the triangle faces into some homogeneous parts, as shown in (b). Faces with an identical color correspond to a separate region. As mentioned above, a small normal proximity angle tolerance, e.g. 15 degrees, will split some similar regions. That is why we can observe a few patches marked as different regions in the preliminary classification. Using the region refinement strategy, these similar regions are merged with adjacent regions to get a coherent segmentation result, as shown in (c). In (d) and (e), the final extracted planar regions and their intra-contour edges are demonstrated. The extracted features faithfully reflect the characteristics of the scene.

3.3. Simplification of mesh models

Conventional edge collapse. Mesh simplification is achieved by finding a tradeoff between model accuracy and geometric complexity. This is realized by a mesh decimation framework that iteratively applies the edge collapse operation. Edge collapse is an operator that merges the two vertices v_i and v_j to a unique vertex $\widehat{v}$, i.e., $v_i, v_j \rightarrow \widehat{v}$. For an edge expressed by $e_{i,j} = v_j - v_i$, a cost $\Delta_{e_{i,j}}$ is defined related to an error metric to decide whether the two vertices should be merged.

In the original edge collapse operator of the GH method (Garland and Heckbert, 1997), each vertex is associated with an error quadric Q_v, which represents an error between the current and the initial mesh. The method minimizes the sum of the projected distances from the vertices to their neighbouring face planes $N_v(f)$. Given a plane $P = [a, b, c, d]^T$, the squared distance of a point $v = [x, y, z, 1]^T$ to the plane is:

$$\left(P^T v\right)^2 = v^T \left(PP^T\right) v =: v^T Q_P v, \tag{7}$$

where $Q_P = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$

Further, the error quadric Q_v of a vertex v accumulates the squared projected distances from v to all supporting planes$\{P_{f_k}\}$, which are coplanar with v's neighbouring triangles $N_v(f)$:

$$\sum_{P_{f_k} \in N_v(f)} \left({P_{f_k}}^T v\right)^2 = \sum_k v^T Q_{P_{f_k}} v = v^T \left(\sum_k Q_{P_{f_k}}\right) v =: v^T Q_v v. \tag{8}$$

This quadric Q_v is encoded as a 4×4 symmetric matrix. For an edge $e_{i,j}$, its quadric $Q_{e_{i,j}}$ is computed by adding the quadrics of its two vertices, i.e., $Q_{e_{i,j}} = Q_{v_i} + Q_{v_j}$. So, the cost function of an edge $e_{i,j}$ is calculated as:

$$\Delta_{e_{i,j}} = \widehat{v}^T Q_{e_{i,j}} \widehat{v}. \tag{9}$$

$\widehat{v}$ is the variable, which represents a new vertex. When$\widehat{v}$ minimizes the cost $\Delta_{e_{i,j}}$, it is the result position for edge collapse operation on $e_{i,j}$. As every edge has a minimized cost with a corresponding potential vertex, we can list all the edges in ascending order according to their costs. The algorithm iteratively merges the end points of the edge with the lowest cost, computes its optimal location, and updates the edge list.

Notice that, in Eq. (8), the quadric Q_v of a vertex v is calculated based on all neighbouring faces of v. This method does not consider the regional structure. Consequently, the quadric of the edge has the same problem, which might lead to local convergence.

Feature-preserving edge collapse. Our approach analyzes neighbouring faces and regional structures to design a hierarchical error quadrics for the edges. Extracted planar regions are used to define a new error metric.

If all neighbouring faces of v belong to one planar region P_v, we calculate the error quadric Q_{P_v} of v based on P_v rather than the neighbouring faces of v. Here, Q_{P_v} is a planar region constrained error quadric. Otherwise, we keep using the conventional error quadric, i.e. the sum of the projected distances from v to all neighbouring triangles' planes (see

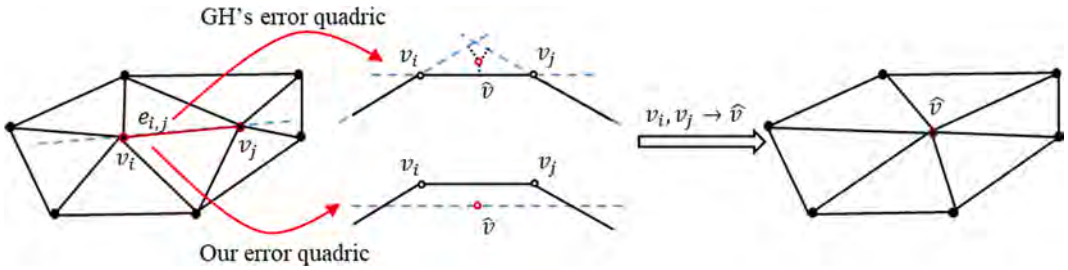


Fig. 5. Examples of edge collapse operation ($v_i, v_j \rightarrow \hat{v}$), using GH's error quadric (middle up) and regional plane constrained error metric (middle bottom). The dash lines denote supporting planes P_f.

Eq. (8)). Hence, the vertex quadric now is designed as a hierarchical formula:

$$Q_v = \begin{cases} Q_{P_v} if \ \forall P_{f_k} \ belong \ to \ P_v \\ \sum_{P_{f_k} \in N_v(f)} Q_{P_{f_k}} otherwise \end{cases}, \tag{10}$$

For an edge, its quadric $Q_{e_{ij}}$ accumulates the two vertex quadrics. The error metric is defined as:

$$\Delta_{e_{ij}} = \hat{v}^T Q_{e_{ij}} \hat{v} = \hat{v}^T \left(Q_{v_i} + Q_{v_j} \right) \hat{v}. \tag{11}$$

The next step is to find $\hat{v}$ that minimizes $\Delta_{e_{ij}}$. We adopt the GH's placement strategy for v. As $\Delta_{e_{ij}}$ is quadratic, an optimal location for $\hat{v}$ can be found by solving $\partial\Delta/\partial x = \partial\Delta/\partial y = \partial\Delta/\partial z = 0$. Taking partial derivatives of Equation (11), $\hat{v}$ is the root of the following linear system:

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{12}$$

where q_{ij} is the corresponding element in $Q_{e_{ij}}$. If the coefficient matrix in this function is invertible, we select either $\hat{v} = v_i$, $\hat{v} = v_j$, or $\hat{v} = (v_i + v_j)/2$ depending on which that produces the lowest value of $\Delta_{e_{ij}}$.

The impact of planar region groups on edge collapse operators is illustrated in Fig. 5, where the dash lines denote supporting planes. The operator using GH's error quadric generates $\hat{v}$, which has the lowest total distance value to the planes of neighbouring triangles. It only uses local geometric information of neighbouring faces, which might be influenced by noises. If the neighbouring faces are near planar on a larger scale, the result cannot reflect this characteristic. On the contrary, as our error metric considers the planar region group as a whole, the new vertex $\hat{v}$ will automatically migrate to the plane.

The hierarchical definition of error metrics for the edge collapse operations ensures that the proposed approach is more adaptable for different data. For example, if a building is composed entirely of planar structures, the error metric in the algorithm is only subject to the regional planar constraints. Conversely, if a building does not contain a plane structure but only curved shapes, the edge collapse operation degrades into the traditional method.

In practice, we use a greedy optimization method that first computes a priority queue of edge collapse operators with increasing cost values. The collapse operator with the lowest cost from the heap is extracted and applied. Then, the priority queue is updated and the algorithm iterates the above processes until the reduction requirement is satisfied.

