

S-LPM: segmentation augmented light-weighting and progressive meshing for the interactive visualization of large man-made Web3D models

Wen Zhou¹  · Kai Tang² · Jinyuan Jia¹

Received: 14 December 2017 / Revised: 13 April 2018 / Accepted: 4 June 2018 /
Published online: 17 July 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract With the advent of the era of “big data”, increasing efforts have been focused on how to process large models to improve transmission over the internet and display in a browser, i.e., Web3D technology. Notwithstanding the many new advancements in Web3D technology, because browsers have limited storage capacity and low computational ability, the efficient display of a large model through the net remains a bottleneck problem. This paper proposes a light-weighting visualization framework, called the S-LPM framework, which includes a novel Dijkstra-based mesh segmentation operation and a new voxel-based repetition detection/removal operation to efficiently display large 3D models in a Web browser. The two key geometric operations substantially reduce the amount of data transmitted over the net, which in turn significantly increases the transmission speed. The partially transmitted data are then aligned through transformations to restore the entire original model and display it in the Web browser. The experimental results show that our approach is generally accurate and feasible, and its performance is superior to that of the benchmarking methods.

Keywords Big data · Web3D · Light-weighting · Mesh segmentation · Repetition · Fine-grained scene graph

✉ Jinyuan Jia
jyjia@tongji.edu.cn

Wen Zhou
zhouwen327@163.com

Kai Tang
mektang@ust.hk

¹ School of Software Engineering, Tongji University, Shanghai 201804, China

² Hong Kong University of Science and Technology, Hong Kong 999077, China

1 Introduction

With the coming internet plus era, VR technology has rapidly been evolving, with increasing emphases/demands on Web browsers. The efficient processing of 3D models is a key to Web browsers. Nonetheless, the conflict between the limited load capacity and the growing data demands is becoming increasingly critical. Effective transmission and rendering of 3D data over Web browsers has never been more critical, especially in Web3D visualization. Level of detail (LoD) technology was believed to be one of the best solutions for Web3D visualization; however, it remains a synchronous processing method of loading and rendering. In addition, LoD technology can process only a single model at a time, rather than a scene of many different models. The new WebGL/HTML5/three.js technology makes it easier to visualize a 3D model over Web browsers. However, this technology still processes only a single mesh at a time and lacks support for progressive meshing. Motivated by the above limitations, in this paper, we propose an S-LPM framework to process 3D models for Web viewing. Our framework supports multi-threads, which significantly accelerates Web viewing because multi-thread asynchronous loading and increment rendering can alleviate the burden placed on browsers. Additionally, the novel framework offers an efficient matrix formulation for mesh segmentation through which the majority of repeated components in the 3D model can be eliminated.

The proposed framework makes several contributions. First, we propose a new method of mesh segmentation for the detection/removal of repeated components. This method consists of mesh segmentation, progressive meshing, repetition detection and transformation of repeated components. The approach covers both data transmission and rendering, spanning the entire lifecycle of Web3D visualization. Second, our mesh segmentation can achieve asynchronous multi-thread transmission. Third, via progressive meshing, we are able to realize incremental loading and rendering. Finally, a Dijkstra-based algorithm is proposed for mesh segmentation to significantly improve both the speed and the results of mesh segmentation.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 introduces our proposed framework. Section 4 provides details about the specific methods and algorithms constituting the framework. Section 5 presents our experiments to validate the effectiveness and soundness of our proposed framework. Section 6 concludes the paper.

2 Related work

A number of solutions, such as stream mesh, mesh simplification, mesh reconstruction, parametric surface and implicit surface models, have been proposed to improve the user experience of large geometric models over a Web browser. Martin et al. [12] proposed a scheme that incrementally encodes a mesh in the order it is given to the compressor using only minimal memory resources. Compression begins after receiving the first few triangles. However, decompression demands substantial memory and computing resources, but Web browsers have limited computing capacity and storage space. Simplification technology can reduce the complexity of an originally very large mesh and, to some extent, can improve the rendering and loading performance of 3D models over a Web browser. However, simplification reduces the accuracy of the model, which is often an undesirable side effect in many Web3D applications. The reconstruction approach eases the burden on memory but requires a

large additional amount of computing time to reconstruct the mesh, and it often leads to poor user experience. Parametric surface representation, while often achieving good results for models with smooth shapes, is generally not suitable for models with sharp corners and planar faces, such as building information models.

As a powerful means of geometric analysis, mesh segmentation has been applied in many areas, such as component-based shape synthesis (Kalogerakis et al. [13]), which uses segmentation and labelling to produce new models by transferring corresponding segments from one model to another; 3D scene analysis; part-based recognition, 3D video compression; and 3D object retrieval (cf. Theologou et al. [30], Savelonas et al. [25]). Golovinskiy et al. [2] introduced a graph-clustering method to balance intra-mesh and inter-mesh segmentation. They built the connection by matching points between rigidly aligned meshes. However, they only handled a limited number of model types due to the requirement of global rigid alignment. Xu et al. [33] classified meshes according to their styles and then established the part correspondences in each style group. However, the group generation process was computationally expensive. Kreavoy et al. [17] created a consistent segmentation by matching parts generated from an initial segmentation. Huang et al. [10] considered the segmentation of individual meshes via linear programming. However, the segmentations generated by these methods cannot guarantee consistency across the whole set of meshes, even if they are mutually consistent. Sidi et al. [29] analysed the descriptor space via spectral clustering to segment a set of shapes with large variability. Meng et al. [22] clustered primitive patches to generate an initial guess and improve the co-segmentation results via multi-label optimization. Hu et al. [9] generated segmentations by grouping the primitive patches of the meshes directly and simultaneously obtained their correspondences. Their method achieves certain success, benefiting from the observation that patches belonging to the same part are likely to be in one common subspace in the feature space. Liu et al. [21] introduced the low-rank representation into semantic mesh segmentation and labelling; however, their method has several limitations, such as model style.

In addition, a rich body of mesh segmentation methods in the computer graphics literature aim at decomposing a mesh into functional parts. Shamir [26] and Agathos et al. [1] published a survey of these methods, and Attene et al. [3] compared several representative mesh segmentation algorithms. The methods in [1, 3, 26] aimed to create segments that are well-formed according to some pre-defined low-level criteria; moreover, the segments are convex, e.g., the boundaries lie along concavities. Well-known segmentation techniques include K-means (Shlafman et al. [28]), graph cuts (Katz et al. [14]), hierarchical clustering (Garland et al. [7], Inoue et al. [11]), random walks (Lai et al. [18]), core extraction (Katz et al. [15]), tubular primitive extraction (Mortara et al. [23]), spectral clustering (LIU et al. [20]), and critical point analysis (Lin et al. [19]). Theologou et al. [31] published a comprehensive survey of 3D mesh segmentation, including the current trends in 3D mesh segmentation.

On the other hand, many 3D models are man-made and comprise a wide range of components that tend to follow some obvious segmentation according to the connectivity. Similar components with different rigid body geometry transformations usually exist in these 3D models. Shikhare et al. [27] and Cai et al. [4] proposed a repetition detection approach to find similar components in 3D models; however, their matching methods are not accurate due to the limitations of their strategies. Wen et al. [32] presented a similarity-aware 3D model reduction method called lightweight progressive meshes, which attempts to search similar components and reuse them through the construction of a lightweight scene graph. However, their method depends on manual segmentation. Zhang et al. [34] proposed a novel cross-media

distance metric learning framework based on sparse feature selection and multi-view matching. They constructed a multi-modal semantic graph to find the embedded manifold for cross-media correlation. However, the scale of their test dataset is too small for genuine validation. Saleem et al. [24] proposed a scalable linked data-driven solution for the integration, query and visualization of bio-medical data; however, again, the scale of the data source is too small.

3 Proposed framework

Figure 1 graphically illustrates the flow of our proposed framework, which consists two main parts – the online pipeline and offline pipeline. In the offline pipeline, a suite of operations – mesh segmentation, mesh split, component repetition detection, and transformation – are performed on the original complete model. Pose alignment is adopted for the later task of correctly reassembling the segmented components into the original 3D shape, and a fine-grained scene graph (FG-SG) file is obtained at the end of the pipeline. In the online pipeline stage, the FG-SG file is first parsed; then, multi-threads are generated to concurrently and asynchronously load the mesh data. Furthermore, the outcome of each sub-thread is sent to the main thread to perform component-wise rendering. In addition, progressive rendering is performed for large components. Finally, the repeated components are represented by a matrix transformation. These processing steps enable a large mesh model to be completely visualized over a Web browser.

