Received July 17, 2018, accepted August 18, 2018, date of publication August 21, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866508

A Novel Compression-Driven Lightweight Framework for Medical Skeleton Model Visualization

WEN ZHOU[1], (Member, IEEE), JINYUAN JIA[1], AND XIN SU[2]
[1]School of Software Engineering, Tongji University, Shanghai 201804, China
[2]College of Electronics and Information, Tongji University, Shanghai 201804, China

Corresponding author: Wen Zhou (zhouwen327@163.com)

This work was supported in part by the Fundamental Research Funds for the Central Universities, China, under Grant 2100102002, Grant 21002450051, and Grant 2100219066, and in part by the Key Fundamental Research Funds for the Central Universities, China, under Grant 0200219153.

ABSTRACT In the field of medical imaging, many medical image data can be rebuilt into 3-D medical models for use in medical education and analysis. In fact, the 3-D medical models help ordinary people (including patients and students) to more easily understand and learn medical knowledge. In addition, with the rapid development of Web3D technology, the demand for 3-D visualization technology based on the web browser increases. However, the limited bandwidth and low load capacity of the browser have seriously restricted the development of technology. Therefore, this paper proposes a framework to better show 3-D skeleton shapes. In particular, mesh compression is one of the most effective methods to achieve a fast transmission data process. In addition, because of the poor rendering capabilities of the browsers, it is notably difficult to render the entire model at once. Moreover, it often makes the browser crash; thus, the transmitted model data are only rendered once. In addition, a mesh segmentation algorithm is proposed to realize component-wise rendering and lightweight for shape. A voxel-based component repetition method is used to detect the repetitive components; thus, we can perform matrix transformation to finish the repetitive-component Web3D visualization, i.e., lightweight for shape. Finally, the related experiments are performed to validate our proposed framework. The results show that the proposed framework is feasible and superior.

INDEX TERMS Lightweight, mesh compression, medical images, segmentation, Web3D, visualization.

I. INTRODUCTION

With the arrival of the ''Internet plus'' era, increasing focus has been put on the Web3D technology. However, for the mobile Internet, which is subject to restrictions on bandwidth and other conditions, it is unrealistic to show a model on a webpage similar to a picture in real time. This bottleneck problem is one of three Web3D issues and has hampered the development of the Web3D technology. Many research scholars have presented their methods to solve this problem. Zhao *et al.* [1] proposed the model-loading strategy according to the adaptive bandwidth based on the level of detail (LoD). Laixiang *et al.* [2] presented the lightweight loading mechanism by computing the repeatable component of the model, which was called the LPM method. Wang *et al.* [3] presented a view-dependent simplification method by computing the visible parts of the model in view frustum. Meanwhile, there are large quantities of medical data in medical science, which can be used for education and research such as medical data visualization. However, medical datasets are often large, dense and difficult to display on web browsers. Although the visualization of medical data, particularly medical skeleton data, receives little focus, there are great demands in this research area. In particular, medical skeleton visualization can be applied in clinical teaching and disease prevention. Furthermore, 3D skeleton data provide good interactive effects, e.g., in education, such data can help students better understand the skeleton structure. Compared with dull 2D image data, 3D skeletons can be interactive for students from many viewpoints. Thus, the Web3D technology can enhance these advantages. In general, the medical 3D models are large and almost impossible to directly display on web browsers. In fact, the disadvanges of existing works can be concluded as following.

First, they only focus on the whole 3D models data, rather than many components of models. Especially, there are many repetitive components in many 3D skeleton data. We can greatly low down the scale of whole model by removal of repetitive components. Whereas, they totally ignore this point. Second,mesh compression can decrease the scale of transmission of models data, in this way, we can low down the load of web browsers. However, it still consumes many time that a whole compression model is decompressed, above all, for large dense skeleton models, but in this case, the users are generally not patient to wait. In other word, exisiting methods always focus on compression for the whole mesh, rather than individual components.Last but not least, there are many repetitive components in 3D skeleton medical models, it is very suitable for conducting lightweight operation based on removal of repetitive components.Nevertheless, in many existing methods, little related researches consider this key factor. Therefore, in this paper, we incorporate mesh compression and segmentation into the 3D skeleton visualization scheme. The motivation of this paper is to present a novel solution for browser-based visualization of medical 3D data, in fact, many different methods is utilized to improve the visualization results for large dense skeleton models. in other word, we achieve the task of components-based transmission and repetitive components removing, so as to complete lightweight operation for large dense skeleton medical models. In this way, the large dense medical models data can easily show in poor web-browser.In particular, our contributions are as follows:

1. We present a mesh compression method and compress the 3D skeleton models by encoding the triangle of mesh to perform the mesh transmission. This method improves the transmission of skeleton data.
2. We present a mesh segmentation method and conduct the segmentation operation to split the entire skeleton mesh into many individual components, so that we can render through the component-wise method.
3. We detect repetitive components to make the 3D skeleton data more lightweight.

The remainder of the paper is organized as follows. In Section 2, we briefly review the related work on medical visualization, mesh compression and segmentation. In Section 3, we show our proposed framework. Section 4 shows the related algorithms. Section 5 shows the results of our experiments. Finally, in Section 6, we draw our conclusions.

II. RELATED WORKS

A. MEDICAL VISUALIZATION

Medical visualization has produced many implementations of 3D medical model visualization in web browsers. John [26] and John *et al.* [27] proposed visualization methods for educational purposes. These approximations require third-party systems for the correct visualization. Poliakov *et al.* [29] and Sun *et al.* [30] proposed the method of a rendering server to realize the medical data visualization, but it limited the scalability of the application. Jung *et al.* [31] proposed a method based on standards such as VRML and texture mapping, which achieve the visualization of volumes in the web browser. However, this method requires the users to install 3rd-party plugins. Congote *et al.* [28] presented the implementation of a volume rendering system for the web based on the WebGL technology. However, this method only aimed at medical data and did not consider how to solve the problem of dense medical data loading and rendering over web browser.