We observe in different experiments that when we set the target number of mesh vertices to about twice the number of extracted planar regions, the result models can reach a better tradeoff between model fidelity and conciseness. However, automatically setting the optimal vertex number of simplified mesh is not easy due to the varying complexity of the scenes. It may require trial and error to better balance the geometric accuracy and the simplicity of the final model. In Section 4.4, we show some examples to discuss the effects of target vertex numbers.

4. Experimental results

4.1. Qualitative results

To evaluate the efficiency of the proposed method, we applied it to a

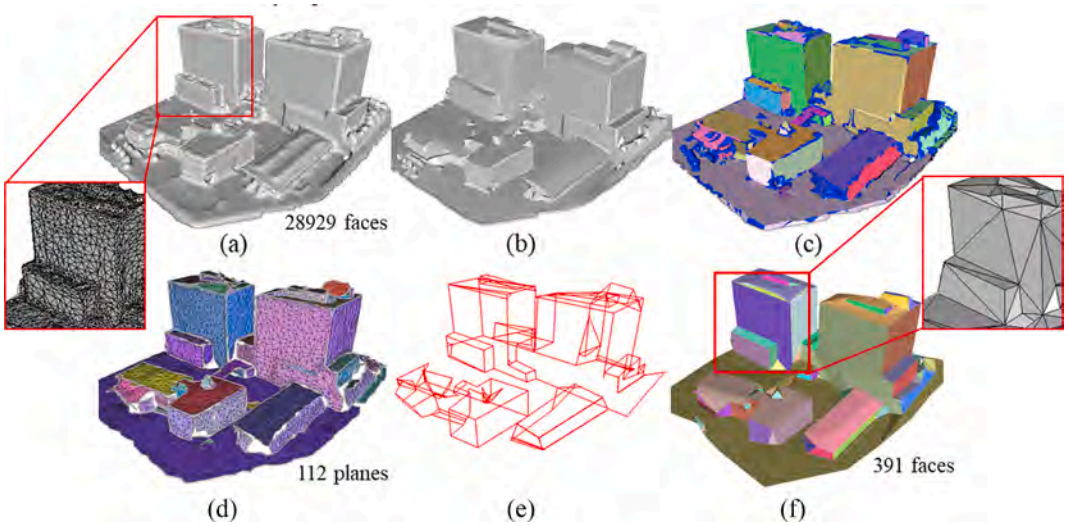


Fig. 6. Simplification of an aerial photogrammetry model. From (a) to (f), initial mesh, mesh filtering result, region growing result, extracted planar region structures, contour features, and the final simplification result.

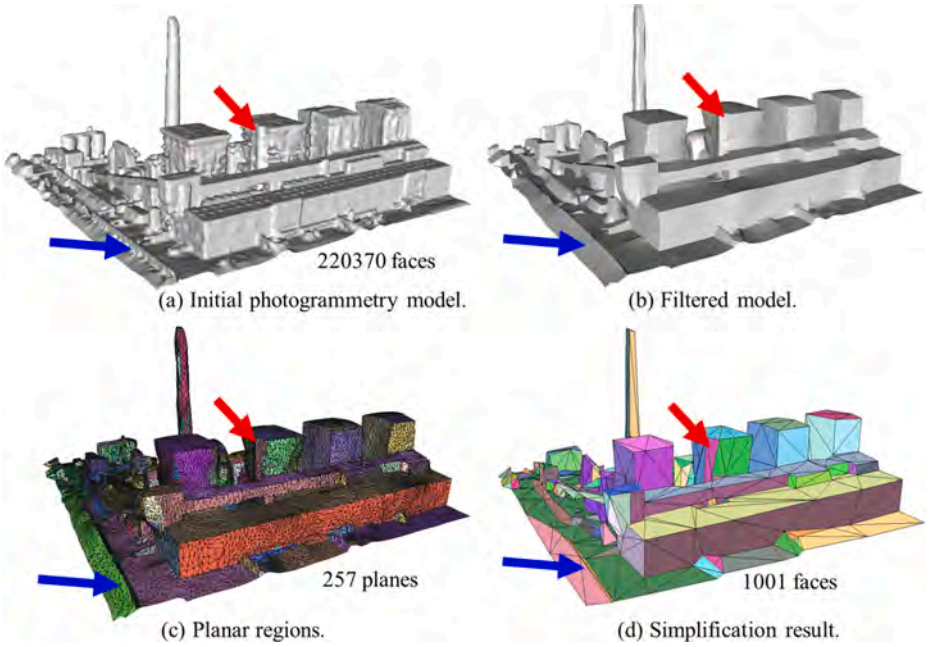


Fig. 7. A mesh simplification example on a power station model. The blue arrows indicate a channel that is degenerated during the procedure; the red arrows indicate an extra undesired planar structure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

variety of building mesh models with irregular connectivity of planar regions and features. These models are generated by different techniques, including photogrammetry techniques, surface modelling from LiDAR point clouds, and Kinect fusion algorithm. In our experiments, we set the normal angle tolerance to 15 degrees and set the result vertex number to about twice the number of the detected planar regions.

Fig. 6 demonstrates how the result evolves on an **office building** model generated by aerial photogrammetry techniques. The proposed method succeeds in (b) smoothing surfaces, (c) extracting planar regions and (d) region structures, and (e) detecting contour features. Note that not all faces have corresponding planar regions. As shown in (c), the faces in dark blue color indicate that they have not been grouped into any planar region. That is because the initial mesh models may contain some over smoothed or highly curved parts. Our segmentation method does not partition these curved regions, so the method simplifies unstructured parts such as trees on the ground and air conditioners at the roofs using traditional edge collapse operators. The structural parts, such as the main facades of the building, are simplified extremely with a much smaller number of faces. Through the comparison between the enlarged parts of (a) and (f), we can see that the triangle number of the model has been significantly reduced after the process.

Fig. 7 shows a similar example of a **power station**, in which (a) to (d) are the original model, smoothed model, segmentation result, and the final simplification result, respectively. We noticed that parts of the model with double thin slices shapes might be shrunk into a degenerated plane. As the blue arrows indicated, a channel on the ground has been converted to a step after the simplification. For large scenes with complex structures, some unexpected planes may be detected because the mesh surfaces at these parts present incorrect shapes caused by noises or occlusions. As the wall indicated by the red arrows, an additional planar region is extracted from the filtered model, which leads to an extra plane in the final model.