Pose alignment processing, which we adopt and implement in our system, was originally proposed by Wen et al. [32]. The main objective of pose alignment is to represent the components of a model in a canonical coordinate system so that the disassembled components can be easily reassembled to form the original 3D model. Pose alignment processing includes translation-invariant, rotation-invariant, and scaling-invariant transformations. Every component can individually perform these transformations. During the process, the transform

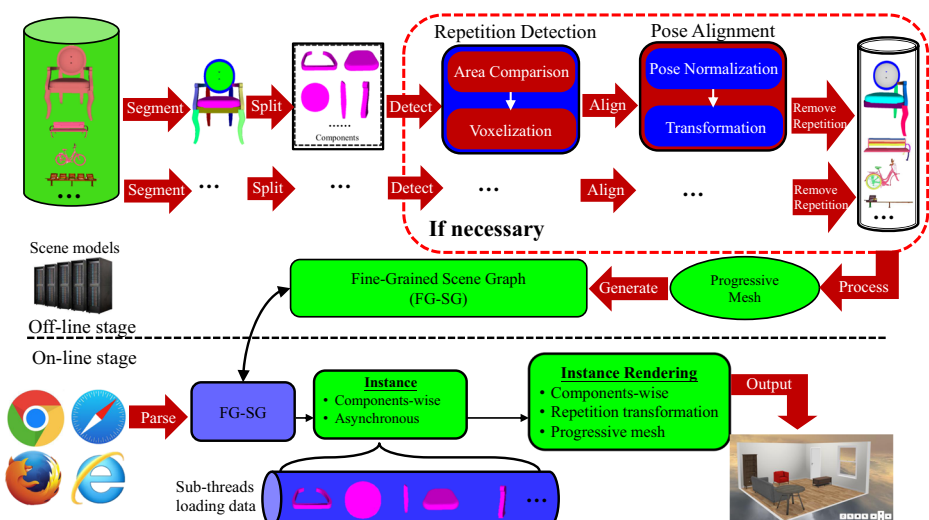


Figure 1 Overview of the proposed framework

quaternion associated with each component of the model has to be recorded for later restoration. When the process is optimized, in particular, by using PCA to add a symmetry-invariant transformation after the rotation-invariant transformation, the alignments of similar components become overlapped.

In the next section, we provide more details and describe the key technologies of our proposed framework.

4 Details of the S-LPM framework

4.1 Dijkstra-based mesh segmentation

Mesh segmentation is a powerful tool in many geometric processing areas, such as mesh compression, surface parameterization, and shape feature identification. A new procedure based on the Dijkstra algorithm is proposed in this paper. The Dijkstra algorithm is used to find the shortest path between nodes in a graph. First, we take any triangle in the mesh as a node. Then, the mesh model is converted into a network. This network can be separated into several regions based on triangle diffusion. Finally, the mesh is split into many different regions to achieve mesh segmentation. The entire process of the Dijkstra-based mesh segmentation consists of two major operations, i.e., determination of source set, regional diffusion (see Figure 2).

4.1.1 Description of the algorithm

1. Determination of the set of diffusion sources \mathcal{T} .

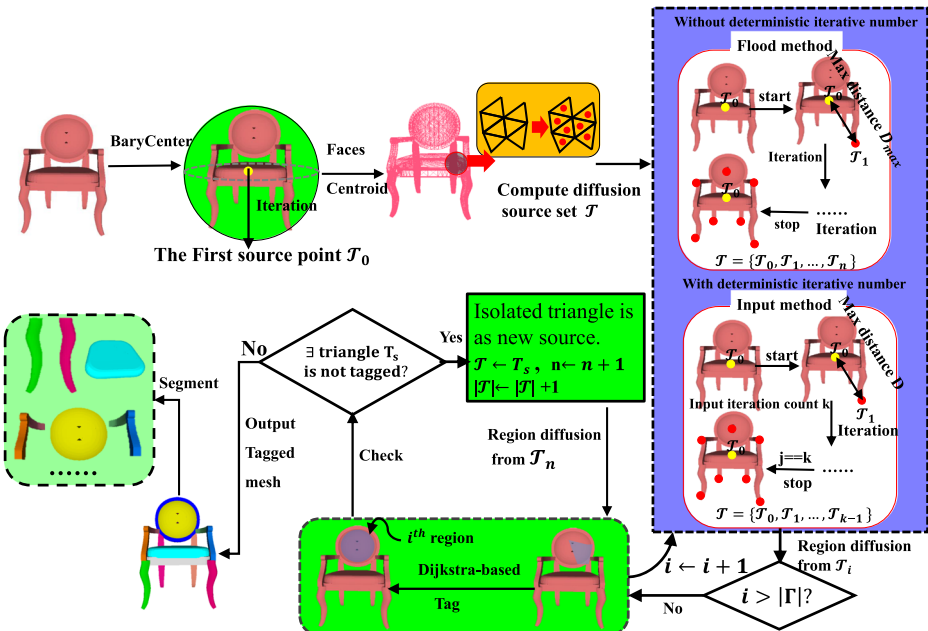


Figure 2 Overview of the proposed mesh segmentation algorithm

We K calculate the position of barycentre \vec{F} of the whole model S and then acquire the nearest triangle T_0 from barycentre \vec{F} as the first diffusion source position. In addition, the centroids of all the triangles are calculated, i.e., $\mathcal{K} = \{k_1, k_2, \dots, k_m\}$, where the term m is the number of triangles in S . The distances between the centroids in \mathcal{K} and the centroid of triangle T_0 are computed. Two different methods can be used to obtain the diffusion sources. The first method, i.e., flood method, is used when the user does not provide an input and can be viewed as the number of segments. The flood method requires iterative computation to obtain the number of segments. That is to say, the number of diffusion sources can be viewed as the number of segments. Let D_{max} denote the largest of these distances. We select the triangle of D_{max} as the next diffusion source position, i.e., T_1 , and update D_{max} accordingly. Then, we continue to iteratively obtain T_{i+1} until a T_n is found such that the distance between every triangle in \mathcal{K} and every source in the source set $\mathcal{T} = \{T_0, T_1, T_2, \dots, T_n\}$ is less than D_{max} . The second method, namely, the input method is used when the user provides an input t as the number of segments. Clearly, the actual number of segments is less than the input. The parameter t also denotes the iteration count. When the iteration count is equal to t , the iteration process stops automatically, and the final source set \mathcal{T} is obtained. where the term $|\mathcal{T}|$ represents the size of the source set, then $t > |\mathcal{T}|$. In contrast to the flood method, for a source T_i , the next source T_{i+1} is selected to maximize the distance between T_i and T_{i+1} . In this case, the source set \mathcal{T} often contains the elements of T_{i+1} . Hence, the actual number of segments is often less than the number of inputs t .

2. Diffusion from every source in \mathcal{T} .

Starting from T_0 , we can obtain all its neighbour triangles $\mathcal{K}_0 = N(T_0)$, where, the term N denotes the neighbour table of whole mesh, whose function is to record neighbour information of every triangle. Let $\mathcal{K}_0 = \{p \in [0, m] | k_p^0\}$, $\forall k_p^0 \in \mathcal{K}_0$, we can then calculate the global distance between T_0 and k_p^0 . The topological network can then be built. In addition, we can obtain the shortest path Ψ_0 from the source point T_0 to its neighbour triangles \mathcal{K}_0 via the Dijkstra algorithm. If triangle k_j ($j \in [0, m]$) belongs to path Ψ_0 , it is included in the current region whose centre is T_0 . We then tag k_j as 0. Next, the position of region centre T_0 is recalculated. In this circumstance, new neighbour triangles are added to set \mathcal{K}_0 . Let diffusion table R denote the tagged information of every source point. Hence, the term $R[T_0]$ represents all triangles tagged as 0. In essence, the term $R[T_0]$ always is updated via diffusion. This process is called *regional diffusion*. Subsequently, the above steps are repeated until there are no triangles in the neighbour set $N(T_0)$ that can be used for diffusion. That is, $\forall T_i \in \mathcal{T}$ and $T_i \neq T_0, \forall k_i \in \mathcal{K}$ and $k_i \in \mathcal{K}_0, \mathbb{1}(k_i) d_{global}(k_i, T_i) > d_{global}(k_i, T_0)$. Likewise, this regional diffusion process is applied to T_i ($0 < i \leq n$). Note that if triangle k_j belongs to the region whose centre is T_i , it is tagged as i .

After all the source points in \mathcal{T} have completed their regional diffusion, any remaining untagged triangles are also diffused. In the end, we obtain the cluster index of each triangle, which is then output as a file.

