IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 20, 2019, accepted July 30, 2019, date of publication August 7, 2019, date of current version August 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933551

An Online Rapid Mesh Segmentation Method Based on an Online Sequential Extreme Learning Machine

FEIYU ZHAO, BUYUN SHENG, XIYAN YIN, HUI WANG, XINCHENG LU, AND YUNCHENG ZHAO

School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China
Hubei Digital Manufacturing Key Laboratory, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Buyun Sheng (shengby@whut.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1101700.

ABSTRACT The existing mesh segmentation methods currently require long training times and have high computational complexity. Consequently, many of these methods cannot meet the rapid requirements of digital geometry processing in the Web environment. This paper proposes an online rapid mesh segmentation method based on an online sequential extreme learning machine (OS-ELM). In the training stage, the OS-ELM is trained by analyzing the mapping relationship between the shape descriptors of the mesh and the Gaussian curvature threshold. We reduce the dimensionality of the shape descriptor vector via principal component analysis (PCA) and extract the Gaussian curvature threshold of the mesh as the sample label using statistics. In the segmentation stage, the Gaussian curvature threshold is quickly extracted to realize the online rapid mesh segmentation via the OS-ELM. Simultaneously, the OS-ELM is updated to realize online incremental learning based on a small number of training samples. Our method is verified using the meshes provided from ShapeNetCore. The experimental results indicate that segmentation results similar to manual segmentation can be rapidly generated online using our method.

INDEX TERMS Online sequential extreme learning machine, rapid mesh segmentation, incremental learning, Gaussian curvature threshold, web environment.

I. INTRODUCTION

Mesh segmentation is a fundamental topic in the field of digital geometry processing [1]. It is widely used in modeling, patching, texture mapping and shape retrieval. In reverse engineering, to transform the reconstructed triangular mesh into a 3D solid model that supports parametric design, the triangular mesh should be segmented, and each segmentation region is matched with some basic solid features, such as extruding and revolving [2]. Therefore, research on accurate mesh segmentation methods is helpful in realizing inverse solid modeling based on 3D scanning [3] and improving the efficiency of reverse product design, which is of great significance in the field of computer-aided design (CAD).

With the continuous development of Web technology, both CAD and digital geometry processing have a tendency to migrate into the Web environment. However, online mesh segmentation requires not only low computational complexity of the algorithm to adapt to the high-concurrency phenomenon but also rapid training and segmentation processes to satisfy the requirements of the Web environment for efficient and real-time digital geometry processing. Therefore, the above situation presents a new challenge for mesh segmentation in the Web environment.

We propose an online rapid mesh segmentation method based on an online sequential extreme learning machine (OS-ELM). The main contributions of this paper are as follows. (1) The fusion of all these components which include Otsu, PCA, OS-ELM and postprocessing algorithms are realized to complete a common task of online rapid mesh segmentation without any manual intervention. (2) A statistical method is used to train the OS-ELM. The dimensionality reduction in the shape descriptor vector is performed, and the Gaussian curvature threshold is extracted as the sample label according

The associate editor coordinating the review of this manuscript and approving it for publication was Shunfeng Cheng.

to Otsu. While improving the reliability of the training sample, the low-dimensionality of the shape descriptor vector ensures the lower computational complexity of the OS-ELM training process. (3) An online mesh segmentation in realized. With the characteristics of incremental learning and random assignment of neuron weights, the OS-ELM can be trained with a small number of training samples and the Gaussian curvature threshold can be quickly extracted. Simultaneously, the accuracy of the Gaussian curvature threshold can be continuously improved through subsequent incremental learning. This algorithm effectively avoids the long training time of machine learning models in existing supervised learning algorithms and is suitable for running in the Web environment. (4) The quality of the mesh segmentation is improved. The postprocessing algorithms are executed such as skeletonizing the feature vertex region, boundary line closure and boundary line optimization, which effectively avoid the unsatisfactory boundaries in the segmentation results generated by the unsupervised learning algorithms.

The remainder of this paper is organized as follows. Section 2 introduces the related work in the field of mesh segmentation and analyses the existing research deficiencies. Section 3 introduces the overview of our method. Section 4 presents the statistically -based OS-ELM training sample set. Section 5 introduces the specific implementation steps of the online rapid mesh segmentation based on the OS-ELM. Section 6 presents the experimental tests using ShapeNetCore and the Princeton Segmentation Benchmark (PSB). Section 7 contains the conclusions.

II. RELATED WORK

Initially, relevant image segmentation theory was applied in the field of mesh segmentation, and manual segmentation was widely used. Chen *et al.* [4] proposed the PSB at SIGGRAPH 2009; this dataset contains more than 4,000 manual segmentation results for 19 types of 380 meshes. Additionally, he presented 4 types of metrics to evaluate different mesh segmentation algorithms compared with manual segmentation results. With the development of machine learning, research regarding mesh segmentation has gradually evolved from manual segmentation to automatic segmentation [5].

A. SUPERVISED LEARNING ALGORITHMS

In the field of mesh segmentation based on a supervised learning algorithm, Kalogerakis *et al.* [6] used the JointBoost algorithm to train a classifier using dozens of shape descriptors to generate the initial segmentation result. A conditional random field (CRF) was also used to obtain the final segmentation result with a smooth boundary. This method can achieve a high accuracy rate using PSB. Then, Kalogerakis *et al.* [7] proposed a 3D shape segmentation architecture with projective convolutional networks, which combines image-based fully convolutional networks (FCNs) and surface-based CRFs to yield coherent segmentations of 3D shapes. Benhabiles *et al.* [8] proposed a fully automatic mesh segmentation algorithm, in which the AdaBoost algorithm was used to train the classifier and the active contour model was used to obtain smooth segmentation results. Xie *et al.* [9] proposed a rapid method for 3D shape segmentation and labeling via an extreme learning machine (ELM); a graph-cut optimization constrained by the super-face boundaries obtained by oversegmentation was executed to generate the final smooth segmentation. Additionally, the real-time sequential learning was realized through an OS-ELM. Then, Xie *et al.* [10] proposed a multiview deep ELM (MVD-ELM) to achieve fast and quality projective feature learning for 3D shapes. The MVD-ELM was extended into a fully convolutional version (referred to as FC-MVD-ELM) to achieve mesh segmentation and labeling. Guo *et al.* [11] presented a robust mesh representation that is learned by nonlinearly combining and hierarchically compressing various geometry features with deep convolutional neural networks (CNNs), which can adapt to various 3D meshes. Yi *et al.* [12] propose a novel active learning method capable of enriching massive geometric datasets with accurate and robust semantic region annotations. Then, Yi *et al.* [13] proposed a method for learning hierarchical shape segmentation and labeling from online repositories, in which the geometric shapes were converted into hierarchically segmented parts with part labels. Qi *et al.* [14] presented a deep learning network named PointNet that provides a unified architecture for part segmentation. Graham *et al.* [15] presented submanifold sparse convolutional networks (SSCNs) for semantic segmentation of spatially-sparse 3D point clouds.