B. MESH COMPRESSION

Mesh compression is one of most challenging issues in computer graphics. Many researchers attempted to improve the compression ratio. Specially, Rossignac *et al.* [4] proposed the Edgebreaker algorithm, which encoded the connectivity of triangular meshes by iteratively nibbling its faces. Gumhold [5] proposed optimizing a Markov model to design an asymptotic optimal arithmetic coder for the Edgebreaker encoder because each Edgebreaker symbol was a state. Moreover, Liu *et al.* [6] designed an algorithm to select the next edge gate to proceed to minimize the number of "S" symbols. In addition, the valence-driven approach is the algorithm of Touma and Gotsman [7]. Its principle is to consider the edge boundary formed by an initial triangle and expand this boundary formed by an initial triangle and expand this boundary by iteratively adding adjacent vertices. Three notably important reviews that summarize the work on mesh compressing were published by Alliez and Gotsman [8], PENG *et al.* [9] and Maglo and Dupont [33]. Generally, the information contained in a mesh is often divided into three categories: geometry, connectivity, and attribute information. The geometry information is the position of each vertex of the mesh in the 3D Cartesian space. Furthermore, the connectivity information mainly describes the incidence relations among the mesh elements. The information associated with scalar or discrete properties is attributed to the useful mesh elements, such as colors, normal, and texture coordinates. Therefore, mesh geometry compression is notably important because it is larger than the connectivity information. The compression of the geometry begins with the quantization of all coordinates. Lee and Park [11] proposed a new scalar quantization approach, which can locate the vertices in four different range sizes because notably few vertices are in the largest range. Meng *et al.* [10] conducted vector quantization, which has experimentally demonstrated its ability to achieve better rate-distortion performance. However, the determination of the quantization can lead to intensive computations. Lavoue *et al.* [12] presented a new framework based on the subvision surface approximation for efficient compression and coding of models; however, it is fit for the model, which has large constant-curvature regions. Moreover, this method is not suitable for natural objects.

C. MESH SEGMENTATION

Mesh segmentation remains notably difficult in computer graphics. Several methodological frameworks have been developed to present this underlying challenge. Aleksey and Funkhouser [13] presented the graph-clustering

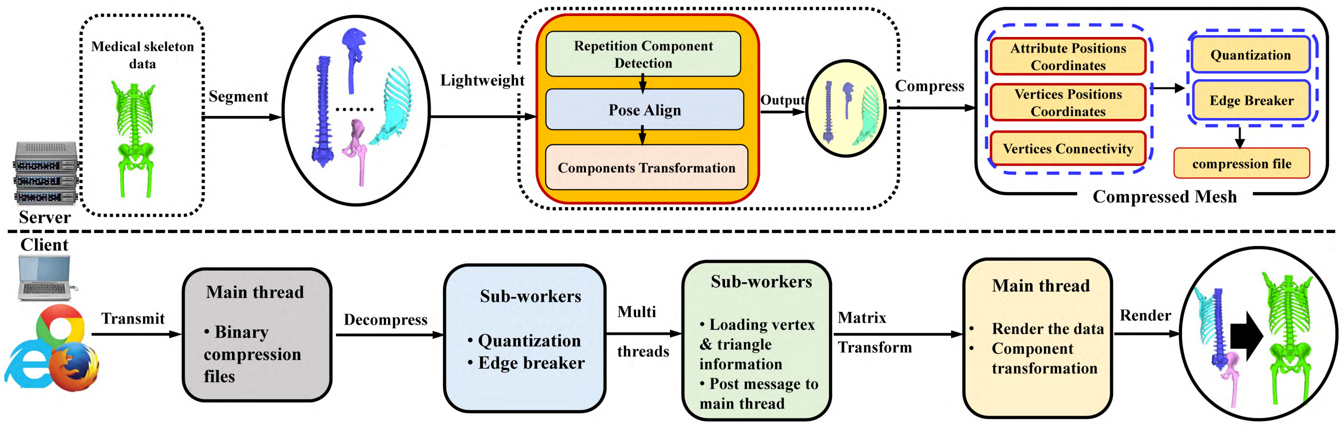


FIGURE 1. Overview of the proposed framework.

method to balance the intra-mesh and inter-mesh segmentations. This method builds the connection by matching the points between rigidly aligned meshes. However, it only handles limited model types for the requirement of global rigid alignment. Xu *et al.* [14] classified the meshes according to their styles and established the part correspondences in each style group. Kraevoy *et al.* [15] created a consistent segmentation by matching the parts generated from an initial segmentation. Huang *et al.* [16] jointly considered the segmentation of individual meshes by linear programming. Nevertheless, the segmentations that are generated by these two methods cannot guarantee the consistency across the entire set even if they are mutually consistent. Sidi *et al.* [17] analyzed the descriptor space via spectral clustering to segment a set of shapes with large variability. Meng *et al.* [18] clustered the primitive patches to generate the initial guess and improve the co-segmentation results by the multi-label optimization. Hu *et al.* [19] generated the segmentation by directly grouping the primitive patches of the meshes and simultaneously obtain their correspondences. This method achieved some success because the patches of the same part were likely to be in one common subspace in the feature space. Liu *et al.* [20] proposed the low-rank representation in semantic mesh segmentation and labeling, but this method has several limitations such as the model style.

III. PROPOSED FRAMEWORK

In this section, we present a framework that includes compression-driven mesh encoding, mesh segmentation, and lightweight for shape. Because it is notably difficult to load the entire model in the browser at once, the model must be split into many individual components. In the paper, we propose a mesh segmentation algorithm to complete the segmentation task. The mesh compression is also notably important for the webpage-based application because mesh compression can significantly improve the mesh transmission efficiency. In this case, we use the Edgebreaker algorithm [4], [5]. The compression task also includes the vertex connectivity information, vertex coordinate information and attribute information. In this paper, the attribute information only refers to the normal coordinate information. Hence, the compression of coordinates mainly indicates the quantization of the vertex coordinates. In addition, the floating-point coordinates are broken into sign, exponent, and mantissa components. The lightweight-for-shape method is used to decrease the model scale. More specially, the repetitive components detection method can find the repetition components in a shape; then, these repetition components are represented by matrix transformation.

In this method, the mesh scale can be reduced, i.e., the lightweight for shape is obtained. The overview of the proposed framework is shown in Figure 1.