4.1.2 Metrics of the distance

Because the Dijkstra algorithm depends on the distance metric, we consider two types of distance – the centroid distance $d_{centroid}$ and the angular distance $d_{angular}$. Figure 3 illustrates

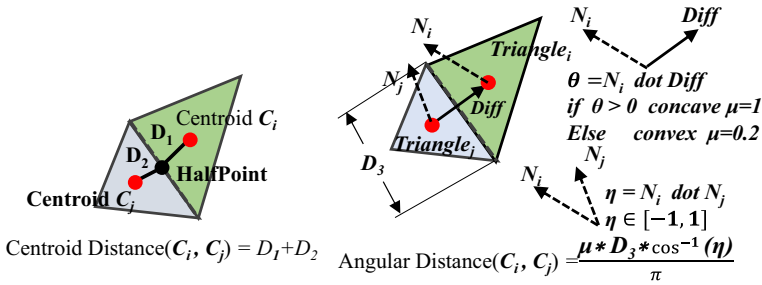


Figure 3 The two-distance metrics: centroid vs. angular

these two types of distance. In the figure, N_i and N_j are the normal vectors of their triangles, **Diff** represents the difference vector between the centroids of triangles T_i and T_j , and θ denotes the angle between **Diff** and N_i and N_j (according to the direction of the vector **Diff**). We define the centroid distance as

$$d_{centroid}(i, j) = \|\vec{c}_i - \vec{c}_{se}\|_2 + \|\vec{c}_j - \vec{c}_{se}\|_2 \tag{1}$$

where SE is the shared edge of triangles T_i and T_j , and \vec{c}_i and \vec{c}_{se} are the centroid of triangle T_i and the middle point of edge SE , respectively. The angular distance is defined as

$$d_{angular}(i, j) = \frac{\mu * |SE| * \cos^{-1}(\eta)}{\pi} \tag{2}$$

where μ is a control value indicating whether the surface is concave. If the dot product between the vector **Diff** and normal vector N_i of triangle T_i , the surface between triangles T_i and T_j is concave, and the value of μ is set to 1. Otherwise, the surface is convex, and the value of μ is set to 0.2. The term η is the dot product between normal vector N_i and normal vector N_j . However, we restrict the parameter η within $[-1, 1]$; i.e., if $\eta > 1$, then $\eta = 1$, else if $\eta < -1$, then $\eta = -1$. The range of angular distance is easy to find, i.e., $0 < d_{angular}(i, j) < |SE|$.

The global distance d_{global} is then defined as a weighted sum of the two distances

$$d_{global}(i, j) = \frac{d_{centroid}(i, j) + \varphi * d_{angular}(i, j)}{\Delta} \tag{3}$$

where Δ denotes the diagonal length of the bounding box of whole model. Here, the weight φ is an empirical weight that determines the ratio of the angular distance to the global distance. In general, the larger φ is, the more streamlined the mesh model becomes. Because we need to assure the global distance be a normalized value ($0 \leq d_{global}(i, j) \leq 1$), by many experiments, we must restrict the term φ within $[0, 500]$. In general, the term φ can be set to a random value in this range. In this paper, for huge man-made model, we find that bigger value of the parameter φ , the result is better, therefore, we set $\varphi = 300$.

4.1.3 Regional diffusion and fragments removal

The process of regional diffusion is shown in Figure 4.

For diffusion source T_0 , we conduct the task of initialization to construct a Dijkstra distance table \mathcal{D} . Except for the source T_0 , all the values are infinity. Then, the distance of every triangle in Dijkstra distance table \mathcal{D} can be computed is as Eq. 4,

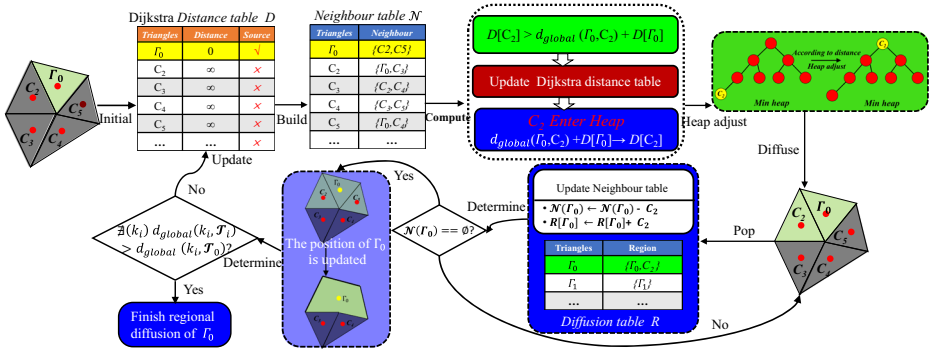


Figure 4 The process of regional diffusion from source T_0

$$D[x][distance] = \begin{cases} 0 & x \in T \\ \min_{0 \leq i < |T|} d_{global}(x, T_i) & x \in \mathcal{K} \end{cases} \quad (4)$$

where, the term $|T|$ denotes the size of the source set T .

Regional diffusion continuously updates the value of distance table \mathcal{D} until diffusion is completed. Furthermore, Neighbour table \mathcal{N} is used to represent their neighbour information of every triangle, and the diffusion table \mathcal{R} is used to denote the diffusion information of every source. In fact, the final segmentation result can be obtained by iteratively updating this table. Clearly, the position of every source also need to continuously updated.

The result of regional diffusion is the segmentation file. In this file, each triangle is assigned to at least one region that has been recorded. The final task is to split the mesh according to the index information given in the file. Because of the many shared edges and vertices, (many) edges and vertices belong to many different regions. Therefore, the vertices and triangles must be re-indexed to split the whole mesh. The unique indexing for the triangles can be easily determined (as each triangle can belong to only *one* region). However, the indexing of edges and vertices is challenging. We use the half-edge data structure (Kettner [16]) to denote the vertex-triangle relationship and construct a list to build the relationships between regions and triangles. To re-index the vertices, we visit them from a triangle index and obtain all the shared edges from each component. Nevertheless, many redundant components, which can be called *fragments*, exist.

The scale of every fragment is very small and has a minimal effect on the restoration or structure of the whole mesh. However, because the number of fragments is often large, process fragments, including repetition detection, consumes a large amount of time. Moreover, in our proposed repetition detection method, the fragments are very similar, but it is almost impossible for these repetitive fragments to be restored via geometric transformation. Above all, these fragments are not similar at all. Therefore, we adopt a strategy to remove these fragments. Consider a component c_i from the whole mesh $\mathfrak{X} = \{(v_i, t_j, s) \mid 0 \leq i \leq N, 0 \leq j \leq M\}$, where s is the number of components, and N and M are the number of vertices of the whole mesh and the number of triangles of the whole mesh, respectively. When the following conditions (both Eqs. (5) and (6)) are satisfied, c_i is identified as redundant and is subsequently abandoned. The number of triangles in every component is controlled by Eq. (5), but in some cases, some small components may not be fragments. In this case, the variance of the area of every component is considered via Eq. (6). If a component is very small but its area variance is large, it can be viewed as a fragment.

$$\Psi(c_i) \leq \min \left(\alpha * \frac{1}{s} \sum_{i=1}^s \Psi(c_i), \beta * M \right) \tag{5}$$

$$\left(\mathcal{H}(c_i) - \frac{1}{s} \sum_{j=1}^s \mathcal{H}(c_j) \right)^2 \geq \xi \left(\frac{1}{s} \mathcal{H}(\mathbf{x}) - \frac{1}{s} \sum_{j=1}^s \mathcal{H}(c_j) \right)^2 \tag{6}$$

where Ψ and \mathcal{H} represent the number and the area of the triangles, respectively, and α, β, ξ are empirical terms. Experiments show that good results are obtained when these terms are, respectively, set to 10%, 0.5%, and 50.

An overview of the re-indexing and outputting components process is shown in Figure 5.

Figure 5 illustrates the steps of how to output components based on segmentation. The above steps have conducted the task for tagging every triangle, and the related segmentation file can be obtained. In this segmentation file, every triangle would be tagged as a part of a component. Then a half-edge data structs [16] is used to better visit related triangles. However, the task of outputting components need to split whole mesh into many individual models, re-indexing mesh is a necessary step. The shared edges between every segmentation should be obtained, so that the triangle of the shared edges must be re-indexed. In this paper, the OBJ format is used to output mesh, therefore, every triangle index need to be re-assigned. Under this circumstance, by means of half-edge data structs, the triangle is re-indexed. Finally, we need to remove these fragments, as the above mention, the fragments would affect the result of repetitive detection and put little effect on the final restoration over Web browser. Moreover, the related components can be outputted as individual models. Hence, the task for segmentation of whole mesh is completed.