The experiments of both Figs. 6 and 7 demonstrate that the proposed method can generate lightweight polygonal mesh models, which have a more compact representation compared to the initial photogrammetry mesh models. Depending on the use, e.g., quantity survey or simulation, features such as baseboards, window ledges, roof tiles, or chimneys may be considered as useless pieces of information. Therefore, these details are usually omitted in the results. As a whole, even though real-world

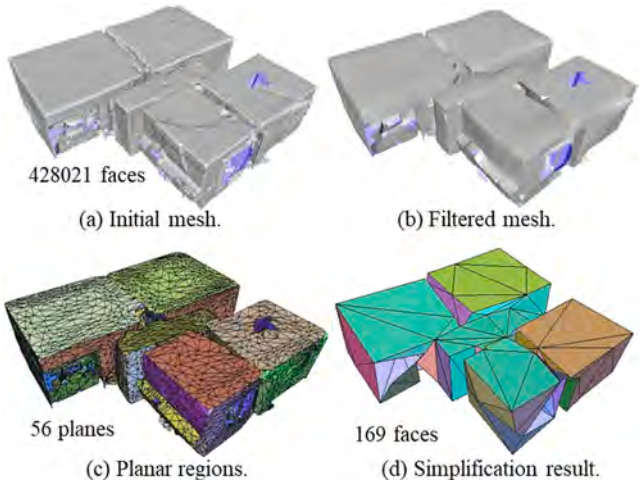
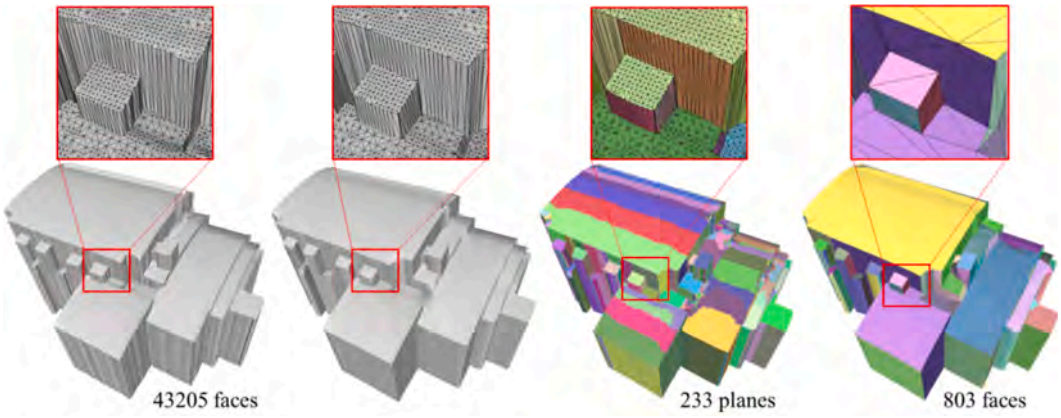


Fig. 8. Simplification of an indoor model generated by the Kinect fusion method from RGB-D data.

models generated from aerial photogrammetry techniques contain many bumbled surfaces, our filtering method can eliminate this type of defects on smooth regions and leads to extremely simplified meshes. The simplified models can well meet the standard of CityGML LOD2 models (Groger et al., 2012). The resulted models can be further mapped with textures coming from oblique photographic images based on the image orientation parameters.

We also verify the adaptability of this method by testing on two other types of data. One is the mesh model of an **indoor flat** generated by an RGBD sensor with the Kinect fusion method (Izadi et al., 2011); the other two are a **factory building** model generated from an aerial LiDAR point cloud by the dual contouring algorithm (Zhou et al., 2010). Intermediate and final results are depicted in Figs. 8 and 9. The characteristic of the indoor model is that there are a lot of holes due to transparent windows and occlusions in the scanning. We can see that there are some holes in the indoor mesh. These holes are retained after simplification. The characteristic of the LiDAR point clouds is that the



(a) Initial mesh. (b) Filtered mesh. (c) Planar regions. (d) Simplification result.

Fig. 9. The simplification of a **factory building** model reconstructed from a LiDAR point cloud.

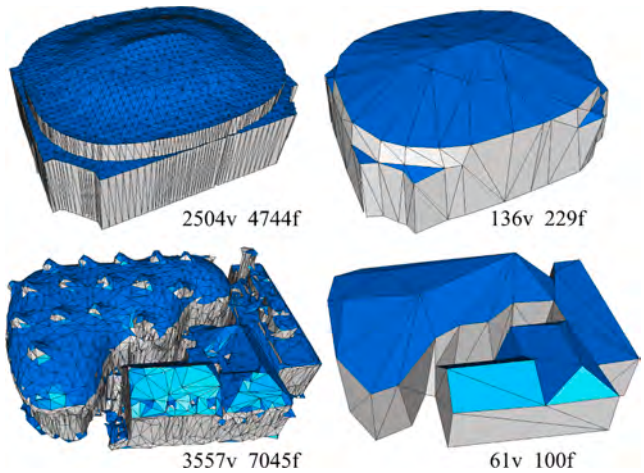


Fig. 10. The simplification of two models with curved domes.

façades of the initial model are sparse while the roofs are dense. That is caused by the top-down observation viewpoint of the aerial LiDAR sensor. Our method is capable of reducing the accumulation of local erroneous approximations during the edge collapse process. Regional structures of the model are preserved by not performing edge collapse operators that violate the minimization rules defined by Eq. (11).

Our method is based on the assumption that buildings are dominated by piecewise planar structures. For buildings that do not comply with this assumption, it has an effect of a piecewise planar approximation of the input model. Fig. 10 shows the models of a stadium and an exhibition hall both with smooth domes. The simplification results demonstrate that our method has good applicability. As there is a judgment in the algorithm, it can automatically fall into the traditional decimation method to process these smooth structures.

4.2. Quantitative analysis

All experiments were performed on a laptop with a 2.50 GHz Intel (R) Core (TM) i7-6500U CPU and 8 GB RAM. We have implemented our method in C++ with the VCG library (VCG Library, 2020) for mesh operation and libQGLViewer (libQGLViewer, 2019) for visualization. Statistics of the above-mentioned results are listed in Table 1. In our implementation, all models are stored by saving their vertices, vertex indices of faces, and face normals. As the compression ratio of the points and that of the faces are not the same after simplification, we compute the overall compression ratios based on the file sizes in ASCII format. To evaluate the accuracy of the simplification results, we calculate the fitting error based on the distance between the vertices of the original model and the faces of the simplified model. Specifically, the distances are calculated from the original model vertices to the closest faces on the simplified model. The mean value and the root mean square (RMS) value of the distances are used to evaluate the fitting accuracy. By using the diagonal length of the scene's bounding box as a reference, the relative error can be analyzed in the global scope. Since the models are generated with calibrated devices and photogrammetry techniques, their lengths have the real physical unit.

Form Table 1, we can also conclude that the compression ratio of the **factory building** model is larger than the others. That is because the main structure of this scene is a large curved dome, which could not be represented by a small number of planar structures. In this case, more face edges are processed by calculating the conventional error metric. As listed in the last column, most of the relative errors are less than 1% of the whole scenes except for the **indoor model**. The indoor scene contains a large number of small pieces of furniture that are eliminated in the result, increasing the errors.

Using the **power station** data, a series of simplified models with

Table 1
Statistics of sizes and errors for the simplification results.

Model	Planar region #	Original vertex / face #	Result vertex / face #	Compress ratio	Distance (meter)		
					Diagonal	Mean error	RMS error
Office building	112	14,643 / 28,929	208 / 391	1.12%	279.827	0.384 (0.137%)	0.601 (0.215%)
Power station	257	100,543 / 220,370	514 / 1001	0.64%	574.940	0.936 (0.163%)	1.663 (0.289%)
Indoor flat	56	223,634 / 428,021	101 / 169	0.03%	15.236	0.109 (0.717%)	0.185 (1.21%)
Factory building	233	22,118 / 43,205	466 / 803	3.32%	228.054	0.162 (0.071%)	0.315 (0.138%)