IV. FRAMEWORK DESCRIPTION

In this section, we explain the detail of our proposed framework. In particular, our framework can be divided into three parts: mesh compression method; mesh segmentation approach; and lightweight for components of the shape.

A. MESH COMPRESSION

The mesh compression is used to encode the related components, including the connectivity and coordinates. In the connectivity patch, a spanning tree encoding method [4] is used to encode the connectivity. In addition, it defines five types of patch configuration of the Edgebreaker algorithm. This method is shown in Figure 2.

The prediction method of three parallelograms is also used to predict the position of the neighbor triangles according to the encoding region. The prediction details are shown in Figure 3. In particular, the prediction term is denoted as (1).

$$p_{predicted} = \frac{p_{red} + p_{orange} + p_{green}}{3} \tag{1}$$

where the terms p_{red}, p_{orange}, and p_{green} represent the prediction positions of three parallelograms. Nevertheless, in the coordinate encoding patch, we use the delta decoding method

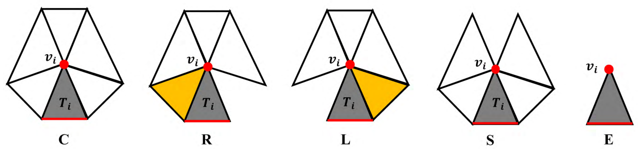


FIGURE 2. Five-patch configuration of the Edgebreaker algorithm. The term v_i is the patch center vertex, and T_i is the current triangle. The current edge is the red edge. In *C*, there is a complete triangle fan around v_i. In *R*, there is missing triangles at the right of the current edge. On the left side, there are missing triangles at the left of the current edge. In *S*, there are missing triangles at other places in addition to the left and right sides of the current edge. In E, there are only the current edge adjacent to current triangle T_i.

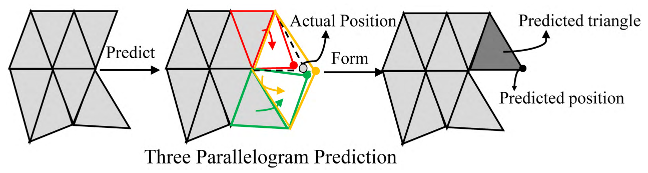


FIGURE 3. Three-parallelogram prediction process. The left patch is actual triangles, and the middle patch illustrates how to predict the new position. The encoded part is shown in light gray.

to compute the difference in position coordinate. Thus, we convert the relative floating-point number into integers based on the bit value; thus, we can finish encoding the attribute positions and vertex positions. The compression process is shown in Figure 4.

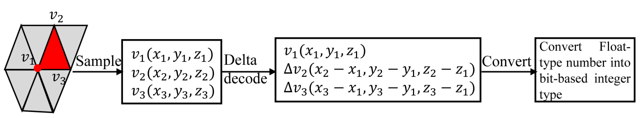

v_2
v_1
v_3
Sample
$v_1(x_1, y_1, z_1)$
$v_2(x_2, y_2, z_2)$
$v_3(x_3, y_3, z_3)$
Delta decode
$v_1(x_1, y_1, z_1)$
$\Delta v_2(x_2 - x_1, y_2 - y_1, z_2 - z_1)$
$\Delta v_3(x_3 - x_1, y_3 - y_1, z_3 - z_1)$
Convert
Convert Float-type number into bit-based integer type

FIGURE 4. Process to compress the floating-point positions and delta decoding.

The entire process to compress the mesh is shown in Figure 5. Specially, we use the stack structure buffer to storage the result. Moreover, the result is written into a compression file.

B. MESH SEGMENTATION

For the dense model, the compression mesh decodes on the browser port, but the decoding result remains large in the web browser, and it remains notably difficult to finish the rendering task. Hence, one of the most efficient and reliable solutions to obtain the lightweight for shape is to perform the algorithms in components of the shape. Zhou *et al.* [34] proposed a new mesh segmentation approach based on Dijkstra algorithm. In this paper, we also use this method to achieve mesh segmentation.

The detail of the proposed method is shown in Figure 6. In addition, we define three new geodesic distances to measure the relation of many triangle: centroid distance (CD), angular distance (AD), and global distance (GD). The GD distance metric is as follows (2).

$$GD(c_1, c_2) = \frac{CD(c_1, c_2) + \phi * AD(c_1, c_2)}{\beta} \tag{2}$$

where the term ϕ is an empirical value, which determines the ratio of the angular distance to the global distance. The experiment shows that a larger value of $\phi \in [0, 500]$ corresponds to a greater ratio of the angular distance in Equation 2 and consequently a stronger model streamline. In this paper, we obtain the value $\phi = 300$. The term β denotes the diagonal length of the bounding box. The process of our proposed mesh segmentation can be described as follows:

step 1: The position of barycenter $\overrightarrow{O}$ of the entire model M_i is calculated, and the nearest triangle $\delta_k^i (k = 0)$ from the barycenter $\overrightarrow{O}$ is obtained, which is the first diffusion source position.

step 2: The centroids of all triangles are computed as the nodes of networks, $K = k_1, k_2, \ldots, k_m (m \in N)$, where m is the number of triangles of the entire model M_i.

step 3: The term δ_k^i is the initial center of the region. Hence, the neighbor triangle set $K_p, p \in [0, m]$ can be obtained.

step 4: The global distance between the triangle δ_k^i and its neighbor is computed. Moreover, the topological networks can be built. In addition, based on the Dijkstra algorithm, the shortest path Ψ_0 from the source point δ_k^i is acquired. If the triangle $k_j, j \in [0, m]$ belongs to path Ψ_0 and triangle k_j belongs to this region, the term δ_k^i is particularly the center position.

step 5: Update the region center, re-calculate the position of the region center, and constantly update the position of the term δ_k^i, where the variable k can be represented as $k \longleftarrow k + 1$.

step 6: Iterate to perform the task of *steps 4 and 5* until the number of steps exceeds the maximal number of iterations or the shortest paths have not existed. Then, new diffusion source $\overrightarrow{O}$ can be built. Then, go to *step 1* to continue to conduct the related operation.