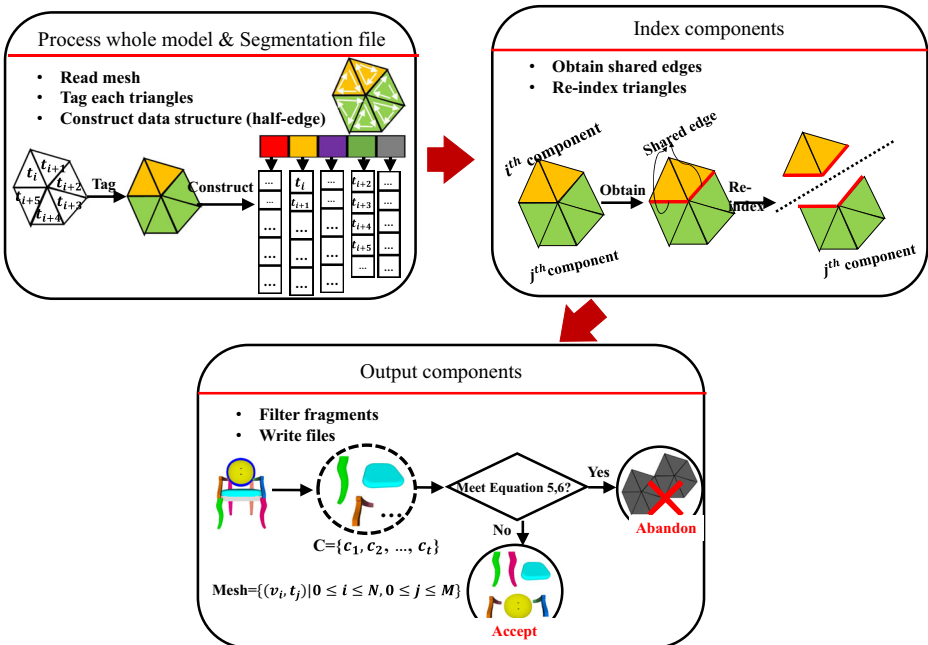


Figure 5 Re-indexing and outputting components

4.2 Voxel-based repetition detection

After mesh segmentation, we have the segmented components. Our next task is to identify repetitive components. The word “repetition” here means identical or nearly identical components subject to rigid body transformations. Typically, man-made models have a high degree of component repetition. Before applying our proposed voxel-based repetition detection procedure to the components, we filter out relatively small (thus less significant) components, specifically, a component is exempted if its surface area (i.e., the summation of the areas of its triangles) is less than 2% of the total surface area of the model. Then, for the remaining components, we first compare the surface areas of two components A and B in pairwise manner; if their difference is less than a given threshold, voxel-based similarity checking is performed, this is pre-processing step. In this step, we need to obtain the possible repetition components set \mathcal{P} . For random two components c_i, c_j , according to Eq. 7, the possible repetition pair p_k can be confirmed,

$$\frac{|A(c_i) - A(c_j)|}{\max\{A(c_i), A(c_j)\}} < \delta \quad (7)$$

where the function A is to compute the area of component, the term δ is a given threshold, because repeated components maybe exist some deformation, their area maybe are not totally same. By many different experiments, we set this threshold $\delta = 0.01$. Under this circumstance, any two components that meet the condition (Eq. 7) can be formed as a pair of possible repetition, which can be mathematically denoted as the term p_k . Finally, we can obtain the possible repetition components set $\mathcal{P} = \{0 \leq k < n | p_k\}$ (the term n is the size of the set \mathcal{P}).

We first need to determine the size of the voxels used to represent A and B; clearly, A and B must have the same number of voxels. We set the base voxel size to $0.01 * 0.01 * 0.01$ and the voxel precision to 0.001 to facilitate the computation. In this case, every voxel consists of k base voxels, where k depends on the size of the original model. In fact, k is the same for every component. Geometrically, each voxel is defined by 6 square faces and 12 triangles. Hence, a 12-triangle index structure must be built to index any voxel cube. We can identify every triangle using 3 integers as its index, which is a popular indexing format adopted by many commercial file systems, such as OFF and OBJ. Then, we can calculate the volume of every voxel cube based on these index sequences. Clearly, the index sequences are not unique. The process of computing the volume of a voxel cube is illustrated in Figure 6.

For every component, we can then utilize the above method to obtain the total volume of all the voxel cubes. As the voxel sequences of all the components are identical, sequential comparison can be performed for every component pair A and B. Let Δ denote the binary function of comparing the volumes of two corresponding voxel cubes in A and B that is defined as

$$\Delta(i) = \begin{cases} 1 & \text{if } |Vol(A_i) - Vol(B_i)| \leq \zeta \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $Vol(A_i)$ and $Vol(B_i)$ represent the volumes of the i^{th} voxel cubes of A and B, respectively, $i = 1, 2, \dots, N$ (N is the total number of voxels), and ζ is the voxel precision, which we set to 0.001. In this paper, the number of voxel cubes is set to 4096, that is, $N = 4096$. Therefore, every component is represented by 4096 voxel cubes, which greatly simplifies the comparison process.

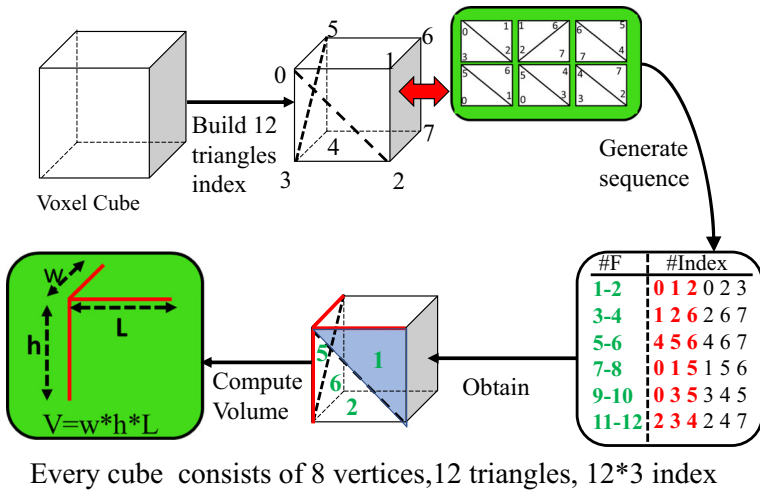


Figure 6 Calculating the volume of a voxel cube

The similarity measurement $S(A, B)$ between any two components A and B uses the simple arithmetic average of $\Delta(i)$, which is defined as

$$S(A, B) = \frac{1}{N} \sum_{i=1}^N \Delta(i) \tag{9}$$

That is, A and B are similar if and only if $S(A, B) > \eta$. In this paper, the threshold η is set at 0.85. Figure 7 shows a flowchart of the entire process. We can find that the entire process can be divided into two steps. First, we conduct the pre-processing step to obtain the related possible repetition set \mathcal{P} . In this set, there are many components pair which consists of every two components. Second, the iterative operation is performed to check all components pair in this set \mathcal{P} , the similarity of every pair can be measured as Eq. 9. After all components pairs are checked, the repeated components can be confirmed. Hence, we can output all repetitive components.

4.3 Pose alignment

Pose Alignment method was proposed by WEN et al. [32]. We also use this method in this paper. Pose alignment includes translation, scaling, and symmetry operation. By these operations performing on related repetition components, we can obtain a transformation matrix, by which we can restore the removed repeated components. Specially, to conduct alignment operation, we need to put these components into a unit space. In this way, it can improve the efficient of transformation.

Figure 8 illustrates the overview of pose alignment (dash line represents reverse operation). After performed the operation of normalization, we need to determinate whether the pair of components align each other in unit space, if they do not align, then the symmetry operation is conducted, and then continue to determinate the relation of alignment. Under this circumstance, if they still not, then they are non-repetitive components. Once the components confirm the relation of alignment, the reverse operation would be performed to obtain related matrix. Finally, the repeated component is removed and the transformation matrix can better be an alternative to repetitive component.