In summary, the abovementioned methods usually use meshes with labeled information to make a training set, and some machine learning algorithms have been used to train the triangular patch classifier. After the initial segmentation result was generated, the CRF, graph cuts, or active contour models (Snakes) were used to generate the final segmentation result with smooth boundaries. Although these supervised learning algorithms can obtain better evaluation scores from PSB, they require several tens of hours to train the machine learning models, which cannot satisfy the requirements of low computational complexity in the Web environment. Although the method proposed by Xie *et al.* [9] performs rapid segmentation compared with other supervised algorithms, the quality of the final segmentation results is highly dependent on that of the oversegmentation. In addition, the intention of ELM training takes only the labeling information in the training data into account, leading to a face classifier, but it does not exploit the boundary information imparted in the data, for which the boundary may easily exhibit obvious zigzags.

B. UNSUPERVISED LEARNING ALGORITHMS

In the field of mesh segmentation based on unsupervised learning algorithms, Luo *et al.* [16] formulated the problem of 3D shapes cosegmentation as a multiview spectral clustering task by cotraining a set of affinity matrices derived from different shape descriptors, which demonstrate robust to

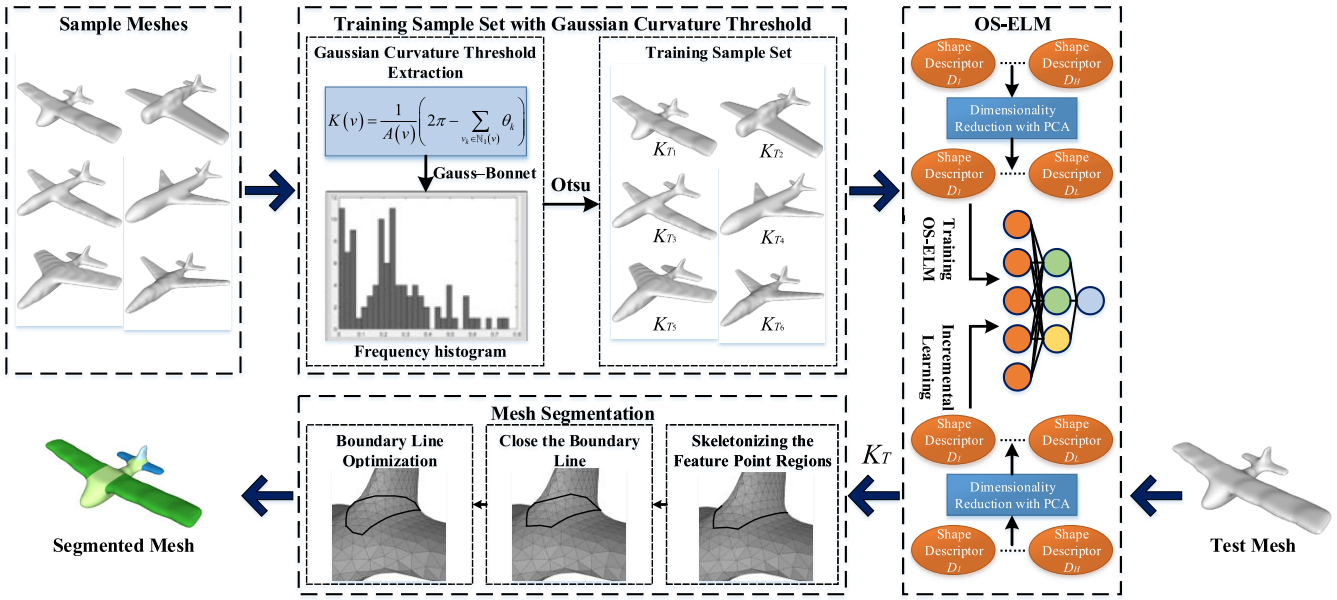


FIGURE 1. An overview of our online rapid mesh segmentation method.

outliers. Shu *et al.* [17] extracted the low-level shape features from the oversegmentation regions, and high-level features are learned, in an unsupervised style, from the low-level ones based on deep learning to realize a clustering of triangular patches in a high-level feature space. Theologou *et al.* [18] presented a fully automatic mesh segmentation scheme using heterogeneous graphs. They developed an unsupervised spectral framework in which local geometry affinities are coupled with surface patch affinities; the segmentation results were comparable to those of best supervised approaches. Wang *et al.* [19] introduced an exemplar-based clustering method based on affinity propagation for 3D shape cosegmentation, which combines the advantages of both affinity-based and model-based cosegmentation methods. Zhang *et al.* [20] took the spectral clustering result as the initialization of the variational refining procedure to realize satisfied 3D shape segmentation. Tong *et al.* [21] transformed the spectral mesh segmentation as the problem of $L0$ gradient minimization, which was highly suitable for detecting sharp features on CAD models.

In summary, the abovementioned methods usually use different clustering methods to segment and label the triangular patches. To give an initial value to the clustering algorithm and accelerate the clustering, the mesh should be oversegmented first, and the feature extraction and clustering should be executed in the oversegmentation region to optimize the segmentation boundaries. Although the faster segmentation speed of such methods satisfies the requirement of lower computational complexity in the Web environment, the final segmentation result is highly dependent on the oversegmentation effect such that the quality of segmentation is far less than that of supervised learning algorithms.

C. OTHER METHODS

In addition to the abovementioned supervised learning and unsupervised learning methods, scholars have proposed some other methods. Fayolle and Pasko [22] realized the segmentation of point clouds by fitting the template primitives from a specified set of candidate template primitives and extracting the points corresponding to the best-fitting primitive in each iteration. Fan *et al.* [23] presented a self-adaptive segmentation method for a point cloud, which performed an automatic selection of seed points according to extracted features and segmentation of the points using an improved region-growing algorithm. The segmentation accuracy rate was as high as 96%. Mejia *et al.* [24] presented an implementation of a hybrid geometry / topology mesh segmentation algorithm, which yields not only a parameterizable mesh but also a functional partition of scanned mechanical workpieces without resorting to oversegmentation. The algorithm allows automatic processing of 3D meshes from scanned workpieces, improving the reverse engineering workflow. Although the abovementioned methods avoid the high computational complexity caused by the machine learning algorithms, they are only effective for a certain type of mesh, which cannot realize satisfied cosegmentation results.

III. OVERVIEW

We present an online rapid mesh segmentation method based on an OS-ELM. First, the statistical-based training sample set of the OS-ELM is collected. Subsequently, the initialization of the OS-ELM is finished. Consequently, the online incremental learning of the OS-ELM is performed based on the new input mesh. Finally, the mesh segmentation is realized using the Gaussian curvature threshold extracted from the OS-ELM. A flow-chart of our method is shown in Fig. 1.