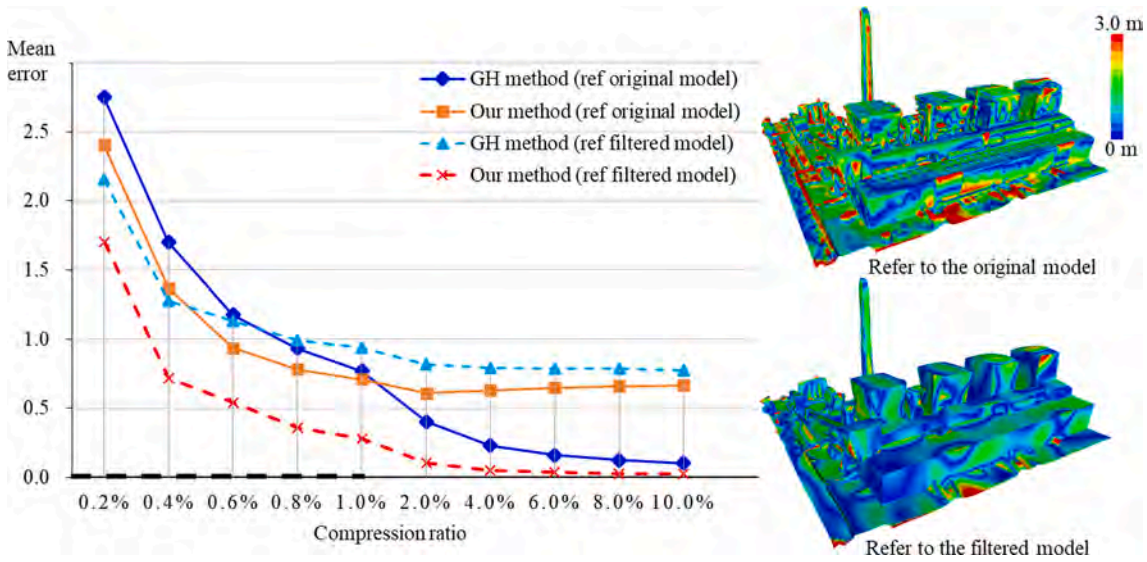


Fig. 11. Mean geometric errors. On the right, we show the errors of the simplified model in Fig. 7(d).

Table 2
Statistics on the running times of each step of the method on four examples.

Model	Timing (Sec.)			
	Mesh filtering	Region extraction	Edge collapse	Total
Office building	1.85	4.27	0.33	6.45
Power station	10.40	83.43	1.91	95.74
Indoor flat	25.25	260.22	3.35	288.82
Factory building	6.53	55.21	0.40	62.14

different compression ratios are analyzed on geometric fitting accuracy. As the geometric errors of traditional methods are close, GH's method is used as a reference to evaluate the accuracy of other methods. We evaluate the mean errors of all results by calculating the projection distances from the original model's vertices and from the filtered model's vertices to the result models, respectively. In addition to the original model, we choose the filtered model as a reference for the following reasons. We assume that the filtered model has already discarded many fine details. Besides, the photogrammetric error of the original model is not considered. As shown in the color texture of the original model, the areas with large errors are mainly uneven parts on the roofs, walls, and the ground. By referring to the filtered model, we can better understand the advantages of the simplified model in terms of structural reservation. Most regions remain consistent geometric positions in the filtered model after simplification. The mean errors are plotted in Fig. 11 (left). As the compression ratio increases, we can observe similar error trends of different methods. In Fig. 11 (right), we depict the original model and the filtered model with color attributes related to their vertex projected distance to the result in Fig. 7 (d).

In the tests referring to the original model, we observed that GH's method has better accuracy when the compression ratio value is larger than 1.0%. That is because it uses the local feature to design the error metric, and more fine details are maintained at low compression ratios. On the contrary, our structure-preserving strategy resulted in a higher error due to the planar region-based abstraction. When the compression ratio increases, the advantage of our method is shown. The results have a lower error at coarse levels by preserving the dominant structural elements. The method performs well in terms of improved accuracy and preservation of scene structures. Hence, our approach is more suitable for extreme simplification tasks.

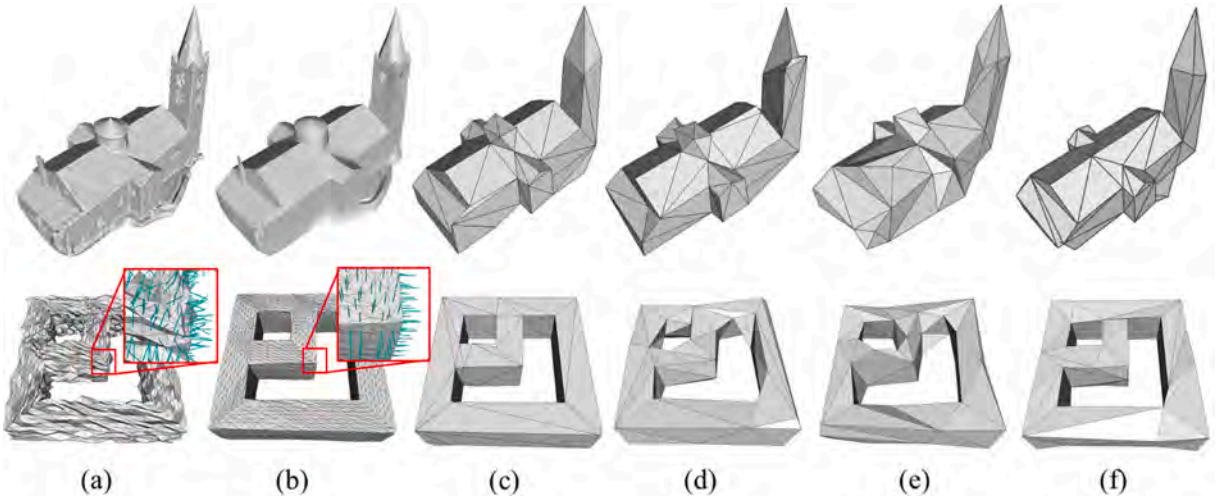


Fig. 12. Comparison on a church building (top row) and a noisy torus model (bottom row, noise magnitude is 30% of the average edge length of the mesh). (a) Initial mesh model, (b) our smoothed model, (c) our simplification result, (d) result generated by GH's method (Garland and Heckbert, 1997), (e) LT's method (Lindstrom and Turk, 1998), and (f) result of SLA's method (Salinas et al., 2015). All methods simplified the model to 50 vertices.

Table 3
Comparison with time consumption and geometry accuracy.

Model	Initial vertex#	Result vertex #	Timing (Sec.)	Mean error (m)	Method
Church	95033	50	19.011	0.17	Ours
			15.110	0.23	SLA
			2.018	0.33	LT
			2.062	0.29	GH
Torus	2686	50	1.640	0.52e-3	Ours
			1.362	0.003	SLA
			0.037	0.013	LT
			0.255	0.009	GH

In Table 2, we report the computation times of the three key steps. Firstly, the normal filtering and vertex update is linear to the number of mesh faces and vertices. So the complexity of the filtering process is $k\mathrm{O}(n^F + n^V)$, where k is the number of iterations, n^F and n^V are the number of mesh faces and vertices respectively. For the coarse region extraction process, the region growing complexity is $\mathrm{O}(n^F)$; the region refinement process consists of finding the region connectivity and calculating region area and fitting distances from faces to their neighbouring planes. The complexity is $\mathrm{O}(n^F m^2)$, where m is the number of regions detected by the coarse region growing method. Consequently, this is the most time-consuming step of the whole method. For the edge collapse operation, the operator iteratively queries the deleting edges in a heap, which has a complexity of the process is $\mathrm{O}(n^E \log n^E)$, where n^E is the total number of edges.