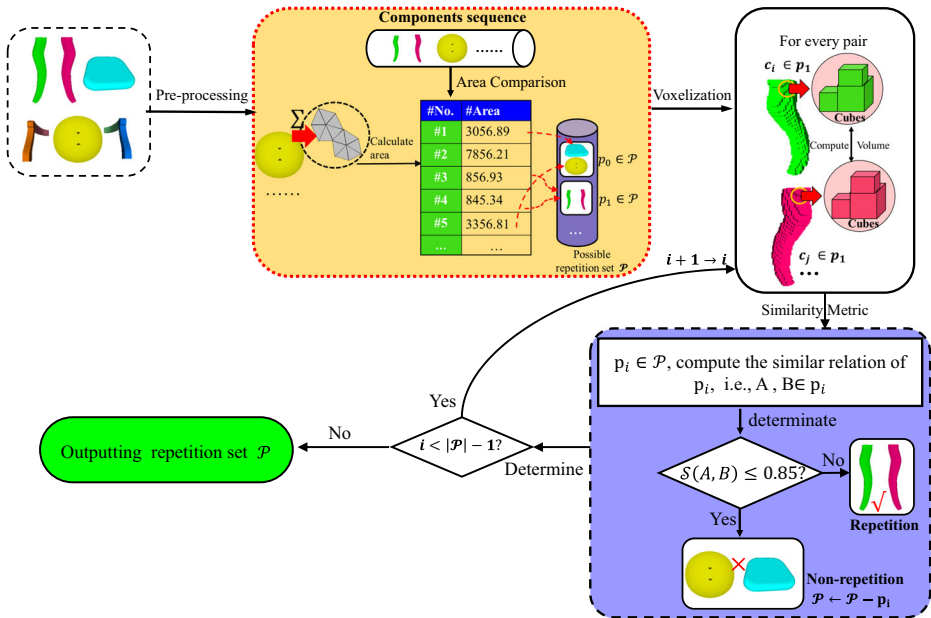


Figure 7 The process of repetition detection

It is not hard to find that the entire of pose alignment method can be divided into two parts, i.e., pre-processing step and transformation step. The aim of pre-processing step is to put these components into a unit space, i.e. pose normalization. To achieve the task of pose

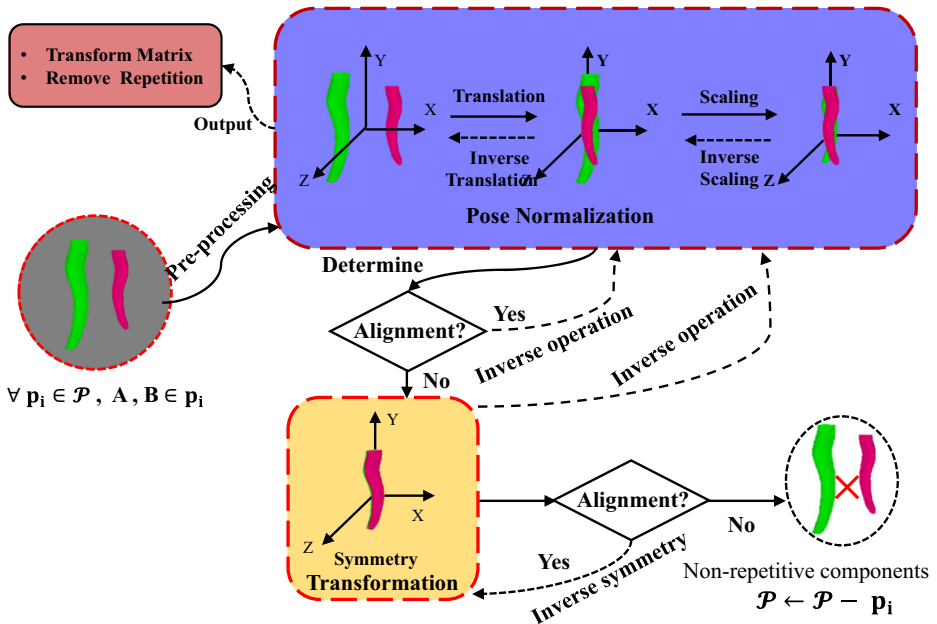


Figure 8 The process of pose alignment

normalization, for every component in repetition pair, we perform translation and scaling operation, respectively.

For the component $c_i \in p_i$, the processing of translation is as Eq. 10,

$$m = \frac{1}{E} \sum_{t_i \in C_i} E_i \frac{(x_i + y_i + z_i)}{3} \tag{10}$$

where the term m is the reference size of translation, and the parameter E is the area of the component c_i . Moreover, the term t_i is the i^{th} triangle of the component c_i . The terms x_i, y_i, z_i denote the x-coordinates, y-coordinates, z-coordinates, respectively. Besides, the term E_i is the area of triangle t_i . Next, the scaling equation is as Eq. 11,

$$S = \frac{1}{\sqrt{s_x^2 + s_y^2 + s_z^2}} \mathbf{I} \tag{11}$$

$$s_x = \frac{1}{|V|} \sum_{v \in V} |v_x| \tag{12}$$

where, the term S is scaling matrix, accordingly, the term s_x represents the scaling value in x-axis. In Eq. 12, the term $|V|$ is the vertices number in the component c_i . Surely, the variable v_x is the x coordinates of the vertex v . Likewise, we can compute the scaling matrix in y-axis and z-axis. In addition, the term \mathbf{I} denotes an identity matrix.

Furthermore, symmetry operation is as Eq. 13,

$$F = \text{diag}\left(\text{sign}(f_x), \text{sign}(f_y), \text{sign}(f_z)\right) \tag{13}$$

$$f_x = \sum_{v \in V} \text{sign}(v_x) v_x^2 \tag{14}$$

where the term F denotes the diagonal matrix, the function diag generates a symmetric

tridiagonal matrix, i.e. $\text{diag}(1, 0, 1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, in addition, the function $\text{sign}(\alpha) = 1$, if $\alpha \geq$

0, otherwise, $\text{sign}(\alpha) = 0$. Through Eq. 14, we can obtain the symmetry value f_x in x-axis. Likewise, the symmetry value f_y, f_z can be obtained. In practice, for majority of man-made models that exist many repetitive components, symmetry operation is used to complete the transformation task.

To determine the relation of alignment, we use the approach of voxel alignment method, instead of AABBs method. After conducting pose normalization operation, for every two components, the position of their corresponding voxel cube are compared. Equation 15 is a decision function to determine the relation of alignment. For two component A, B , their corresponding voxel cube can be represented as the term A_i, B_i ,

$$\Theta(A, B) = \begin{cases} 1 & \text{if } \frac{1}{N} \sum_{i=0}^{N-1} \prod_{j=0}^8 \Psi\left(\vec{A}_i, \vec{B}_i\right) \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$\Psi\left(\vec{A}_i, \vec{B}_i\right) = \left\| \vec{A}_i - \vec{B}_i \right\|_2 = \begin{cases} 1 & \text{if } \Psi\left(\vec{A}_i, \vec{B}_i\right) \leq \theta \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

where the terms \vec{A}_i, \vec{B}_i represent vertex vector in voxel cube A_i, B_i , respectively. The Euclidean distance is used to measure their differences. The term θ is the voxel precision, which we set to 0.001. That is, A and B aligned each other if and only if $\Theta(A, B) = 1$. The term ϵ still is experimental value, by experiments, we set the term $\epsilon = 0.95$. Likewise, N is the total number of voxels, it is set to 4096, that is, $N = 4096$.

4.4 Fine-grain scene graph (FG-SG)

At this point, for a given mesh model, we have segmented it into individual components and identified repetitions. For any group of identical components obtained by pose normalization and geometrical transformation, only one representative needs to be transmitted over the Web, thereby substantially decreasing the transmission time. On the other hand, to further increase the transmission efficiency and to target models with few repetitions, we employ progressive meshing and multi-thread loading.

Hopper [8] proposed the general algorithm for progressive meshing (PM) during decimation, which made substantial improvement relative to classic decimation methods. Hopper's algorithm was the first algorithm that employed the edge collapse operator. However, the algorithm difficulties in distinguishing important shape features, such as high-curvature regions. Therefore, the PM method is typically better suited for a single smooth component. In this paper, we utilize the PM method for mesh transmission. In the initial stage, the Web browser loads a base mesh structure, which is then progressively restored to the original whole mesh structure via the edge split operation.

In addition, every individual component can be independently and asynchronously loaded in its sub-thread and have its data sent to the main thread. In this way, the impact of data loading on mesh rendering can be reduced to near its minimum.

For a better realization of the proposed framework over Web browsers, we define a fine-grained scene graph file (FG-SG) to indicate which components should be operated by the PM method and which components should be processed by repetition transformation. The structure of FG-SG and the process of parsing an FG-SG file are shown in Figure 9.

Figure 9 illustrates the structure of fine-grain scene graph (FG-SG), for a complex scene \mathcal{S} , there are many different models, it can be represented as follows, $\mathcal{S} = \{1 \leq i \leq n | M_i\}$, then for every model $M_i = \{1 \leq k \leq K | c_k\}$, where the term K represents the number of components. Clearly, different models include different the number of components. In addition, the model M_i can be represents as follows, $M_i = \{\mathcal{P}_i, \mathcal{W}_i\}$, where the term \mathcal{P}_i represents the repetition components set, and the term \mathcal{W}_i denotes the non-repetition set. Moreover, for the repetition components set $\mathcal{P}_i = \{1 \leq u \leq U | p_i^u\}$, $\forall p_i^u$, it is represented a component and transformation matrix T_i^u . Besides, by the progressive mesh (PM) method, the component can be reduced. We assume the ratio of PM $\theta = 0.5$, then the ratio λ of the size of the processed model to the original model size can be calculated as follows, $\lambda = \frac{\theta[\frac{U}{2} + (K-U)]}{K} = \frac{2K-U}{4K} < 0.5$, where the term K represents the number of components, the term U represents the number of repetition components pair, i.e., the number of repetition set \mathcal{P}_i . In addition, the parameter θ represents the ratio of PM operation. In this paper, this parameter θ can be set to 0.5. Obviously, the more repetitive components in the model, the more the size of model is reduced by our method.