In Figure 7, two types of geodesic distance equations are shown. Specially, the global distance consists of two types of distance. Our proposed segmentation is based on Dijkstra algorithm region diffusion, the detail of which is shown in Figure 8. The algorithm is shown in Algorithm 1.

C. LIGHTWEIGHT FOR SHAPE

In general, there are many repetitive components in a shape, particularly a dense shape. Therefore, the components of the shape must be used to operate the repetition detection. Furthermore, the repetitive components can be represented by a matrix transformation such as translation, scaling, or rotation, so it can significantly decrease the scale of the model. Laixiang *et al.* [2] proposed a pose normalized method to obtain the matrix of component transformation. Specially, this operation includes translation, scaling, and rotation. With these operations, we can restore the repetitive components on the web browser port.

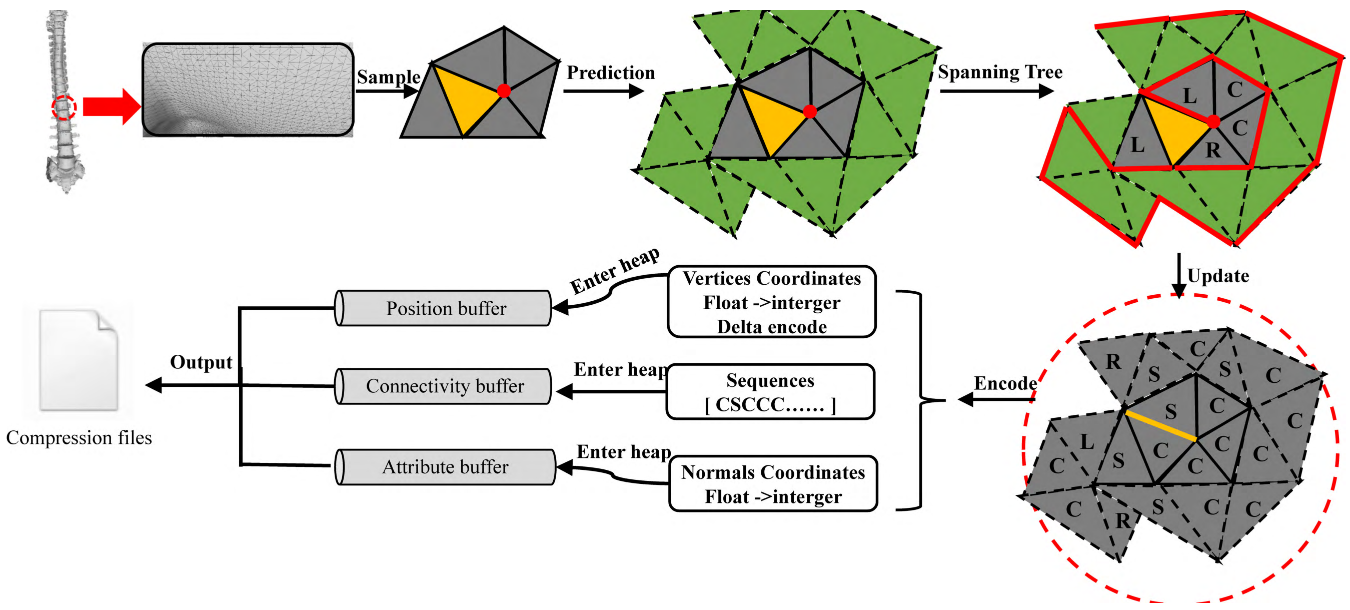


FIGURE 5. Overview of the mesh compression. The orange triangle is the current value, the gray solid zone is the actual value, and the green dotted-line zone is the predicted value. The red line is the encoding path based on the spanning tree, and the orange line is the current edge.

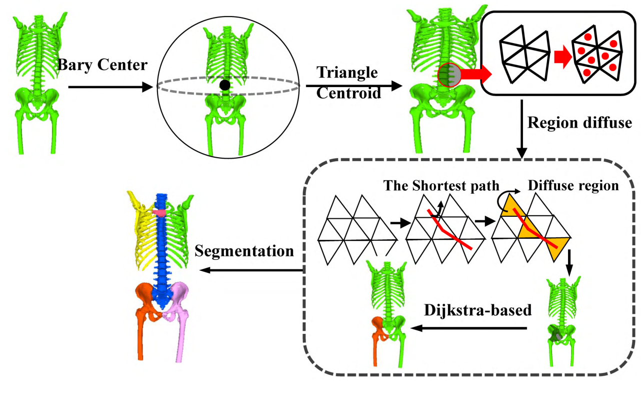


FIGURE 6. Overview of the mesh segmentation based on the Dijkstra algorithm

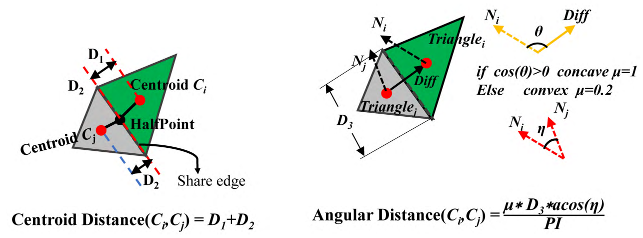


FIGURE 7. Overview of two types of geodesic distance. $CD(c_1, c_2)$ is presented, and the equation of $AD(c_1, c_2)$ is shown on the right side.