4.3. Comparison

We further compared our method to those of the GH's method (Garland and Heckbert, 1997), the LT's method (Lindstrom and Turk, 1998), and the SLA's method (Salinas et al., 2015), using a photogrammetric **church building** model and a **torus model** with additive Gaussian noises. In our experiment, the results of the GH's method and the SLA's method are obtained by the demo program of Salinas et al. (2015). LT's method and our method are implemented using the VCG library (VCG Library, 2020). Adopt the comparison method used in (Salinas et al., 2015), we set the expected vertex number of the final model to 50 in both cases. The comparison results are shown in Fig. 12.

The original models may contain different levels of noises and some defects near sharp features. After the mesh filtering phase, these defects are significantly reduced. This is particularly noticeable in the example of the torus model. Fig. 12(b) indicates that our results converge to piecewise smooth regions with clean sharp features. Based on the planar region constraint, the vertex positions tend to adapt themselves to the mesh curvature as it can be observed on the models (Fig. 12(c)).

A smaller vertex number indicates a coarse complexity of the model. Both GH's and LT's approaches have troubles to reach low approximation error while preserving structures (see Fig. 12(d) and (e)). The boundary areas of the input data are over smoothed or noisy, which leads to the shrinkage and degeneration during the edge collapse processes and progressively cause the accumulation of erroneous.

The SLA's approach also used the detected planar structures to design the edge collapse rule. However, it lacks the strategy on the non-planar structures, so its result (Fig. 12(f)) is sensitive to the quality of planar proxy detection. As the tower in the middle of the roof is not detected by any plane elements, it is discarded in the final result. When the input model is seriously contaminated by noise (the torus mode), SLA's method performs poorly at recovering clean planar features. This finally causes an unstable result. Compared to these methods, our method introduces a mesh filtering process before the edge collapse operations, which significantly improves the robustness of the pipeline and prevents error accumulation during the subsequent edge collapse operations. As our definition of error metric of vertices uses a hierarchical strategy, the method can handle planar regions as well as non-planar regions.

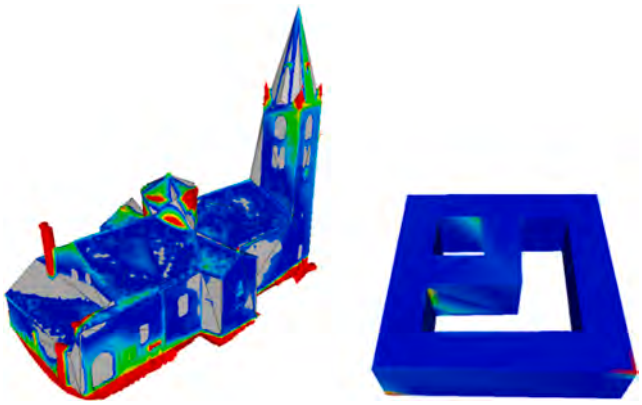


Fig. 13. Fitting errors of the Church and Torus models.

A comparison of time consumption and geometry accuracy is given in Table 3. The proposed method and the LSA method took longer for the preprocessing of plane extraction. Besides, the proposed method is slower in mesh smoothing. The majority of the computational costs of the algorithm is due to updating the normal vector and vertex coordinates. Although our method is inferior to other methods in running time, it has obvious advantages in terms of result accuracy. Fig. 13 shows the fitting error of our simplification models on the initial mesh (Church) and the ground truth model (Torus). Large errors mainly occur in some small corners and bumps, such as chimneys and horns. These objects do not affect the overall structures. From the Torus model, it can be seen that there is some offset in the corners, which is caused by the smoothing step. Smoothing improves the simplification in the flat areas but may introduce small errors at the corners.

The robustness to noise is one of the characteristics of our method. As the filtering process can suppress the noise level, the complex level of the simplified model is less affected by noise. The noise tolerance of our method is illustrated in Fig. 14, in which the models are manually designed and corrupted with different levels of noise. We can observe that the quality of filtering results and region extraction are stable to the noise levels. As shown in the second row, large noise blurs the sharp edges, while the filtered result would smooth the roof eaves. With the increase in noise, the number of small planes detected also increases. The region refinement method merges these small planes into large regions, reducing the number of structures. When the noise magnitude is less than 20% of the average edge length, the algorithm can faithfully recover the model to the same quality as without noise. Therefore, our algorithm can be applied to a wide range of models.

4.4. City model

Fig. 15 shows the simplification of a city model with various building structures. The area is 890 m long and 350 m wide. The result shows that our method can generate well-decimated meshes in terms of both visual coherence (Fig. 15(b)) and geometric accuracy (Fig. 16). As shown in the

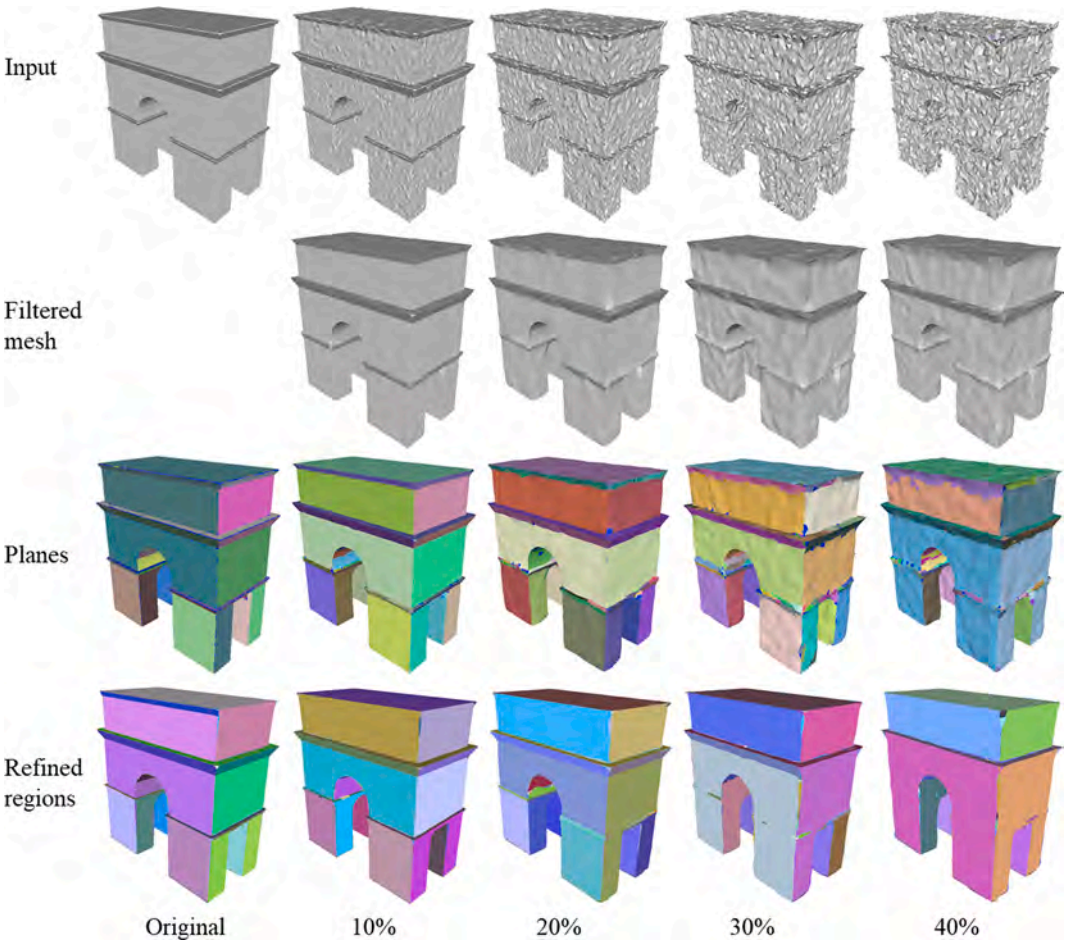


Fig. 14. The robustness to noise. From left to right, the meshes in the columns are corrupted with noises, whose magnitude is 0%, 10%, 20%, 30%, and 40% of the average edge length respectively.

enlarged windows in Fig. 16, the protrusions such as chimneys are the main source of errors.