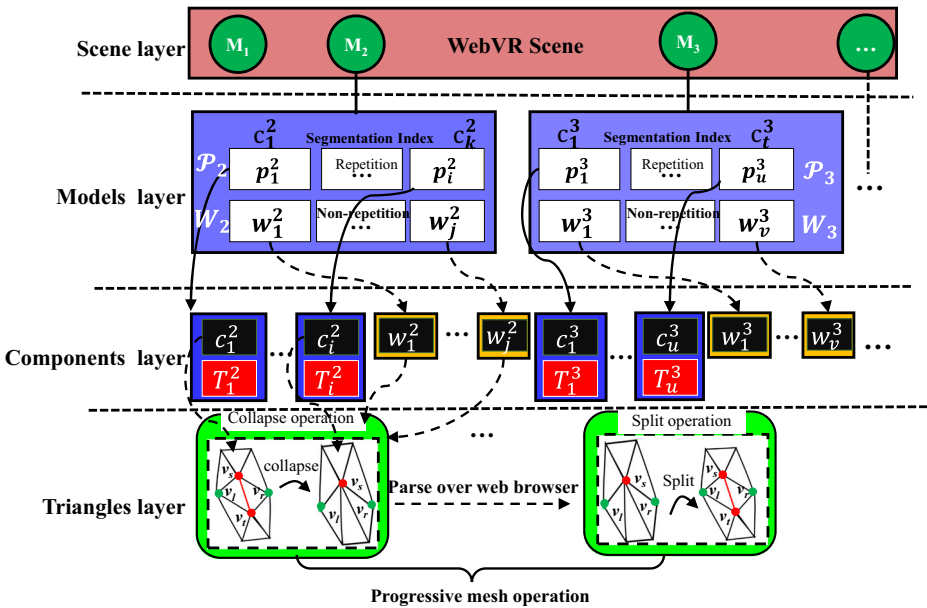


Figure 9 Overview of building and parsing an FG-SG file

5 Experimental results

We implemented a prototype of the S-LPM framework using C++. In this section, we report the results of a batch of experiments performed to validate the proposed methodology. Specifically, we tested the proposed mesh segmentation method and the repetition detection method and finally displayed the results on a Web browser. To evaluate our segmentation algorithm, we compared our method with state-of-the-art approaches on Watertight Track from the SHREC 2007 dataset, which is a popular test dataset used by many mesh segmentation systems/solvers. The experiments were conducted on a PC running Windows 7 OS with an Intel core I5-M580 processor and a 4 GB of memory.

5.1 Mesh segmentation

We compare our proposed method with state-of-the-art methods on the Watertight Track from the SHREC 2007 dataset, which consists of 380 mesh models that are grouped into 19 categories. The compared mesh segmentation methods are *K*-means (Shlafman et al. [28]), graph cuts (Katz et al. [14]), random walks (Lai et al. [18]), and core extraction (Katz et al. [15]).

Table 1 Comparison of computing time (measured on a 2.4 GHz PC)

Segmentation algorithm	Speed (s)	Evaluation
Graph cuts	43.2	Slow
Core extraction	18.3	Medium
Random walks	0.9	Fast
K-means	1.6	Fast
Ours	1.8	Fast

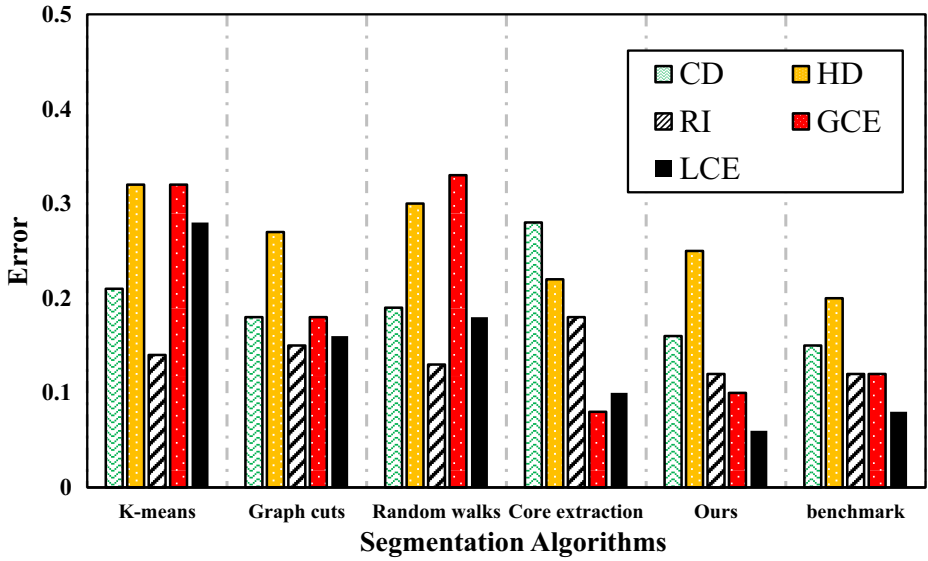


Figure 10 Comparison in terms of five different indicators

Table 1 presents the comparison of the total computer time consumed. In terms of running time, our method is relatively fast and on par with the *K*-means method; therefore, our method is expected to be particularly suitable for mobile and Web browser environments.

Additional comparison tests were performed on a benchmark set for 3D mesh segmentation proposed by CHEN et al. [6]. This benchmark set comprises 4300 manually generated segmentations of 380 surface meshes in 19 object categories. Five different indicators are proposed by CHEN et al. [6]: cut discrepancy (*CD*), Hamming distance (*HD*), rand index (*RI*),







Whole model	Component Pairs	Similarity	Human judgment (Benchmark)
		0.899131	✓
		0.807221	✗
		1	✓
		0.49873	✗
		0.3892	✗

Figure 11 An example of the repetition detection test on the chair model

Table 2 The sizes of the test models

Models	#V	#F	#Size (MB)
Chair	15,324	29,516	0.982
Bench	10,288	42,304	0.98
Slatted bench	50,679	133,968	4.02
Bike	69,515	130,446	5.28
Suit	242,791	455,494	66.6

global consistency error (*GCE*), and local consistency error (*LCE*). The comparison results in Figure 10 show that our proposed method is closest to the benchmark.

5.2 Repetition detection

The test for repetition detection is focused on the correctness and robustness of the results. Figure 11 provides a test example to illustrate how closely our method is to human judgement (i.e., the ground truth).

5.3 Overall evaluation

Five different models, whose sizes are given in Table 2, were used to test our overall system for displaying a complex mesh model on a Web browser.

Figure 12 shows the chair model processed by our system after the repetitive components are automatically detected and removed. As a result of the data reduction, only 0.267 MB data is transmitted through the internet, compared to the original 1 MB. The Web browser then performs pose alignment (cf. [32]) to correctly restore the original complete 3D shape. Even including the model restoration time on the browser side, the total computer time consumed to display the whole chair on the screen for our method is approximately 1/3 that of the original. Three more examples are given in Figure 13.

The proposed framework mainly includes mesh segmentation, repetition detection, pose alignment, and restoration in a Web browser. We performed additional related experiments on

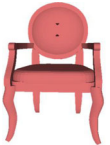
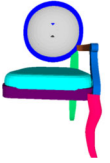
#Models		#Size(MB)
Original		1
Processed by S-LPM		0.267

Figure 12 Reduction of the data by our repetition detection/removal method on the chair model







Mesh Names	#Mesh		#Mesh	
	Original	#Size(MB)	Processed by S-LPM	#Size(MB)
Bench		0.98		0.58
Slatted bench		4.02		1.59
Bike		5.28		2.85

Figure 13 Data reduction by our S-LPM framework

the SHREC’07 dataset to further evaluate our proposed framework. Figure 14 shows that the overall accuracy of our proposed framework is higher than the segmentation accuracy. The accuracy of framework does not depend on segmentation, because the aim of segmentation is to achieve repetition removal. When low repetition components are detected, by multithread loading many different components, our framework still achieves the visualization task over Web browser. Therefore, the high accuracy of framework can still be obtained, whereas, more time maybe spent to visualize the model over Web browser.