A. COLLECTION OF THE TRAINING SAMPLE SET

The first step of our method is to solve the Gaussian curvature of each vertex on the sample mesh based on the Gauss–Bonnet theorem and draw the frequency histogram to extract the Gaussian curvature threshold using Otsu. The operations are uniformly performed on all meshes in the sample set such that we collect the training sample set of the OS-ELM containing the labels of the Gaussian curvature threshold.

B. INITIALIZATION OF THE OS-ELM

Then we construct the original shape descriptor vector $[D_1, \ldots, D_H]$ of a sample mesh and perform dimensionality reduction on the compound shape descriptor vector $[D_1, \ldots, D_L]$ using PCA ($L < H$). We initialize the OS-ELM using a small sample of meshes that contains their compound shape descriptor vectors and Gaussian curvature thresholds.

C. ONLINE INCREMENTAL LEARNING OF THE OS-ELM

In this stage, we also construct the original shape descriptor vector $[D_1, \ldots, D_H]$ of a test mesh and perform dimensionality reduction on the compound shape descriptor vector $[D_1, \ldots, D_L]$ using PCA as the input of the OS-ELM. Thus, the Gaussian curvature threshold K_T of this test mesh is extracted using the OS-ELM. Meanwhile, the output weight vector of the OS-ELM is updated to realize online incremental learning.

D. MESH SEGMENTATION AND OPTIMIZATION

Finally, the vertices with Gaussian curvature greater than K_T are marked as feature vertices, and accurate boundary lines are generated through the skeletonizing of feature vertex regions. Then, the boundary lines are closed based on the ray method, and the active contour model (Snakes) is used to optimize the boundary lines.

IV. STATISTICALLY-BASED OS-ELM TRAINING SAMPLE SET COLLECTION

A. GAUSSIAN CURVATURE ESTIMATION

In differential geometry, the Gaussian curvature K of a vertex on a surface is the product of the principal curvatures K_1 and K_2 [25]; that is, $K = K_1 K_2$. The Gaussian curvature is an intrinsic measure of a curvature that reflects the degree of local curvature of the surface [26]. According to the analysis of the manual segmentation results in the PSB, the values of the Gaussian curvature of the boundary vertices are often significantly higher than the nonboundary vertices [27]. We define the Gaussian curvature threshold K_T as a key metric to distinguish the boundary vertices from the nonboundary vertices and consider the vertices with a Gaussian curvature greater than K_T as the feature vertices for distributing around the accurate boundary lines.

Let the three vertices of triangular patch T be v_i, v_j and v_{j+1}. As shown in Fig. 2(a) and Fig. 2(b), the Gaussian curvature estimation of vertex v_i depends on the coordinate values of v_i and other vertices in the one-dimensional neighborhood of v_i. Meanwhile, let the angle between $v_i v_j$ and $v_j v_{j+1}$ be φ_i, and the angle between $v_i v_{j+1}$ and $v_j v_{j+1}$ be ψ_i. As is known from the Gauss–Bonnet theorem [28],

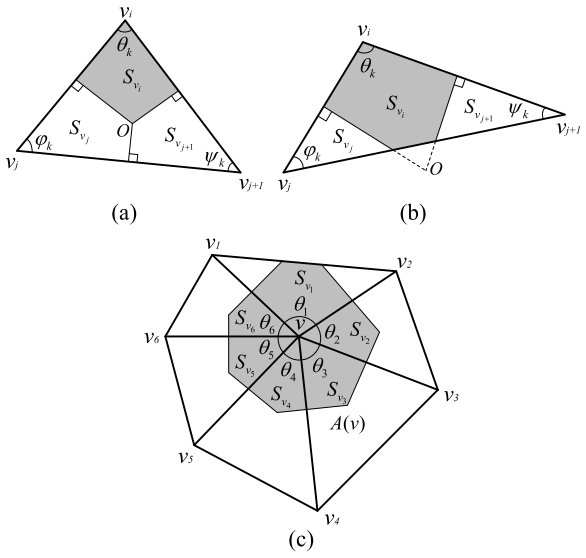


FIGURE 2. Gaussian curvature estimation of a vertex on the mesh. (a) Triangular patch with three acute angles. (b) Triangular patch with one obtuse angle. (c) Gaussian curvature estimation of v through six triangular patches.

$$K(v) = \frac{1}{A(v)} \left(2\pi - \sum_{v_k \in \mathbb{N}_1(v)} \theta_k \right) \tag{1}$$

where $K(v)$ represents the Gaussian curvature of vertex v, $A(v)$ represents the areas of the region corresponding to the neighborhood triangle patches of vertex v, and θ_k represents the angle between v and the k-th triangle patch in the neighborhood of v. $A(v)$ is formulated as follows:

$$A(v) = \sum_i S_{vi} \tag{2}$$

where S_{vi} represents the area of the region segmented by the Voronoi diagram in the neighborhood of v_i (which are marked in gray in Fig. 2) [29]. S_{vi} is formulated as follows:

$$S_{vi} = \begin{cases} \frac{1}{8}\left(\left\| v_i v_{j+1} \right\|^2 \cot \varphi_k + \left\| v_i v_j \right\|^2 \cot \psi_k \right), \\ \qquad 0 < \theta_k \le \frac{\pi}{2} \\ \frac{1}{2} \left\| v_i v_j \right\| \left\| v_i v_{j+1} \right\| \sin \theta_k \\ -\frac{1}{8}\left(\left\| v_i v_j \right\|^2 \tan \varphi_k + \left\| v_i v_{j+1} \right\|^2 \tan \psi_k \right), \\ \qquad \frac{\pi}{2} < \theta_k < \pi \end{cases} \tag{3}$$

As shown in Fig. 2(c), the Gaussian curvature $K(v)$ of vertex v can be estimated as follows:

$$\begin{aligned} K(v) &= \frac{2\pi - \sum_{v_k \in \mathbb{N}_1(v)} \theta_k}{A(v)} \\ &= \frac{2\pi - (\theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 + \theta_6)}{S_{v1} + S_{v2} + S_{v3} + S_{v4} + S_{v5} + S_{v6}} \end{aligned} \tag{4}$$