In Fig. 17, we present a comparison using a single building from the city model, whose face number is reduced from 1667 to 60. This experiment revealed that plane feature-preserving approaches, such as the SLA and our method, are sensitive to the quality of plane detection. As our method benefits from the smoothing process, the plane detection results are cleaner and more accurate. Plane features perform well at recovering clean sharp features of small roof patches. Because the GH method has no feature-based constraints, it will flatten some adjacent non-coplanar regions. Therefore, compared with SLA and ours, the GH result has a smaller number of planar areas. As indicated by the arrows in Fig. 17, after applying the GH and SLA methods, some small roof structures were deformed, and these parts have larger errors. Overall, the proposed method has better accuracy.

4.5. Limitations

An ideal simplified mesh should be visually coherent to the initial model and contain only the minimal complexities required to represent the model without losing structures. However, how to define minimal complexities is still very ambiguous. It is difficult to find a universally applicable criterion to terminate the simplification method. In Fig. 18, we show a set of results to explain the visual effect of the target number of vertices. Since the models in Fig. 1 and Fig. 8 are relatively simple and can be evaluated intuitively, we use these two groups of results for analysis. As can be seen, when the vertex numbers of the models are greater than twice the number of planes we set, the overall structures of the models change little. When the number of target vertices is further reduced, the structure of the model may be completely lost at a certain time indicating a failure in simplification. We notice that increasing or decreasing one or two vertices has little effect on the model when the total number is large. However, when the vertex number is reduced to a low number, individual vertex adjustment will cause unexpected results in the simplified model. We set the number of vertices twice the number of extracted planes. Although this is not necessarily the optimal number of vertices, it provides a good compromise.

Our target objects are mainly piecewise planar buildings since these structures are dominated by planar shapes. For buildings that consist of large free-form surfaces, the computational complexity of our method will increase and the algorithm may lose its advantages. Besides, the proposed algorithm cannot guarantee all refined vertices perfectly aligned with the abstracted planes, which is a common issue of quadratic optimization-based mesh simplification methods.

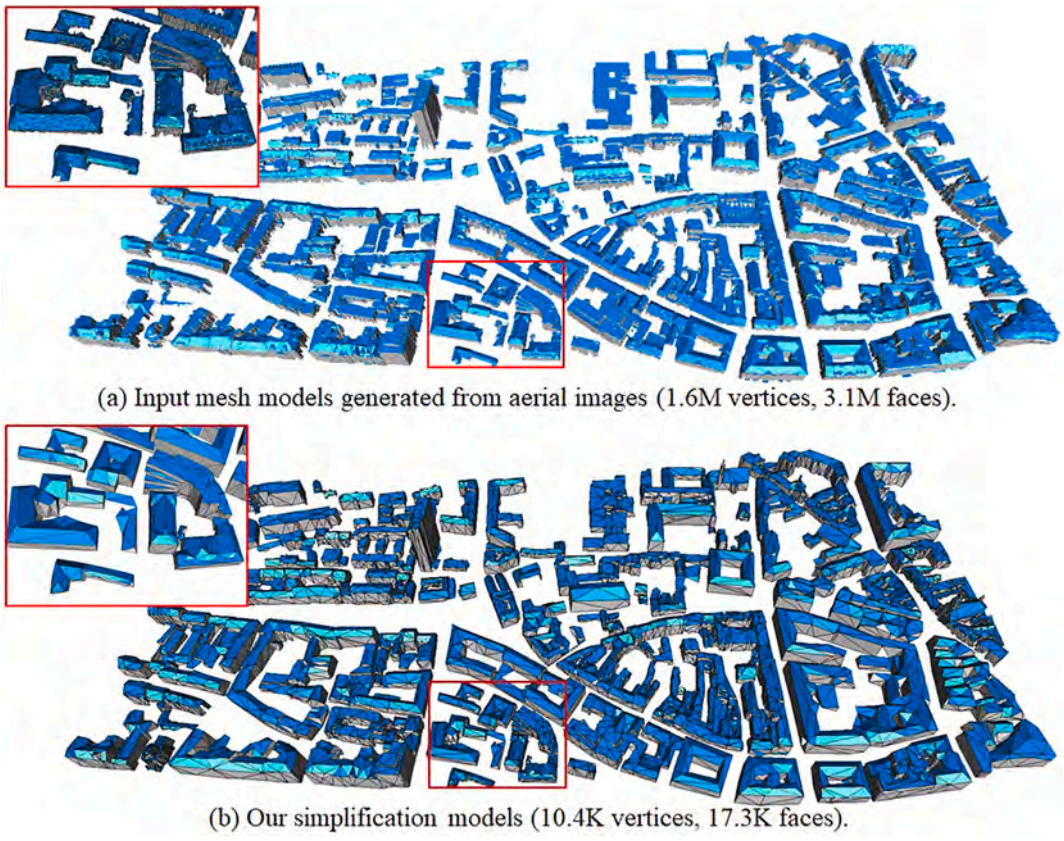


Fig. 15. City models of Graz. We separate the models into roofs and walls with different colors for better visualization.

Fig. 16. Fitting error. The color map indicates the distance from the initial meshes to the simplified meshes. (Max: 4.998 m; mean: 0.445 m; RMS: 0.750279). Two enlarged single models (bottom) show the overlay of error maps on the simplified models. The parts with large errors are mainly at chimneys and dormers.