Algorithm 1 Dijkstra-Based Mesh Segmentation

```
Input: Shape $S_i$
Output: Component Set $C^i = c_1^i, c_2^i, ..., c_n^i$
Initial: Read Shape,$S_i = K_0, K_1, \ldots, K_m m \in M$
step 1. Compute the barycenter vertex $\overrightarrow{F}$ of Shape $S_i$
step 2. From Shape $S_i$, obtain the nearest triangle $T_k^i$ from vertex $\overrightarrow{F}$
step 3. Obtain all triangle sets $K_p$ that are connected with triangle $T_k^i$
step 4. While($K_p$ is not $NULL$)
step 5.   Region diffusion, obtain triangle $\Theta$
step 6.   $K_p \longleftarrow K_p - \Theta$
step 7.   if ($\Theta$ is $NULL$)
step 8.     Update region center $T_k^i$, obtain new triangles set $\delta$,
              and $\delta \cap K_p = \phi$
step 9.     $K_p \longleftarrow K_p + \delta, k \longleftarrow k + 1$
step 10.    if ($\delta$ is $NULL$)
step 11.      $S_i \longleftarrow S_i - K_p, c_p^i \longleftarrow K_p, p \longleftarrow p + 1$
step 12.      Build new region $K_p, k \longleftarrow 0$
step 13..     Go to step 3
step 14.    end if
step 15.  end if
step 16.end While
step 17. $c_p^i \longleftarrow K_p$
step 18. Output components set $C^i = \{c_1^i, ..., c_p^i\}$
```

However, the repetitive component detection is a key step to obtain the lightweight for shape. We present a voxel-based repetitive-component approach for shape M and its component set $C = \{x|c_x\}, x \in [0, T-1]$.The component set C can be acquired as shown Section 2. The repetitive-component detection includes two main steps.

step 1: $\forall c_i c_j (i, j \in [0, T-1], i \neq j)$ the axis-aligned bounding box (AABBs) of component c_i, c_j are calculated.

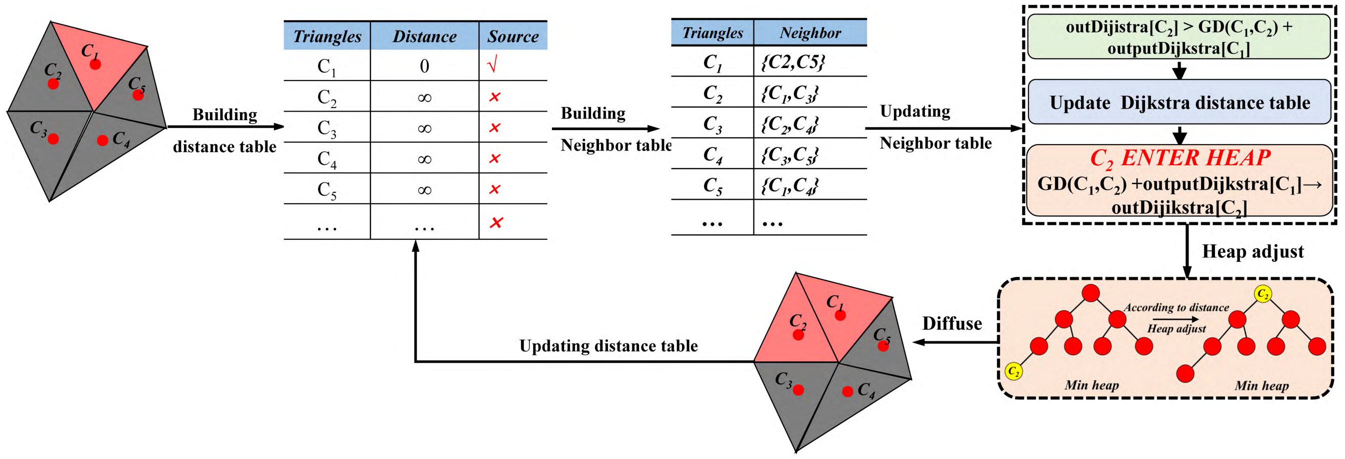


FIGURE 8. Details of Dijkstra-based region diffusion.

If the AABBs of components are same, then two components are possibly similar. Then, the operation will go to *step 2*. Otherwise, the components are not totally the same or similar.

step 2: Voxelization operation is performed for the components c_i, c_j. In this paper, we assume that the number of voxels is 4096. Moreover, we assign a number to each voxel of the component. The components can be denoted as followed. For each voxelization object $V^i = \{m|v^i_m\}$ of component c_i, the volume of each voxel box v^i_m can be represented as Ψ^i_m. Hence, the similarity relationship between the components c_i, c_j can be computed as follows in (3):

$$\Phi(c_i, c_j) = \frac{\sum_{m=0}^{N} h(v^i_m, v^j_m)}{N} \tag{3}$$

where $h(\cdot)$ is a discrimination function and can be defined as follows in (4)

$$h(x, y) = \begin{cases} 0 & if \quad x \neq y \\ 1 & otherwise \end{cases} \tag{4}$$

where the variable N is a constant value that represents the size of the voxels. In this paper, we set it equal 4096. By experiments, we find that when the term $\Phi > 0.9$, the two components can be believed as the repetitive components.

step 3: Matrix transformation: this step includes the translation, scaling, and symmetry operations. The processing of these operations on related repetition components can obtain a transform matrix, by which we can restore the related repetition components in Web3D. The alignment operation can unify the components in the unit space and improve the efficient of transformation. For component c_i, the processing of pose alignment includes translation alignment, scaling alignment, and symmetry alignment.

The translation alignment equation is as follows in (5).

$$m = \frac{1}{E} \sum_{t_i \in C_i} E_i \frac{x_i + y_i + z_i}{3} \tag{5}$$

where m is the reference value, and 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, and Z-coordinates, respectively. In addition, the term E_i is the area of triangle t_i.

Next, the scaling alignment equation is as follows in (6).

$$S = \frac{1}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \tag{6}$$

$$S_x = \frac{1}{|V|} \sum_{v \in V} |v_x| \tag{7}$$

where S is a scaling matrix, and S_x represents the scaling value in X-axis.

In (7), the term $|V|$ is the vertex number in the component C_i. Surely, the variable v_x is the x coordinate of the vertex v. Likewise, we can compute the scaling matrix in the Y-axis and Z-axis.

Finally, the symmetry alignment equation is as follows in (8).

$$F = diag(sign(F_x), sign(F_y), sign(F_z)) \tag{8}$$

$$F_x = \sum_{v \in V} sign(v_x) v_x^2 \tag{9}$$

where F is the diagonal matrix, but a function diag generates a symmetric tridiagonal matrix, i.e., $diag(1, 0, 1) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}$.

In addition, the function $sign(\alpha) = 1$ if $\alpha \geq 0$; otherwise, $sign(\alpha) = 0$.

With (9), we can obtain the symmetry value F_x on the X-axis. Likewise, the symmetry value F_y, F_z can be computed.