B. GAUSSIAN CURVATURE THRESHOLD EXTRACTION BASED ON FREQUENCY HISTOGRAM ANALYSIS

To make the values of the Gaussian curvature easy to process as the frequency histogram, they should be linearly normalized [30]:

$$K'(v) = \frac{K(v) - \min(K(v))}{\max(K(v)) - \min(K(v))} \tag{5}$$

where $K'(v) \in [0, 1]$. Taking the number of vertices V as the ordinate and $K'(v)$ as the abscissa, the transverse axis is equally divided into n parts. The number of vertices in each part is calculated, and the histogram should be smoothed [31]. The Gaussian curvature frequency histogram is binarized to obtain a Gaussian curvature threshold using Otsu [32]. First, the hypothesis threshold $K'_t(v)$ for the t-th equal partition is formulated as follows:

$$K'_t(v) = \frac{t}{n} \max\left(K'(v)\right) \tag{6}$$

where $\max(K'(v))$ represents the maximum value of the Gaussian curvature of all vertices after normalization. Then, the algorithm calculates the number of vertices of 2 adjacent partitions V_{t-1} and V_{t+1}. Therefore, the expectations of Gaussian curvature $E(V_{t-1})$ and $E(V_{t+1})$ are formulated as follows:

$$E(V_t) = \frac{\max\left(K'(v)\right)}{nV_t} \tag{7}$$

where V_t represents the number of vertices in the t-th partition. While the hypothesis threshold is $K'_t(v)$, the variance $D(V_t)$ is formulated as follows:

$$D(V_t) = V_{t-1} \times V_{t+1} \times [E(V_{t-1}) - E(V_{t+1})]^2 \tag{8}$$

Based on the distribution of $D(V_t)$, the hypothesis threshold corresponding to $\max(D(V_t))$ is selected as the Gaussian curvature threshold K_T. Thus, the OS-ELM training sample set can be collected via the Gaussian curvature threshold estimation for a plurality of meshes with the same or similar appearance:

$$S = \left\{M_1\left(K_T^1\right), M_2\left(K_T^2\right), \cdots, M_w\left(K_T^w\right), \cdots, M_W\left(K_T^W\right)\right\} \tag{9}$$

where W represents the number of samples, $1 \le w \le W$, and K_T^w represents the Gaussian curvature threshold of the w-th sample mesh.

V. ONLINE RAPID MESH SEGMENTATION BASED ON THE OS-ELM

A. PCA-BASED DIMENSIONALITY REDUCTION IN SHAPE DESCRIPTORS

The shape descriptor is an abstract and concise representation of the 3D shape, which reflects the geometric feature information of the mesh. The shape descriptor has the characteristics of invariance of mesh rotation and translation and insensitivity of mesh deformation [33]. The input vector contains a variety of shape descriptors, not only increasing the complexity of OS-ELM but also increasing the computational complexity of the algorithm, which cannot satisfy the requirement of low computational complexity in the Web environment. Therefore, we reduce the dimensionality of various shape descriptors using PCA. Let the number of shape descriptors be H in the sample set S; the sample matrix is formulated as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1w} & \cdots & x_{1W} \\ x_{21} & x_{22} & \cdots & x_{2w} & \cdots & x_{2W} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{h1} & x_{h2} & \cdots & x_{hw} & \cdots & x_{hW} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{H1} & x_{H2} & \cdots & x_{Hw} & \cdots & x_{HW} \end{bmatrix} \tag{10}$$

where $\mathbf{x}_w = [x_{1w}, x_{2w}, \cdots, x_{hw}, x_{Hw}]^T$ represents the input vector that contains H kinds of shape descriptors of the w-th sample mesh, $1 \le w \le W$, $1 \le h \le H$. The mean of each row μ is calculated and zero-averaged to generate a matrix $\overline{X}$:

$$\overline{X} = [x_{hw} - \mu] = \left[x_{hw} - \frac{1}{W}\sum_{w=1}^{W} x_{hw}\right] \tag{11}$$

The covariance matrix C is constructed, and the eigenvalues λ_h and eigenvectors $\mathbf{u}_h$ of C are solved using singular value decomposition (SVD) [34]:

$$C = \frac{1}{W}\overline{X}\overline{X}^T \tag{12}$$

The eigenvectors $\mathbf{u}_h$ are arranged in a row from top to bottom according to the corresponding eigenvalues λ_h, so that a new matrix is generated. The first L rows ($L < H$) are used to form another new matrix P, and the dimensionality-reduced matrix Y can be generated through $Y = PX$, which is formulated as follows:

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1W} \\ y_{21} & y_{22} & \cdots & y_{2W} \\ \vdots & \vdots & \ddots & \vdots \\ y_{L1} & y_{L2} & \cdots & y_{LW} \end{bmatrix} \tag{13}$$

B. INITIALIZATION OF THE OS-ELM BASED ON A SMALL SAMPLE OF MESHES

The OS-ELM is a single hidden layer feedforward neural network suitable for the Web environment. Its input weight is subject to random assignment by a certain distribution function, and the output weight is directly calculated via the least squares method. Thus, its training and recognition processes are rapid. The basic structure of OS-ELM is shown in Fig. 3.

OS-ELM supports incremental learning for continuously increasing sample meshes. That is, the initialization relies on only a small number of meshes. After being deployed on a Web server, it supports incremental learning of continuously increasing sample meshes and updates the output weight synchronously. Therefore, the training speed is dramatically

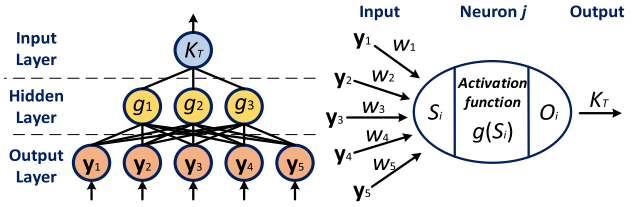


FIGURE 3. The basic structure of OS-ELM and a neuron in the hidden layer.

improved, which is suitable for performing machine learning in a high-concurrency environment.

Let the Gaussian curvature threshold vector be $\mathbf{t}_{K_T} = [K_T^1, K_T^2, \cdots, K_T^W]^T$. As is shown in Ref. [35], for an OS-ELM with R hidden layer nodes and an activation function of g($\mathbf{y}$), the output O_w is formulated as follows:

$$O_w = \sum_{i=1}^{R} \beta_i g_i(S_i) = \sum_{i=1}^{R} \beta_i g(W_i \cdot \mathbf{y}_w + b_i) \tag{14}$$

where $W_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ represents the input weight vector, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots \beta_{im}]^T$ represents the output weight vector, and b_i represents the bias of the i-th hidden layer node. We select the Sigmoid function as the activation function [36]. The training goal of OS-ELM is to approximate W sample meshes with minimal error, which can be formulated as

$$\sum_{w=1}^{W} \left\| O_w - K_T^w \right\| = 0 \tag{15}$$

That is, the existence of β_i, W_i and b_i make the following equation true:

$$\sum_{i=1}^{R} \beta_i g(W_i \cdot \mathbf{y}_w + b_i) = K_T^w \tag{16}$$

We make a matrix form of equation (16) as $H\beta = \mathbf{t}_{K_T}$. H represents the output matrix of the hidden layer nodes, and β represents the output weight vector. To train this model, we want to find W_i, b_i and β_i that satisfy the following equation:

$$\left\| H(W_i, b_i)\beta_i - \mathbf{t}_{K_T} \right\| = \min_{W,b,\beta} \left\| H(W_i, b_i)\beta_i - \mathbf{t}_{K_T} \right\| \tag{17}$$