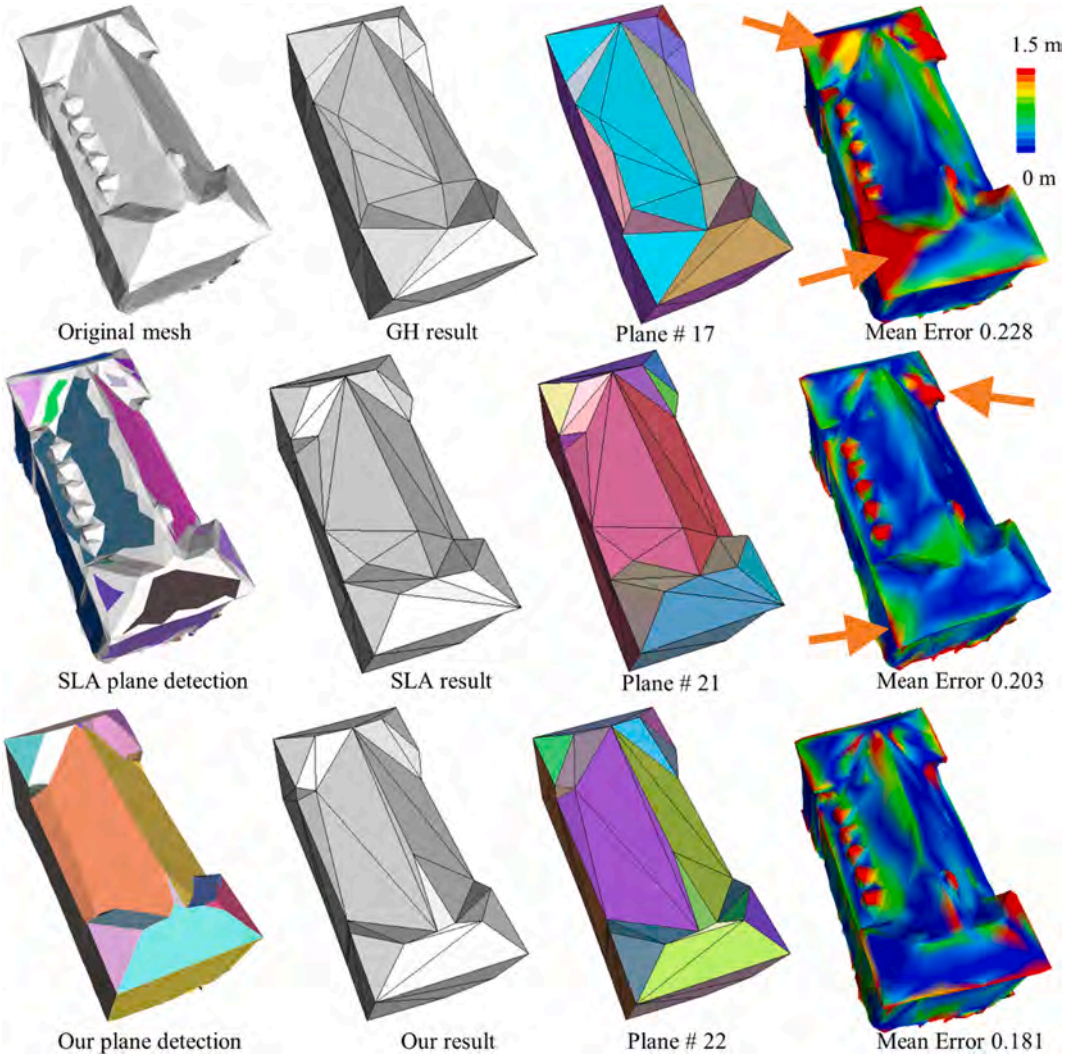


Fig. 17. Comparison on a single building from the city model.

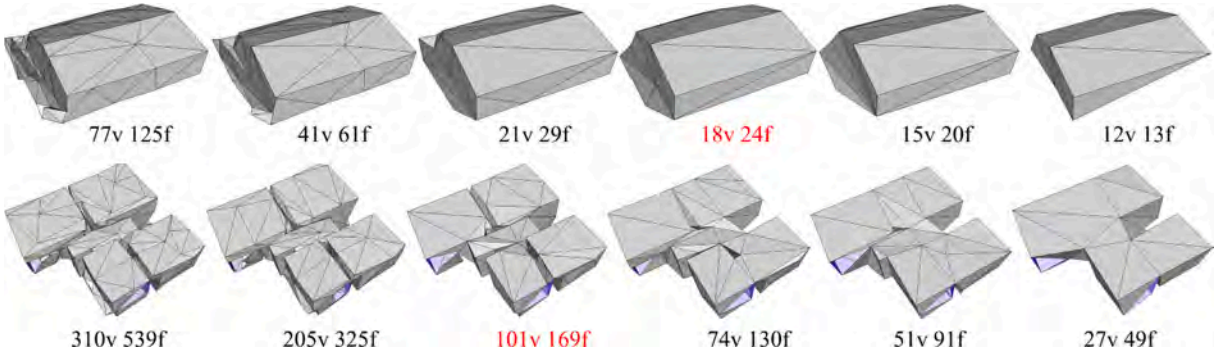


Fig. 18. The simplification results by setting different vertex numbers. The red numbers correspond to the proposed method, which uses about twice the number of extracted planes. (v: vertex number; f: face number. 9 planes detected in the first model; 56 planes detected in the second model.) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Conclusion

This paper presented an approach for producing compact building models from dense triangle meshes. The method first applies a filtering process on the face normals, followed by updating the vertex positions. The filtering process can denoise meshes on planar regions while enhancing sharp edge features. We have revealed that the mesh filtering process can significantly improve the performance of the subsequent simplification algorithm, which was simply ignored by most previous approaches. After the mesh filtering and planar region extraction, an improved edge collapse method is proposed to reduce the number of mesh faces. Based on a hierarchical definition strategy of error metric, the method retains the building structures in a good manner for both planar and non-planar regions. As shown in the experimental results, the

proposed approach has strong generality and can be applied to many different data types. The simplified models can be widely used in a range of graphics and mapping applications. As future work, we plan to parse the simplified building models into some semantic components, allowing users to efficiently edit and analyze the models.

Declaration of Competing Interest

The author declare that there is no conflict of interest.

Acknowledgements

We thank Vexcel Imaging for providing us the mesh models of the city Graz for experiments. This work was supported in part by the National Natural Science Foundation of China under Grant: 41801342, and in part by the Natural Science Foundation of Jiangsu Province, China, under Grant: BK20170781.