V. EXPERIMENTS

A. ENVIRONMENT

The framework presented in this paper has been implemented with C++ and the executed program on PC under Windows 7 OS, Intel core I5-M580 processor, 4 G memory increase.

In this section, all results are obtained in this machine. Html5/WebGL technology has been used to implement visualization on a webpage port.

B. EVALUATION OF COMPRESSION

To evaluate the result of the mesh compression, we use the criteria of Hausdorff distance. Specially, the Hausdorff distance consists of the max Hausdorff distance d_{MAX}, the mean Hausdorff distance d_{MEAN}, and the RMS Hausdorff distance d_{RMS}. Furthermore, we use (10) (11) (12) to perform the error criteria between the original mesh and the processed mesh. In other words, the integrity of mesh is not almost changed.

$$E_{MAX} = \frac{| d_{MAX}(X, Y) - d_{MAX}(X, X) |}{d_{MAX}(X, X)} \tag{10}$$

$$E_{MEAN} = \frac{| d_{MEAN}(X, Y) - d_{MEAN}(X, X) |}{d_{MEAN}(X, X)} \tag{11}$$

$$E_{RMS} = \frac{| d_{RMS}(X, Y) - d_{RMS}(X, X) |}{d_{RMS}(X, X)} \tag{12}$$

where X and Y represent the original mesh and processed mesh, respectively. We also compare the error relation between ours and that of Wang *et al.* [3] to demonstrate that our method better preserves the integrity of the mesh. The result is shown in Figure 9.

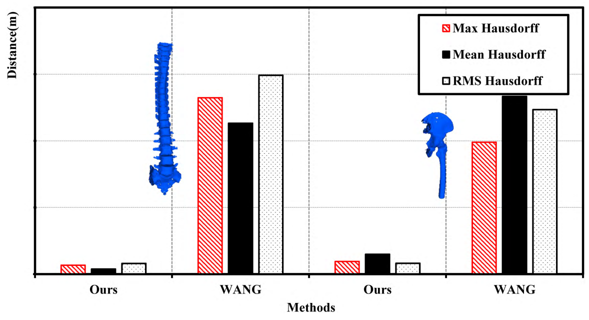


FIGURE 9. **Compared results on Hausdorff distance criterion between our method and WANG's method [3].**

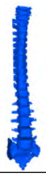
To verify the performance of the compression method over the large skeleton models, we test some skeleton models, finding that the result is superior and that the compressed rate of approach 3%; the decompressed time is notably short in the web browser port. These results totally satisfy our requirements. The detailed information is shown in Table 1.

C. EVALUATION OF SEGMENTATION

In this section, we will evaluate our proposed mesh segmentation based on the Dijkstra approach.

A benchmark for 3D mesh segmentation was proposed by Chen *et al.* [21], which described a benchmark for the evaluation of 3D mesh segmentation algorithms. The benchmark comprises a dataset with 4300 manually generated segmentations for 380 surface meshes of 19 different object categories. Five different indicators were proposed by Chen *et al.*[21],

TABLE 1. **Table 1. Performance of the compression method in a Web3D environment.**

Skeleton	original size	After Compressed size	rate	Decompressed time
Vertebra 	67.1M	2.21M	3.2%	2559ms
Buttocks	19.4M	0.704M	3.6%	1335ms
Sternum	25.7M	0.9M	3.5 %	909ms

which were cut discrepancy (CD), Hamming distance (HD), Rand index (RI), global consistency error (GCE), and local consistency error (LCE). The other state-of-the art approaches include K-means (Shlafman *et al.* [22]), graph cuts (Katz and Tal [23]), random walks (Lai *et al.* [24]), and core extraction (Katz *et al.* [25]). The comparison result in the above indicators will be shown in Figure 10. Hence, it is not difficult to find that our proposed method is more approaching the benchmark than the others methods.

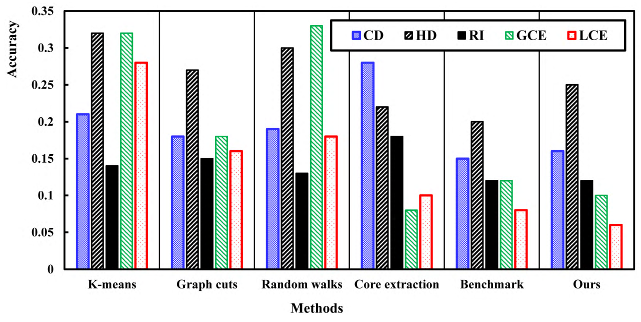


FIGURE 10. **Comparison results in five different indicators.**

D. RESULTS OF LIGHTWEIGHT FOR SHAPE

We perform the related lightweight operation for shape. As mentioned, the lightweight for shape includes two parts: repetitive component detection of the shape and repetitive component transformation. In fact, the repetitive component detection is a key step. Therefore, the results processed by repetitive component detection of some models are shown

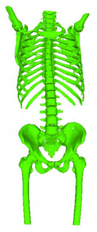
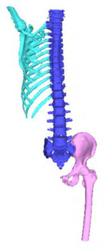
	Original skeleton	Lightweight for skeleton
skeleton		
Size	165M	112M

FIGURE 11. **An example of lightweight for shape.**

in Figure 11. The size of the model will decrease for the lightweight for shape.

E. RESULTS OF THE FRAMEWORK

In this section, our framework is compared with the method of directly loading mesh and Cai *et al.*'s method [32] in terms of the time consumption on the browser. The result is shown in Figure 12, where the time consumption of directly loading mesh is large, and our method is superior. In addition, an example is shown how the mesh displays on web browser using our framework, which is shown in Figure 13.