Equation (17) is equivalent to minimizing the loss function as follows:

$$E = \sum_{w=1}^{W} \left(\sum_{i=1}^{R} \beta_i g(W_i \cdot \mathbf{y}_w + b_i) - K_T^w \right)^2 \tag{18}$$

Therefore, the process of solving the OS-ELM is transformed into solving the linear system of $H\beta = \mathbf{t}_{K_T}$. The output weight vector β can be solved using the following equation:

$$\widehat{\beta} = H^+ \mathbf{t}_{K_T} \tag{19}$$

where H^+ represents the Moore-Penrose generalized inverse matrix of H [37].

C. ONLINE INCREMENTAL LEARNING OF THE OS-ELM BASED ON NEW INPUT MESHES

The initialized OS-ELM is deployed in the Web server. Assume that there are W' new input meshes uploaded to the Web server at the same time. It is known from the classic ELM theory that β' can be solved using the following equation:

$$\min_{W',\beta',b'} \left\| \begin{bmatrix} H \\ H' \end{bmatrix} \beta' = \begin{bmatrix} \mathbf{t}_{K_T} \\ \mathbf{t}'_{K_T} \end{bmatrix} \right\| \tag{20}$$

where H' represents the output matrix from the hidden layer nodes and is generated from the new input meshes. $\mathbf{t}'_{K_T}$ represents the Gaussian curvature threshold vector of the new input meshes. As is known from Ref. [38], when $W' > 1$, β' should be satisfied as follows:

$$\begin{cases} \beta' = \beta + P'H'^T \left(\mathbf{t}'_{K_T} - H'\beta \right) \\ P' = P - PH'^T \left(I + H'PH'^T \right) H'P \end{cases} \tag{21}$$

However, when $W' = 1$, H' is transformed into a vector $\mathbf{h}'$ such that β' should be satisfied as follows:

$$\begin{cases} \beta' = \beta + P'\mathbf{h}' \left(\mathbf{t}'^T_{K_T} - \mathbf{h}'\beta \right) \\ P' = P - \dfrac{P\mathbf{h}'\mathbf{h}'^T P}{1 + \mathbf{h}'^T P\mathbf{h}'} \end{cases} \tag{22}$$

where $P = \left(H^T H\right)^{-1}$. As shown in equations (21) and (22), during the initialization process, the OS-ELM is trained with a small number of sample meshes to obtain the initial output weight vector β. Thus, it can be deployed in the Web server. During the online incremental learning process, the output weight vector β' of the OS-ELM is updated through the newly input meshes, which effectively improves the training speed.

D. MESH SEGMENTATION STAGE

1) SKELETONIZING FEATURE VERTEX REGIONS

The mesh M to be segmented is input into the trained OS-ELM to obtain its Gaussian curvature threshold K_T, and the vertices whose Gaussian curvatures are greater than K_T should be marked as feature vertices. The feature vertices tend to present regional distributions near the segmentation boundaries. To extract accurate boundary lines, some fake feature vertices should be removed; that is, the skeletonizing of feature vertex regions should be executed first. We present a feature vertex region $G = \sum_{i=1}^{30} v_i$, which is shown in Fig. 4(a), as an example to describe the process of skeletonizing as follows:

Step 1 Remove the Obvious Outliers in G: Calculate the average Euclidean distance from vi to the remaining feature vertices, which is formulated as $\overline{d_i} = \frac{1}{30} \sum_{k=1,k\neq i}^{30} \|v_i - v_k\|$. We consider that $\overline{d_i}$ obeys Gaussian distribution and calculate the mean μ and variance σ of $\overline{d_i}$. As shown in Fig. 4(b), we treat the feature vertices distributed outside $\pm 3\sigma$ as the outliers (the outliers are v_3, v_6, v_9, v_{14} and v_{18}) and remove them from G.

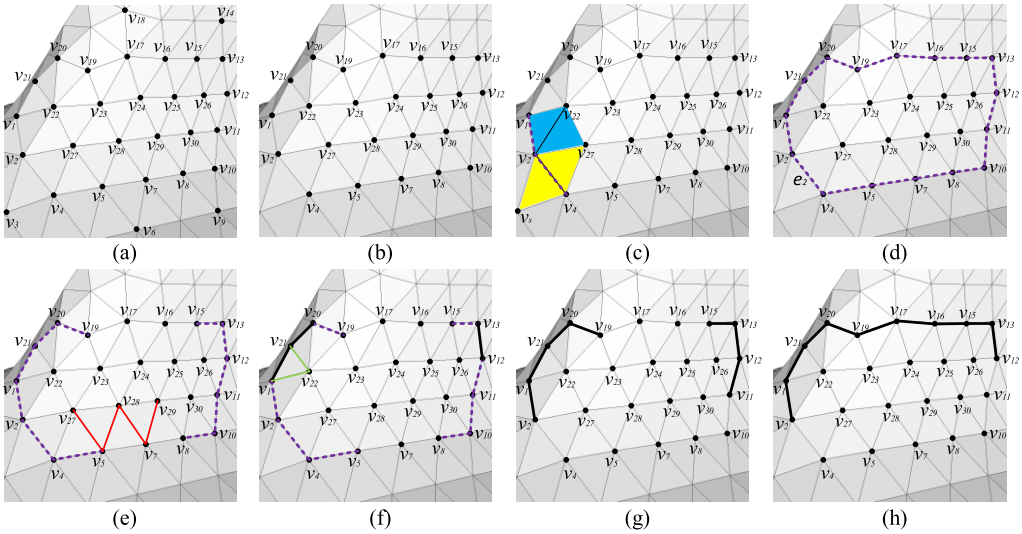


FIGURE 4. Skeletonizing the feature vertex region. (a) Feature vertex region *G*. (b) Remove the obvious outliers in *G*. (c) Judge the boundary edge. (d) Generate the boundary edge set of *G*. (e) Extract the suspected skeleton edges. (f), (g) Extract the skeleton edges. (h) Extract the approximate boundary.