References

Agisoft Metashape, Agisoft LLC, http://www.agisoft.com, February 2020.
Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T., 2003. Computing and rendering point set surfaces. IEEE Trans. Visual Comput. Graphics 9 (1), 3–15.
Bormann, R., Hampp, J., Hägele, M., Vincze, M., 2015. Fast and accurate normal estimation by efficient 3d edge detection. In: In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3930–3937.
Botsch, M., Kobbelt, L., 2001. Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. Comput. Graphics Forum 20 (3), 402–410.
Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R., 2001. Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, p.67-76, August 2001.
Chauve, A.L., Labatut, P., Pons, J.P., 2010. Robust piecewise-planar 3D reconstruction and completion from large scale unstructured point data. In: In: IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp. 1261–1268.
Chen, D., Wang, R., Peethambaran, J., 2017. Topologically aware building rooftop reconstruction from airborne laser scanning point clouds. IEEE Trans. Geosci. Remote Sens. 55, 7032–7052.
Chen, T., Wang Q., 2011. 3d line segment detection for unorganized point clouds from multi-view stereo. Computer Vision–ACCV 2010, Springer (2011), pp. 400-411.
Choi, C., Trevor, A. J., and Christensen, H.I., 2013. RGB-D edge detection and edge-based registration. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.
Cohen, J., Manocha, D., Olano, M., 2003. Successive mappings: an approach to polygonal mesh simplification with guaranteed error bounds. Int. J. Computat. Geometry Appl. 13 (1), 61–96.
Furukawa, Y., Ponce, J., 2010. Accurate, dense, and robust multiview stereopsis. IEEE Trans Pattern Anal Mach Intell 32 (8), 1362–1376.
Garland, M., Heckbert, P.S., 1997. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97, pages 209-216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
Groger, G., Kolbet, H., Nagel, C., et al., 2012. OGC City Geography Markup Language (CityGML) encoding standard, version 2.0.0. [S.l]: Open Geospatial Consortium, 2012.
Guennebaud, G., Gross, M., 2007. Algebraic point set surfaces. ACM Trans. Graphics (TOG) 26 (3), 23–42.
Hackel, T., Wegner, J.D., Schindler, K., 2016. Contour detection in unstructured 3D point clouds. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016, 1610–1618.
He, L., Schaefer, S., Mesh denoising via L0 minimization. ACM Trans. Graph. 32, 4, 64:1-8, 2013.
Henn, A., Gröger, G., Stroh, V., Plümer, L., 2013. Model driven reconstruction of roofs from sparse LiDAR point clouds. ISPRS J. Photogramm. Remote Sens. 76, 17–29.
Hofer, M., Maurer, M., Bischof, H., 2015. Line3d: Efficient 3d scene abstraction for the built environment. In: Gall J., Gehler P., Leibe B. (eds) Pattern Recognition. DAGM 2015. Lecture Notes in Computer Science, vol. 9358. Springer, Cham.
Huang, H., Ascher, U., 2008. Surface mesh smoothing, regularization, and feature detection. SIAM J. Scientific Computing 31 (1), 74–93.
Huang, H., Li, D., Zhang, H., Ascher, U., Cohen-Or, D., 2009. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. Graphics 28 (5), 1–7.
Izadi, S., Kim, D., Hilliges, O., et al., 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011, 559-568.
Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, 2006, pp. 61–70.
Kelly, T., Femiani, J., Wonka, P., and Mitra, N., 2017. BigSUR: Large-scale structured urban reconstruction. ACM Transactions on Graphics (Proceedings of ACM Siggraph Asia), 2017.
Lafarge, F., Mallet, C., 2012. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. Int. J. Comput. Vision 99 (1), 69–85.
Landrieu, L., Simonovsky, M., 2018. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake, USA, pp. 4558–4567.
Li, M., Nan, L., Smith, N., Wonka, P., 2016. Reconstructing building mass models from UAV images. Comput. Graph. 54 (C), 84–93.
Li, M., Rottensteiner, F., Heipke, C., 2019. Modelling of buildings from aerial LiDAR point clouds using TINs and label maps. ISPRS J. Photogramm. Remote Sens. 154, 127–138.
LibQGLViewer, libQGLViewer, http://libqglviewer.com/, 2019.
Lin, Y., Wang, C., Cheng, J., Chen, B., Jia, C., Chen, Z., Li, J., 2015. Line segment extraction for large scale unorganized point clouds. ISPRS J. Photogramm. Remote Sens. 102, 172–183.
Lindstrom, P., Turk, G., 1998. Fast and memory efficient polygonal simplification. IEEE Proceedings of the Conference on Visualization, Los Alamitos, CA, USA 1998, 279–286.
Lipman, Y., Cohen-or, D., Levin, D., Tal-ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graphics 26 (3), 22.
Marinov, M., Kobbelt, L., 2005. Automatic generation of structure preserving multiresolution models. Comput. Graph. Forum 24 (3), 479–486.
Mitra, N., Wand, M, Zhang, H., Cohen-Or, D., Kim, V, Huang, Q.X., 2013. Structure-aware shape processing. In: ACM SIGGRAPH Asia 2013 Courses, New York, NY, USA, 2013, 1:1–20.
Nan, L., Wonka, P., 2017. PolyFit: Polygonal surface reconstruction from point clouds. In: IEEE International Conference on Computer Vision (ICCV). Venice, Italy, Oct. 2017.
Niemeyer, J., Rottensteiner, F., Soergel, U., Heipke, C., 2016. Hierarchical higher order CRF for the classification of airborne LiDAR point clouds in urban areas. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 41, 655–662.
Ok, A.O., Wegner, J.D., Heipke, C., Rottensteiner, F., Soergel, U., Toprak, V., 2012. Matching of straight line segments from aerial stereo images of urban areas. ISPRS J. Photogramm. Remote Sens. 74, 133–152.
Oude Elberink, S., Vosselman, G., 2009. Building reconstruction by target based graph matching on incomplete laser data: analysis and limitations. Sensors 9 (8), 6101–6118.
Overby, J., Bodum, L., Kjems, E., Iisoe, P., 2004. Automatic 3D building reconstruction from airborne laser scanning and cadastral data using Hough transform. Int. Arch. Photogram. Remote Sens. 35 (B3), 296–301.
Pix4D, 2020, https://cloud.pix4d.com, 2020.
Rottensteiner, F., Trinder, J., Clode, S., Kubik, K., 2005. Automated delineation of roof planes from LiDAR data. Int. Arch. Photogram. Remote Sens. Spat. Inf. Sci. 36 (3/W19), 221–226.
Salinas, D., Lafarge, F., Alliez, P., 2015. Structure-aware mesh decimation. Comput. Graphics Forum 34 (6), 211–227.
Schnabel, R., Wahl, R., Klein, R., 2007. Efficient RANSAC for point-cloud shape detection. Comput. Graphics Forum 26 (2), 214–226.
Shen, C., O'Brien, J.F., Shewchuk, J.R., 2004. Interpolating and approximating implicit surfaces from polygon soup. ACM Trans. Graph. 23 (3), 896–904.
Solomon, J., Crane, K., Butscher, A., Wojtan, C., 2014. A general framework for bilateral and mean shift filtering. arXiv preprint arXiv:1405.4734 (2014).
Tomasi, C., Manduchi, R., 1998. Bilateral filtering for gray and color images. In: In: Proceedings of the Sixth International Conference on Computer Vision (ICCV), p. 839.
VCG Library: The VCG Library. http://vcg.isti.cnr.it/vcglib/, Jun. 2020.
Vosselman, G., Gorte, B.G.H., Sithole, G., Rabbani, T., 2004. Recognizing structure in laser scanning point clouds. In: Proceedings of the ISPRS working group VIII/2: laser scanning for forest and landscape assessment, Freiburg, October 2004, 33–38.
Wang, P., Shen, X., Russell, B., Cohen, S., Price, B., Yuille, A., 2016. SURGE: Surface regularized geometry estimation from a single image. In: 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain. 2016.
Wei, M., Yu, J., Pang, W.M., Wang, J., Qin, J., Liu, L., Heng, P.A., 2015. Bi-normal filtering for mesh denoising. IEEE Trans. Vis. Comput. Graphics. 21 (1), 43–55.
Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015a. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286–304.
Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F., Jutzi, B., 2015b. Contextual classification of point cloud data by exploiting individual 3D

neighborhoods. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Inf. Sci. II-3/W4, 271–278.
Wu, C., Agarwal, S., Curless, B., Seitz, S.M., 2011. Multicore bundle adjustment. In: Proceedings of the 2011 IEEE conference on computer vision and pattern recognition (CVPR), Washington, DC, USA, 11, 3057–64.
Xie, J., Feng, C.C., 2016. An integrated simplification approach for 3D buildings with sloped and flat roofs. ISPRS Int. J. Geo-Inf. 5, 128.
Xiong, B., Jancosek, M., Oude Elberink, S., Vosselman, G., 2015. Flexible building primitives for 3D building modeling. ISPRS J. Photogramm. Remote Sens. 101, 275–290.
Xiong, B., Oude Elberink, S., Vosselman, G., 2014. A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. ISPRS J. Photogramm. Remote Sens. 93, 227–242.
Yang, B., Fang, L., Li, J., 2013. Semi-automated extraction and delineation of 3D roads of street scene from mobile laser scanning point clouds. ISPRS J. Photogramm. Remote Sens. 79, 80–93.
Zelinka, S., Garland, M., 2002. Permission grids: practical, error-bounded simplification. ACM Trans. Graphics 21 (2), 207–229.
Zhou, Q.Y., Neumann, U., 2010. 2.5D dual contouring: a robust approach to creating building models from aerial LiDAR point clouds. In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. Lecture Notes in Computer Science, vol. 6313. Springer, Berlin, Heidelberg.