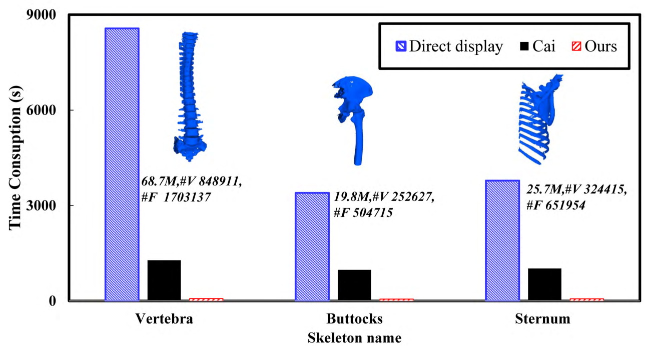


FIGURE 12. **Comparison results between our framework and others.**

VI. CONCLUSIONS

We propose a framework to finish the task of medical skeleton models visualization on web browsers. The framework is based on mesh segmentation and is lightweight for shape and compression. Above all, in the data transmission stage, a component of the mesh is compressed binary bit data, which can realize better transmission on the web browser. In addition, to improve the efficiency of rendering mesh data, we must split the entire dense mesh into many individual components. Then, the repetitive-component detection operation and repetitive-component transformation are conducted to realize the lightweight for shape. In summary, the experiment result shows that our framework is feasible and superior.

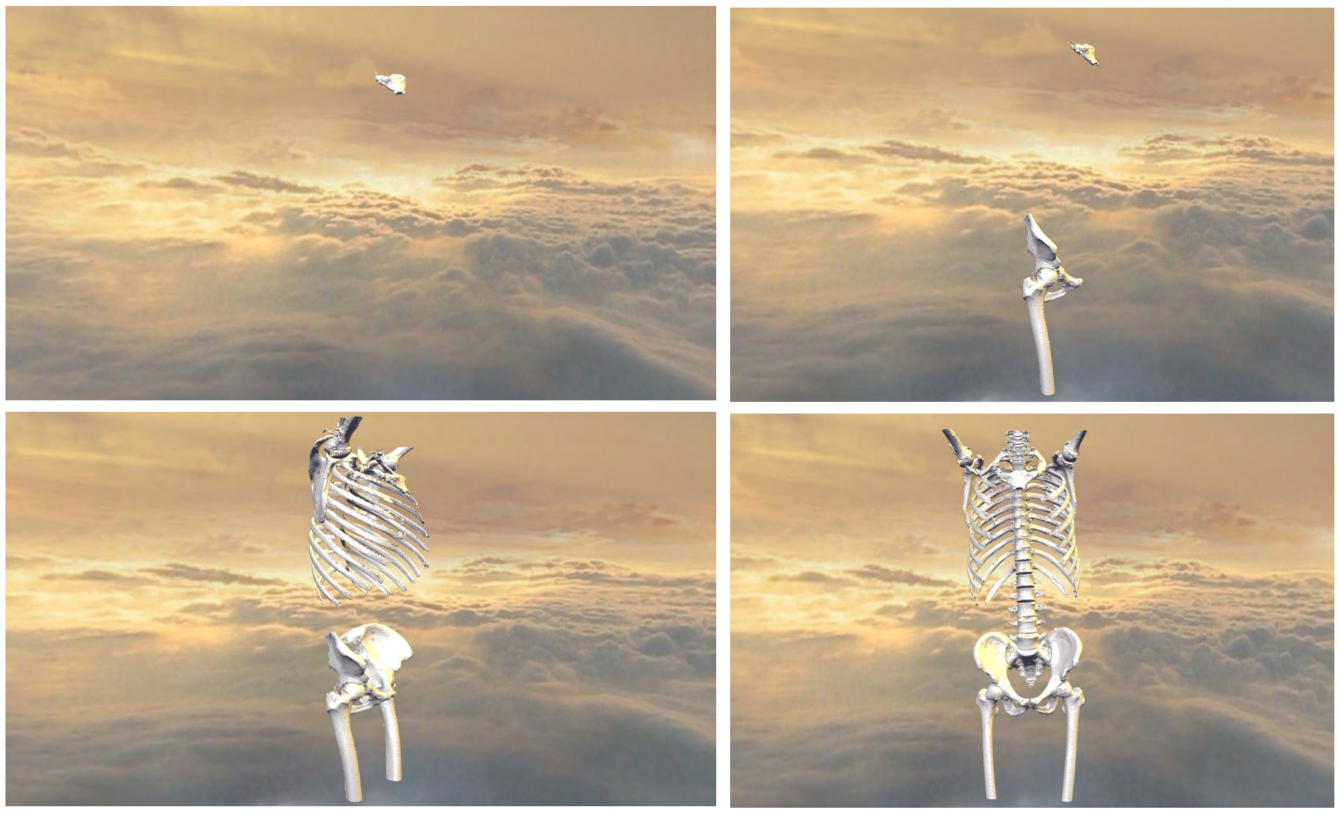


FIGURE 13. **Processed result with our proposed framework on the skeleton model. From top to bottom, from left to right, the skeleton model is progressively displayed on the web browser. In the upper left, the 1*st* components of the skeleton model are first rendered. In the bottom left, the matrix transformation is conducted to render the task symmetry components.**

ACKNOWLEDGMENT

The authors appreciate the comments and suggestions of all anonymous reviewers, whose comments significantly improved this paper.

REFERENCES

[1] S. Zhao, W. T. Ooi, A. Carlier, G. Morin, and V. Charvillat, "3D mesh preview streaming," in *Proc. ACM Multimedia Syst. Conf.*, Barcelona, Spain, 2013, pp. 178–189.

[2] W. Laixiang, X. Ning, and J. Jia, "Fast accessing Web 3D contents using lightweight progressive meshes," *Comput. Animation Virtual Worlds*, vol. 27, no. 5, pp. 466–483, 2016.

[3] Y. Wang *et al.*, "Efficient implementation of adaptive view-dependent mesh simplification," *Proc. SPIE*, vol. 6751, p. 675102, Aug. 2007.

[4] J. Rossignac and A. Szymczak, "Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker," *Comput. Geometry*, vol. 14, nos. 1–3, pp. 119–135, 1999.

[5] S. Gumhold, "Optimizing Markov models with applications to triangular connectivity coding," in *Proc. 16th Annu. ACM-SIAM Symp. Discrete Algorithms*, Vancouver, BC, Canada, 2005, pp. 331–338.