Step 2 Extract the Boundary Edges: Traverse all the edges in G. Consider an edge that belongs to two adjacent triangles. If any of the other four edges of the two adjacent triangles does not belong to G, then it is a boundary edge; otherwise, it is an inner edge. As shown in Fig. 4(c), edge $\overline{v_2v_4}$ belongs to two triangles, $\Delta v_2v_4v_{27}$ and $\Delta v_2v_4v_s$. The edges $\overline{v_2v_s}$ and $\overline{v_4v_s}$, which belong to $\Delta v_2v_4v_s$, are not the inner edges or the boundary edges, so $\overline{v_2v_4}$ is boundary edge. Meanwhile, edge $\overline{v_2v_{22}}$ belongs to two triangles $\Delta v_1v_2v_{22}$ and $\Delta v_2v_{22}v_{27}$. $\overline{v_1v_2}$ is the boundary edge of G, and $\overline{v_1v_{22}}$, $\overline{v_2v_{27}}$, $\overline{v_{22}v_{27}}$ are all the inner edges of G; thus, $\overline{v_2v_{22}}$ is an inner edge. We mark all of the boundary edges with purple bold dotted lines in Fig. 4(d).

Step 3 Extract the Suspected Skeleton Edges: Traverse all the boundary edges extracted from ***Step 2***. If two endpoints of a boundary edge belong to two or more inner edges in G, the boundary edge is removed, and the remaining boundary edges are defined as suspected skeleton edges. We take the boundary edge $\overline{v_5v_7}$ as an example, which is shown in Fig. 4(e). The endpoint v_5 belongs to the inner edges $\overline{v_5v_{27}}$ and $\overline{v_5v_{28}}$, whereas the endpoint v_7 belongs to the inner edges $\overline{v_7v_{28}}$ and $\overline{v_7v_{29}}$ such that $\overline{v_5v_7}$ is not a skeleton edge, which should be removed. Meanwhile, the boundary edges $\overline{v_7v_8}$, $\overline{v_{15}v_{16}}$, $\overline{v_{16}v_{17}}$ and $\overline{v_{17}v_{19}}$ should also be removed. The remaining boundary edges are suspected skeleton edges, which need to be further judged in ***Step 4***.

Step 4 Extract the Skeleton Edges: Traverse all the suspected skeleton edges extracted from ***Step 3***. If both endpoints of a suspected skeleton edge belong to only one inner edge in G or do not belong to any inner edge in G, then it must be a skeleton edge, and the suspected skeleton edges to which it is connected must also be skeleton edges. We take the suspected boundary edge $\overline{v_1v_{21}}$ as an example, which is shown in Fig. 4(f). The endpoints v_1 and v_{21} belong only to the inner edges $\overline{v_1v_{22}}$ and $\overline{v_{21}v_{22}}$, respectively; thus, $\overline{v_1v_{21}}$ is the skeleton edge, and $\overline{v_1v_2}$, to which it is connected, must also be skeleton edge, which are indicated by black bold lines. Meanwhile, as shown in Fig. 4(g), $\overline{v_{20}v_{21}}$ and $\overline{v_{12}v_{13}}$ are also the skeleton edges, and $\overline{v_{19}v_{20}}$, $\overline{v_{11}v_{12}}$, and $\overline{v_{13}v_{15}}$, to which they are connected must also be skeleton edges.

Step 5 Extract the Approximate Segmentation Boundary Line: All the extracted skeleton edges are connected to generate the approximate boundary line along the boundary edges of G using the shortest path. As shown in Fig. 4(h), from v_{15} to v_{19} along the boundary edges of G, only two points, v_{16} and v_{17}, must be passed, less than the five points–v_4, v_5, v_7, v_8, and v_{10}–that must be passed through v_2 to v_{11}. Therefore, the discrete skeleton edges are connected to generate the approximate segmentation boundary line along v_{16} and v_{17}.

2) BOUNDARY LINE CLOSURE

The approximate segmentation boundary line extracted using the skeletonizing method often does not close, which makes it impossible to ensure the closedness of the segmented area. We design a ray-based boundary line closure method to realize the boundary line closed by capturing the intersection of the ray and the triangular patch to learn the internal shape of the mesh.

a: CONSTRUCT THE RAY

To ensure that the ray is emitted from inside the mesh, the centroid of all vertices v_g on the approximate boundary line is selected as the starting point of the ray. v_g is formulated as follows:

$$v_g = \frac{1}{q}\sum_{i=1}^{q} v_i \tag{23}$$

where v_i represents the vertices on the approximate boundary line and q represents the number of vertices. Unit vectors $\mathbf{e}_s$

and $\mathbf{e}_e$ are created by connecting v_g, v_2 and v_g, v_{12}, respectively. Let the angle between $\mathbf{e}_s$ and $\mathbf{e}_e$ be $\Phi = \arccos(\mathbf{e}_s \cdot \mathbf{e}_e)$. If a series of rays with an angle of φ should be generated from v_g ($0 < \varphi < \Phi$), Slerp [39] should be used from $\mathbf{e}_s$ to $\mathbf{e}_e$, which is formulated as follows:

$$q_t = q_s \frac{\sin(1-t)\,\Phi}{\sin\Phi} + q_e \frac{\sin t\Phi}{\sin\Phi} \tag{24}$$

where q_s and q_e represent the pure quaternions of $\mathbf{e}_s$ and $\mathbf{e}_e$, respectively; q_t represents the pure quaternion of $\mathbf{e}_t$, which is to be solved as the unit vector of the ray; t represents the interpolation parameter; and $t = \varphi/\Phi$. q_t can be transformed into $\mathbf{e}_t$ such that the ray is started along the direction of $\mathbf{e}_t$.

b: COMPUTE THE INTERSECTION

To quickly calculate the intersection of the ray and the triangular patch, the *octree* spatial index of the mesh is established, and vector decomposition is used to determine whether ray l intersects the triangular patch.

We take the intersection P_c as an example such that the ray l constructed from v_g to P_c can be formulated as follows:

$$p = v_g + \eta \mathbf{e_c} \tag{25}$$

where $\mathbf{e_c}$ represents the unit vector constructed from v_g to P_c, which can be calculated using equation (24), and $\eta \geq 0$. P_c is located inside the triangular patch T (v_i, v_j, v_k), which is formulated as follows:

$$p = v_i + \mu\left(v_j - v_i\right) + \xi\left(v_k - v_i\right) \tag{26}$$

where $(\mu, \xi) \in [0, 1] \times [0, 1]$, such that P_c can be solved via the following equation:

$$v_g + \eta \mathbf{e_c} = v_i + \mu\left(v_j - v_i\right) + \xi\left(v_k - v_i\right) \tag{27}$$

We rewrite equation (27) in matrix form:

$$\left[-\mathbf{e_c}\ v_j - v_i\ v_k - v_i\right] \begin{bmatrix} \eta \\ \mu \\ \xi \end{bmatrix} = v_g - v_i \tag{28}$$