To further validate the advantages of our proposed framework, we compared it with the method by Cai et al. [5], which identifies similarities by degenerating the shape’s PCAs. Additionally, as a comparison benchmark, we recorded the time required to directly load the original large model in the browser – let it be called the *direct display* mode. Figure 15 shows

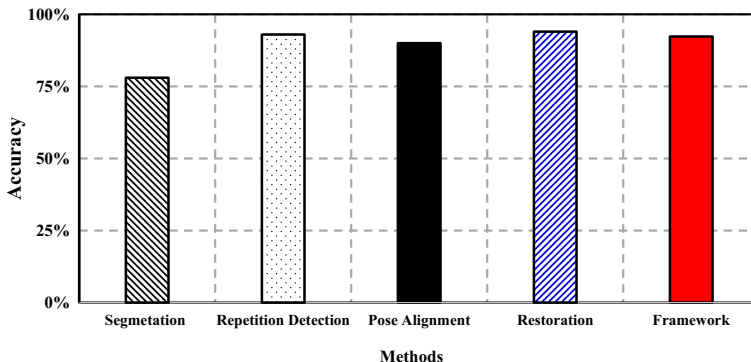


Figure 14 Accuracy comparison of every method and the overall framework for the SHREC’07 dataset

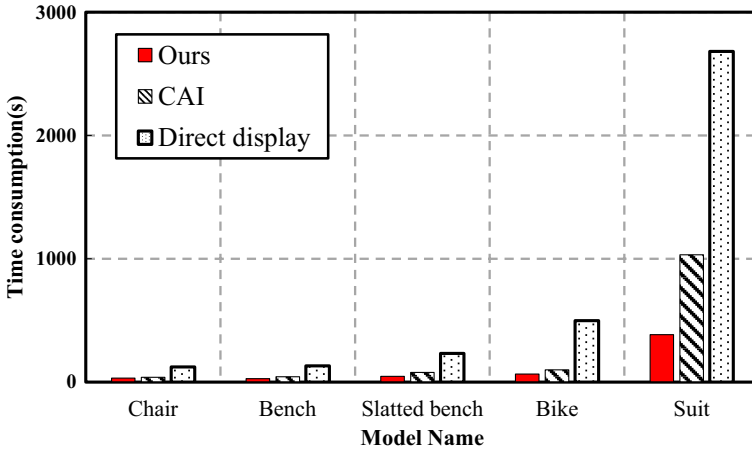


Figure 15 Comparison results for five different 3D models (measured on a 2.4 GHz PC, 4 M bandwidth)

the time consumed by the three methods. As clearly shown in the figure, when the scale of the model is not large, our method and Cai’s method are on par with each other. However, when the size of the model becomes large, such as the suit model, the advantage of our method becomes clear. Besides, we compared our framework with the above methods in accuracy indicator. Where the term A, B represented the number of triangles of original mesh, and the actual number of triangles over web browser, respectively. The accuracy indicator is denoted as following Eq. 17. The comparison result over accuracy criterion can be seen in Figure 15.

$$\text{Accuracy} = \frac{B}{A} \tag{17}$$

Figure 16 shows the comparison result for five different models, we find that accuracy of our framework is very approaching to other methods. i.e., the accuracy is greater than 95%, for huge man-made model, because these tiny differences are very difficult to distinguish by human eye, this result can be totally accepted.

Figure 17 shows how the entire mesh of the suit is gradually displayed on the screen on the browser side. Initially, in step (1), only a few components have been loaded rather than the whole model. As already emphasized, a component-wise strategy is required because the browser typically has little storage capacity and limited computing ability. When repetitive

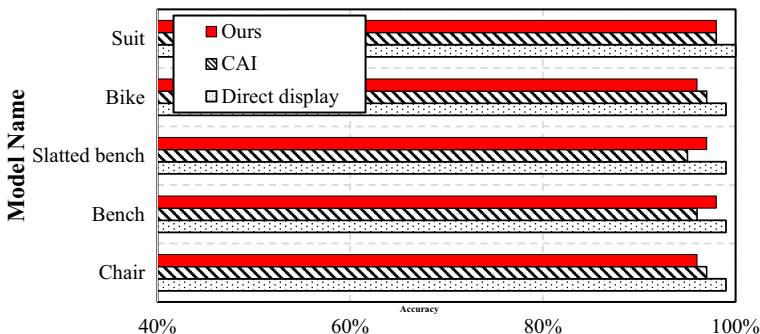


Figure 16 Comparison results for five different models on accuracy indicator

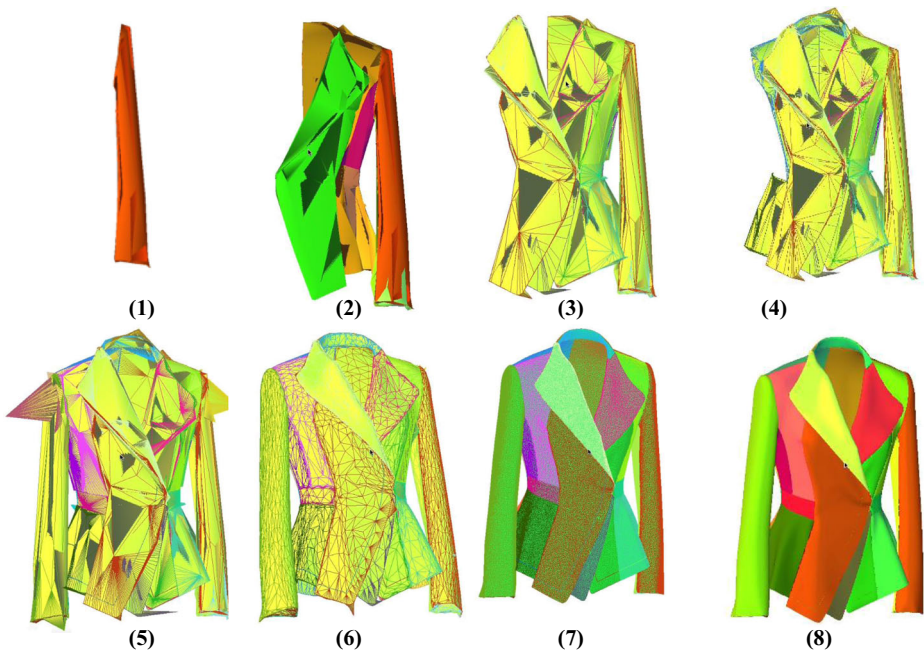


Figure 17 The sequence of progressive loading and displaying of the suit model over the Web browser

components need to be displayed (steps (3)–(4)), instead of being loaded, they are obtained via the transformation quaternion. Moreover, after the rendering task of the base mesh is completed, PM is performed to restore (from step (5) to step (8)) the final model. The mesh structure changes in steps (5)–(7), which in turn increases the density of the mesh. Therefore, the size of the current mesh progressively increases until reaching the original size (step (8)) when the model is completely restored and displayed. For steps (1)–(8), the size of the loading and rendering is 3%, 20%, 30%, 33%, 50%, 60%, 80%, and 100%.

5.4 Examples test

To further validate our proposed methodology, especially on large models, two additional models, a coat and a shirt (see Table 3 for their sizes), were tested. Figures 18 and 19 show the progressive loading and browser display of the two models; the original model is completely restored and displayed on the screen at step (8).

As a final test, a scene with many different man-made models was loaded and displayed using the proposed framework, and the results are shown in Figure 20. In this test, our method reduces the size of the scene by 51%, which translates into a substantial increase in the loading and displaying speed of the whole model.