[6] L. Ying, H. Zhongming, D. Mingli, and D. Dagao, "An Edgebreaker & code-mode based connectivity compression for triangular meshes," in *Proc. Int. Conf. Adv. Comput. Control*, Shenyang, China, Mar. 2010, pp. 96–101.

[7] C. Touma and C. Gotsman, "Triangle mesh compression," U.S. Patent 6 167 159 A, Apr. 30, 1998.

[8] P. Alliez and C. Gotsman, "Recent advances in compression of 3D meshes," in *Advances in Multiresolution for Geometric Modelling*. Berlin, Germany: Springer, 2005, pp. 3–26.

[9] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Represent.*, vol. 16, no. 6, pp. 688–733, 2005.

[10] S. Meng, A. Wang, and S. Li, "Compression of 3D triangle meshes based on predictive vector quantization," in *Proc. 3rd Int. Symp. Syst. Control Aeronaut. Astronaut.*, Shenyang, China, Jun. 2010, pp. 1403–1406.

[11] H. Lee and S. Park, "Adaptive vertex chasing for the lossless geometry coding of 3D meshes," in *Proc. Pacific-Rim Conf. Multimedia*. Jeju Island, Korea, 2005, pp. 108–119.

[12] G. Lavoué, F. Dupont, and A. Baskurt, "High rate compression of 3D meshes using a subdivision scheme," in *Proc. 13th Eur. Signal Process. Conf.*, Antalya, Turkey, 2005, pp. 1–4.

[13] A. Golovinskiy and T. Funkhouser, "Technical section: Consistent segmentation of 3D models," *Comput. Graph.*, vol. 33, no. 3, pp. 262–269, 2009.

[14] K. Xu, H. Li, H. Zhang, D. Cohen-Or, Y. Xiong, and Z.-Q. Cheng, "Style-content separation by anisotropic part scales," *ACM Trans. Graph.*, vol. 29, no. 1, 2010, Art. no. 184.

[15] V. Kreavoy, D. Julius, and A. Sheffer, "Model composition from interchangeable components," in *Proc. 15th Pacific Conf. Comput. Graph. Appl.*, Seoul, South Korea, 2007, pp. 129–138.

[16] Q. Huang, V. Koltun, and L. Guibas, "Joint shape segmentation with linear programming," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 125.

[17] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, "Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 126.

[18] M. Meng, J. Xia, J. Luo, and Y. He, "Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization," *Comput.-Aided Des.*, vol. 45, no. 2, pp. 312–320, 2013.

[19] R. Hu, L. Fan, and L. Liu, "Co-segmentation of 3D shapes via subspace clustering," *Comput. Graph. Forum*, vol. 31, no. 5, pp. 1703–1713, 2012.

[20] X. Liu, J. Zhang, R. Liu, B. Li, J. Wang, and J. Cao, "Low-rank 3D mesh segmentation and labeling with structure guiding," *Comput. Graph.*, vol. 46, pp. 99–109, Feb. 2015.

[21] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 73.

[22] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," *Comput. Graph. Forum*, vol. 21, no. 3, pp. 219–228, 2002.

[23] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 954–961, 2003.

[24] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Rapid and effective segmentation of 3D models using random walks," *Comput. Aided Geometric Des.*, vol. 26, no. 6, pp. 665–679, 2009.

[25] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *Vis. Comput.*, vol. 21, no. 8, pp. 649–658, 2005.

[26] N. W. John, "The impact of Web3D technologies on medical education and training," *Comput. Educ.*, vol. 49, no. 1, pp. 19–31, 2007.

[27] N. W. John *et al.*, "MedX3D: Standards enabled desktop medical 3D," *Stud. Health Technol. Inform.*, vol. 132, no. 1, pp. 189–200, 2008.

[28] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz, "Interactive visualization of volumetric data with WebGL in real-time," in *Proc. 16th Int. Conf. 3D Web Technol.*, Paris, France, 2011, pp. 137–146.

[29] A. V. Poliakov *et al.*, "Server-based approach to Web visualization of integrated three-dimensional brain imaging data," *J. Amer. Med. Inform. Assoc.*, vol. 12, no. 2, pp. 140–151, 2005.

[30] S. K. Yoo, J. Key, K. Choi, and J. Jo, "Web-based hybrid visualization of medical images," in *Proc. Int. Conf. Image Video Retr.*, 2005, pp. 376–384.

[31] J. Behr, P. Eschler, Y. Jung, M. Zöllner, "X3DOM: A DOM-based HTML5/X3D integration model," in *Proc. 14th Int. Conf. 3D Web Technol.*, Darmstadt, Germany, 2009, pp. 127–135.

[32] K. Cai, W. Wang, Z. Chen, Q. Chen, and J. Teng, "Exploiting repeated patterns for efficient compression of massive models," in *Proc. 8th Int. Conf. Virtual Reality Continuum Appl. Ind.*, Yokohama, Japan, 2009, pp. 145–150.

[33] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, "3D mesh compression: Survey, comparisons, and emerging trends," *ACM Comput. Surv.*, vol. 47, no. 3, 2015, Art. no. 44.

[34] W. Zhou and J. Jia, "Lightweight Web 3D visualization framework using dijkstra-based mesh segmentation," in *Proc. Int. Conf. Technol. E-Learn. Digit. Entertainment*, Bournemouth, U.K., 2017, pp. 138–151.

WEN ZHOU (M'18) is currently pursuing the Ph.D. degree with the School of Software Engineering, Tongji University. His research interests include Web3D visualization, virtual reality, sketch-based retrieval, and machine learning. He is a Student Member of the Chinese Computer Federation.

JINYUAN JIA received the Ph.D. degree from The Hong Kong University of Science and Technology in 2004. Since 2007, he has been with the School of Software Engineering, Tongji University, Shanghai, China, where he is currently a Professor. His research interests include computer graphics, CAD, geometric modeling, Web3D, mobile VR, game engine, digital entertainment, computer simulation, and peer-to-peer distributed virtual environment. He is an ACM Member, and a Senior Member of the Chinese Computer Federation and the Chinese Steering Committee of Virtual Reality.

XIN SU is currently pursuing the B.S. degree with Tongji University, Shanghai, China. Her research interest includes image reconstruction and 3-D visualization.