According to Kramer's law, η, μ, and ξ are computed when the ray l intersects the triangular patch T, which is formulated as follows:

$$\begin{bmatrix} \eta \\ \mu \\ \xi \end{bmatrix} = \frac{1}{\mathbf{e_c} \times (v_k - v_i) \cdot \left(v_j - v_i\right)} \times \begin{bmatrix} \left(v_g - v_i\right) \times \left(v_j - v_i\right) \cdot (v_k - v_i) \\ \mathbf{e_c} \times (v_k - v_i) \cdot \left(v_g - v_i\right) \\ \left(v_g - v_i\right) \times \left(v_j - v_i\right) \cdot \mathbf{e_c} \end{bmatrix} \tag{29}$$

The ray l intersects the triangular patch T at P_c if and only if $\eta \in [0, 1]$ such that P_c is computed by substituting η into equation (25). To avoid P_c lying inside the triangular patch T, which may lead to mesh subdivision and increase the computational complexity, the position of P_c should be corrected. That is, the Euclidean distance between P_c and the three vertices of the triangular patch T are determined, and the vertex with the smallest distance is selected as the corrected intersection v_c, which is formulated as follows:

$$\|P_c - v_c\| = \min\left\{\|P_c - v_i\|, \left\|P_c - v_j\right\|, \|P_c - v_k\|\right\} \tag{30}$$

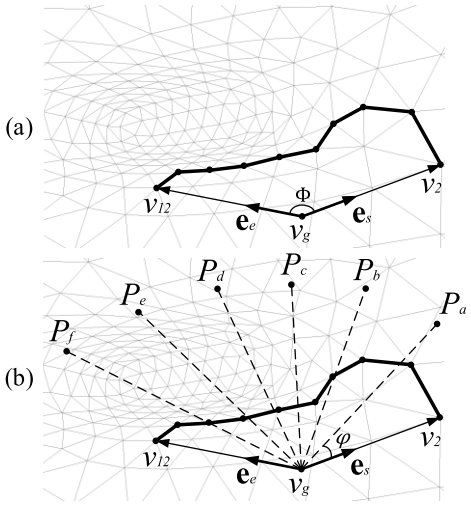


FIGURE 5. Construction of the ray. (a) Calculate the centroid of all vertices on the approximate boundary line. (b) Intersections of the ray and the triangular patches.

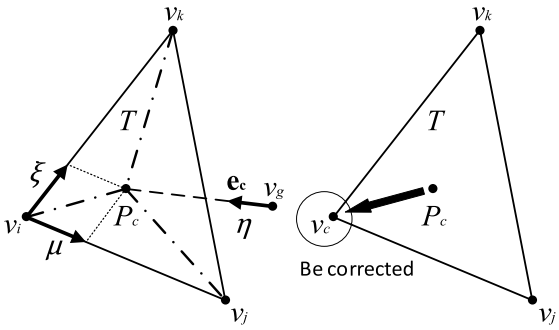


FIGURE 6. Computation of the intersection of the ray and the triangular patch.

c: CLOSE THE BOUNDARY LINE

Let v_2 be the starting point, which is shown in Fig. 5. The shortest path from v_2 to v_{12} passing through all intersection vertices is obtained to generate the closed boundary line L_b using the Dijkstra algorithm [40].

As shown in Fig. 7(a), P_c and P_d are corrected to obtain v_c and v_d, respectively. Let $\overline{v_c v_d}$ be the diameter of the reference sphere S; the vertices v_{I}, v_{II}, v_{III}, v_{IV} and v_{V} located inside S are used as nodes of Dijkstra algorithm, which is shown in Fig. 7(b). The length of each edge is used as the weight, such that the shortest path from v_c to v_d is "$v_c \rightarrow v_{\mathrm{III}} \rightarrow v_d$", which is shown in Fig. 7(c). The closed processing of all the discrete vertices in Fig. 5(b) is sequentially performed to obtain a closed boundary line L_b, which is shown in Fig. 7(d).

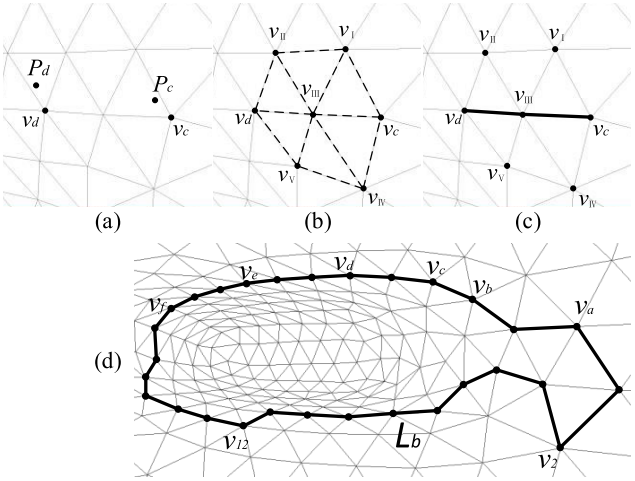


FIGURE 7. Boundary line closure based on the Dijkstra algorithm. (a) P_c and P_d are corrected to v_c and v_d. (b) Nodes of the Dijkstra algorithm. (c) Generation of the shortest path from v_c to v_d. (d) Boundary line closed.

3) BOUNDARY LINE OPTIMIZATION

To create a good visualization effect of the boundary line L_b, a smoothing optimization must be performed. The active contour model (Snakes) is used to define the internal energy E_{int} and the external energy E_{ext} of the vertices on the boundary line [41]. With the greedy algorithm, the vertices on the curve are iteratively moved to the minimum position of the energy in the local range to achieve smoothing optimization. The pseudocode of this algorithm is presented below.

Algorithm 1 Boundary Line Smoothing Based on an Active Contour Model

Input: boundary line L_b with its vertexes v_i; neighborhood k;
Output: smooth boundary line L_b';

foreach vertex v_i **in** L_b to be smoothed **do**
  $E_{i,min} = E_{\text{snake}}(v_i) = E_{\text{int}}(v_i) + E_{\text{ext}}(v_i)$;
  foreach vertex v_j **in** the $k-$neighborhood of v_i **do**
    $E_{\text{snake}}(v_j) = E_{\text{int}}(v_{i-1}, v_j, v_{i+1}) + E_{\text{ext}}(v_{i-1}, v_j, v_{i+1})$;
    if $(E_{\text{snake}}(v_j) < E_{i,min})$ **do**
      $E_{i,min} = E_{\text{snake}}(v_j)$;
      $v_{i,min} = v_j$;
    end
  end
  $v_i = v_j$;
end
return L_b';

As is known from Ref. [41], E_{int} is formulated as follows:

$$E_{int}(v_i) = a \cdot \|v_i - v_{i-1}\| + b \cdot \|v_{i+1} - 2v_i + v_{i-1}\| \tag{31}$$

where a is used to control the length of L_b, and the larger a is, the shorter L_b is. b is used to ensure the smoothness of L_b, and the larger b is, the less the zigzag in L_b. As is also known from Ref. [41], E_{ext} is formulated as follows:

$$E_{\text{ext}}(v_i) = \begin{cases} -k_1(v_i), & \text{if } k_1(v_i) > k_1(v_j) \text{ and } v_j \in D(v_i) \\ C, & \text{else} \end{cases} \tag{32}$$

where C is a large constant, $k_1(v_i)$ and $k_1(v_j)$ represent the maximum principal curvatures of v_i and v_j, respectively, and $D(v_i)$ is a set of vertices lying on or near the principal direction of v_i. The smoothing optimization effect of the boundary line is shown in Fig. 8.