Table 3 Two large models

Model	#V	#F	#Size (MB)
Coat	184,879	298,866	33.2
Shirt	151,984	221,722	25.4

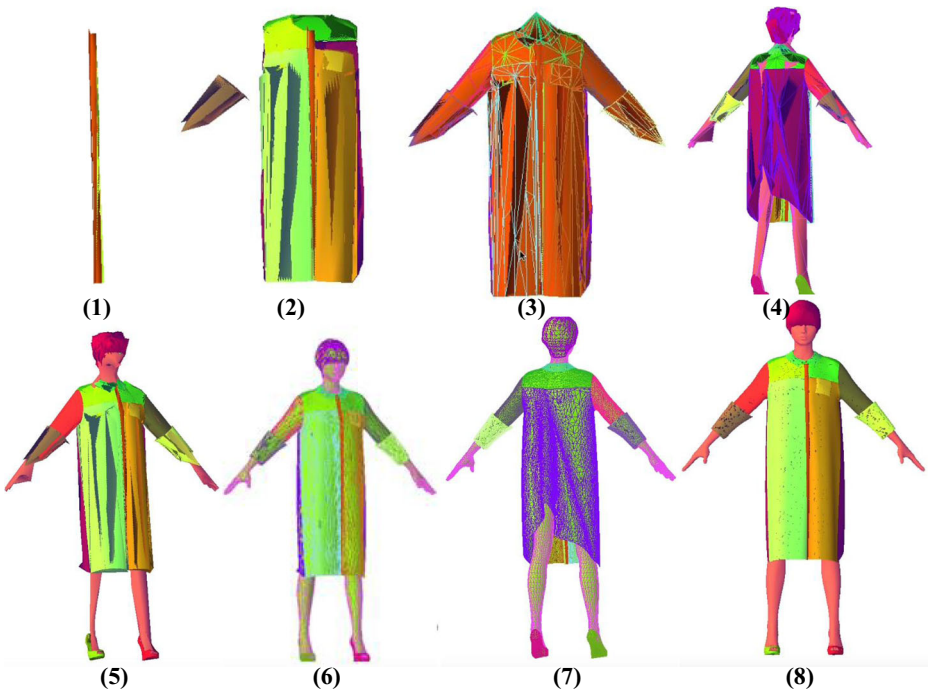


Figure 18 Progressive loading and display of the coat

6 Conclusions

In this paper, we have proposed a new lightweight framework – the S-LPM framework – to process large models to improve the efficiency of their transmission through the net and their



Figure 19 Progressive loading and display of the shirt



Figure 20 A scene with many different man-made models

visualization in Web browsers. The proposed framework comprises two key technical components, i.e., Dijkstra-based mesh segmentation and voxel-based repetition detection, which together significantly reduce the amount of data transmitted over the net and hence the demands on computer time and memory. In addition, a fine-grained scene graph file is obtained to determine for which components to adopt the PM method for data transmission and for which repeated components to perform the transformation from the mesh level to the component level. Experiments performed on some representative models validated the advantages of the proposed framework over some traditional methods, such as direct loading and display.

Regarding future research, first, our mesh segmentation algorithm currently crucially depends on the geometric topology but ignores the semantics. It will be interesting to see how considering semantics could help the segmentation. Second, certain unique geometric properties, e.g., symmetry, should be explored to simplify either the mesh segmentation or the repetition detection. Finally, we need to test our framework on models with unique features, such as buildings or furniture, to determine whether modification is required to adapt the models.

Acknowledgments The authors appreciate the comments and suggestions of all anonymous reviewers, whose comments helped significantly improve this paper. This work is supported by the Fundamental Research Funds for the Central Universities in China (2100219066) and the Key Fundamental Research Funds for the Central Universities in China (0200219153).

References

1. Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., Azariadis, P.: 3D mesh segmentation methodologies for CAD applications [J]. *Comput.-Aided Des. Applic.* **4**(6), 827–841 (2007)
2. Aleksey, G., Funkhouser, T.: Consistent segmentation of 3D models [J]. *Comput. Graph.* **33**(3), 262–269 (2009)
3. Attene, M., Katz, S., Mortara, M., et al.: Mesh segmentation - a comparative study [C]. *IEEE International Conference on Shape Modeling and Applications*, pp. 7–7. DBLP (2006)
4. Cai, K., Wang, W., Chen, Z., et al.: Exploiting repeated patterns for efficient compression of massive models [C]. *VRCIA*, pp. 145–150. ACM (2009)

5. Cai, K., Teng, J., Teng, J., et al.: Exploiting repeated patterns for efficient compression of massive models [C]. International Conference on Virtual Reality Continuum and ITS Applications in Industry, pp. 145–150. ACM (2009)
6. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation [J]. *ACM Trans. Graph.* **28**(3), 1–12 (2009)
7. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical face clustering on polygonal surfaces [C]. Symposium on Interactive 3d Graphics, Si3d 2001, Chapel Hill, Nc, Usa, March, pp. 49–58. DBLP (2001)
8. Hoppe, H.: Progressive meshes [J]. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques [C], ACM SIGGRAPH '96, pp. 99–108 (1996)
9. Hu, R., Fan, L., Liu, L.: Co-segmentation of 3D shapes via subspace clustering [C]. *Comput. Graphics Forum.* Blackwell Publishing Ltd, pp. 1703–1713 (2012)
10. Huang, Q., Koltun, V., Guibas, L.J., et al.: Joint shape segmentation with linear programming [C]. International Conference on Computer Graphics and Interactive Techniques, **30**(6) (2011)
11. Inoue, K., Itoh, T., Yamada, A., Furuhashi, T., Shimada, K.: Face clustering of a large-scale CAD model for surface mesh generation [J]. *Comput. Aided Des.* **33**(3), 251–261 (2001)
12. Isenburg, M., Lindstrom, P., Snoeyink, J.: Streaming compression of triangle meshes [C]. In Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP '05) (2005)
13. Kalogerakis, E., Chaudhuri, S., Koller, D., et al.: A probabilistic model for component-based shape synthesis [J]. *ACM Trans. Graph.* **31**(31), 1–11 (2012)
14. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts [J]. *ACM Trans. Graph.* **22**(3), 954–961 (2003)
15. Katz, S., Leifman, G., Tal, A., et al.: Mesh segmentation using feature point and core extraction [J]. *Vis. Comput.* **21**(8), 649–658 (2005)
16. Kettner, L.: Using generic programming for “designing a data structure for polyhedral surfaces” [J]. *Comput. Geom.* **13**(1), 65–90(26) (1999)
17. Krevov V, Julius D, Sheffer A. Model composition from interchangeable components [C]. *Computer Graphics and Applications*, 2007 PG'07. 15th Pacific Conference on. IEEE. 129–138 (2007)
18. Lai, Y., Hu, S., Martin, R., et al.: Rapid and effective segmentation of 3D models using random walks [J]. *Comput. Aided Geom. Des.* **26**(6), 665–679 (2009)
19. Lin, H.S., Liao, H.M., Lin, J., et al.: Visual salience-guided mesh decomposition [J]. *IEEE Trans. Multimedia.* **9**(1), 46–57 (2007)
20. Liu, R., Zhang, H.: Segmentation of 3D meshes through spectral clustering [C]. *Pacific Conference on Computer Graphics and Applications*, pp. 298–305 (2004)
21. Liu, X., et al.: Low-rank 3D mesh segmentation and labeling with structure guiding [J]. *Comput Graph.* **2015**, 99–109 (2015)
22. Meng, M., Xia, J., Luo, J., He, Y.: Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization [J]. *Comput. Aided Des.* **45**(2), 312–320 (2013)
23. Mortara, M., Patane, G., Spagnuolo, M., et al.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies [J]. *JISS*, pp. 339–344 (2004)
24. Saleem, M., Kamdar, M.R., Iqbal, A., Sampath, S., Deus, H.F., Ngonga Ngomo, A.C.: Big linked cancer data: integrating linked TCGA and PubMed [J]. *Web Semantics Science Services & Agents on the World Wide Web.* **27–28**, 34–41 (2014)
25. Savelonas, M.A., Pratikakis, I., Sfikas, K.: An overview of partial 3D object retrieval methodologies [J]. *Multimedia Tools and Applications.* **74**(24), 11783–11808 (2015)
26. Shamir, A.: Segmentation and shape extraction of 3D boundary meshes [C]. *State of the Art Report Eurographics* (2006)
27. Shikhare, D., Bhakar, S., Mudur, S.P.: Compression of large 3D engineering models using automatic discovery of repeating geometric features [C]. *Vision Modeling and Visualization Conference*, pp. 233–240. Aka GmbH (2001)
28. Shlafman, S., et al.: Metamorphosis of polyhedral surfaces using decomposition [J]. *Comput. Graphics Forum.* **21**(3), 219–228 (2002)
29. Sidi O, Kaick O V, Kleiman Y, et al.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering [J]. *ACM Trans. Graph.* **30**(6), 1–10 (2011)
30. Theologou, P., Pratikakis, I., Theoharis, T.: A review on 3D object retrieval methodologies using a part-based representation [J]. *Comput.-Aided Des. Applic.* **11**(6), 670–684 (2014)
31. Theologou, P., Pratikakis, I., Theoharis, T., et al.: A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation [J]. *Comput. Vis. Image Underst.* **135**, 49–82 (2015)
32. Wen L., Xie N, Jia J. Fast accessing Web3D contents using lightweight progressive meshes [J]. *Comput. Anim. Virtual Worlds.* **27**(5), 466–483 (2016)

33. Xu, K., Li, H., Zhang, H., et al.: Style-content separation by anisotropic part scales [J]. *ACM Trans Graph.* **29**(1), 184 (2010)
34. Zhang, H., Gao, X., Wu, P., et al.: A cross-media distance metric learning framework based on multi-view correlation mining and matching [J]. *Web Semantics Science Services & Agents on the World Wide Web.* **19**(2), 181–197 (2016)