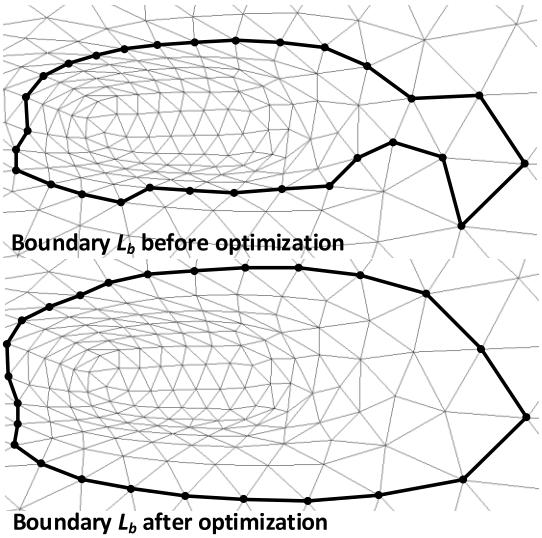


FIGURE 8. Smoothing optimization effect of the boundary using Snakes.

VI. EXPERIMENT AND ANALYSIS

A. INITIALIZATION OF THE EXPERIMENT

1) EXPERIMENTAL PLATFORM

To verify the feasibility of the proposed method, we established an experimental platform, which is shown in Fig. 9. Three kinds of experimental tests were conducted: segmentation visualization, time complexity and segmentation consistency. Additionally, we performed a comprehensive evaluation compared with other mesh segmentation algorithms using the entropy method [42]. The platform was run using MeshLabJS, an open-source digital geometry processing software package, which was deployed on an Apache Server. We developed a mesh segmentation filter plugin in MeshLabJS using JavaScript. A sample set containing Gaussian curvature thresholds was stored in MySQL and encapsulated as JSON, which was uploaded to the Apache Server through WebSocket to train the OS-ELM. Users can upload the test mesh to an Apache Server such that the Gaussian curvature threshold of the test mesh is extracted through the OS-ELM and downloaded to the local browser using HTTP communication. The online mesh segmentation and rendering can be realized through geometric processing and the WebGL graphics rendering function in a browser.

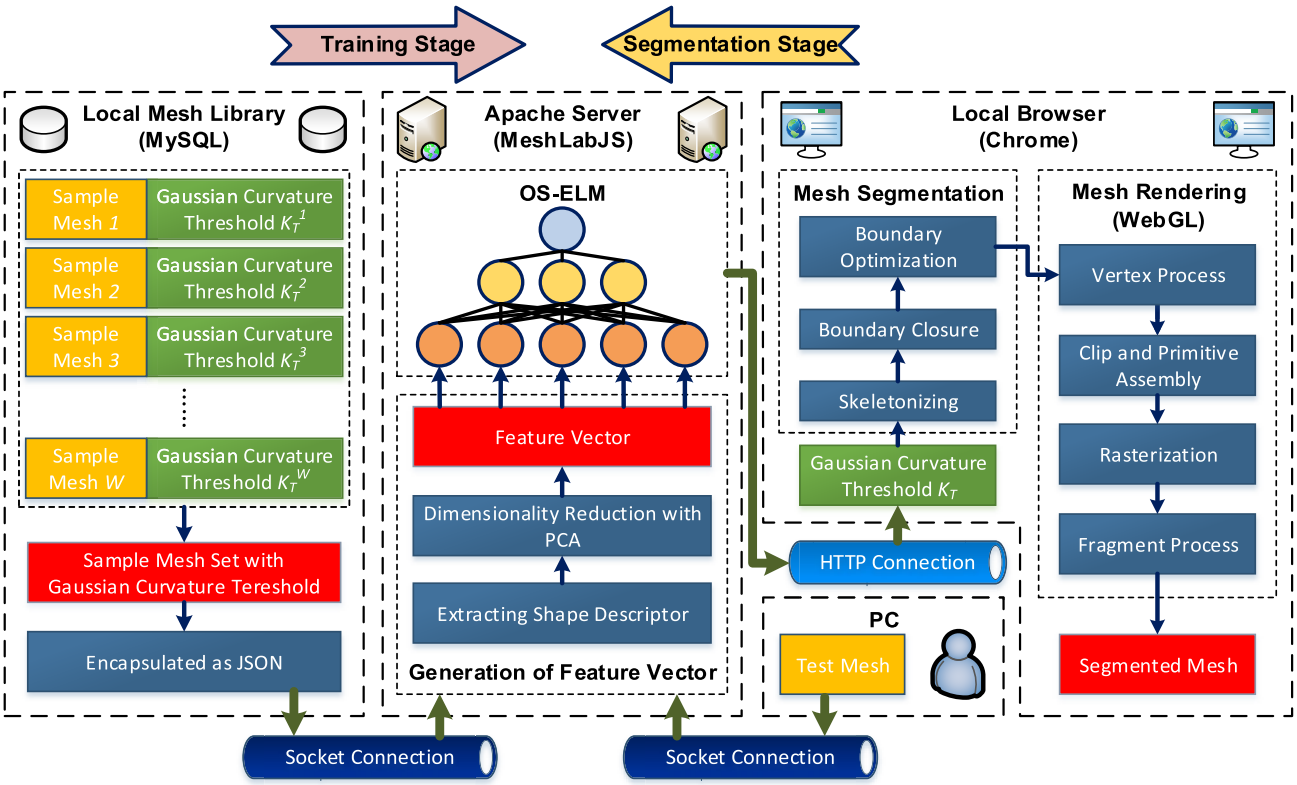


FIGURE 9. Experimental platform of online rapid mesh segmentation based on MeshLabJS, which is divided into two stages: The training stage and segmentation stage.

We used Windows 7 Ultimate x64 as the OS and WebStorm 2016.3.1 as the IDE. The server PC was equipped with an Intel Xeon-E5 2620 V4 2.1 GHz CPU and 32 GB RAM, and the client PC had an Intel(R) i5-7200U 2.5 GHz CPU and 8 GB RAM.

2) EXPERIMENTAL DATASET AND SHAPE DESCRIPTORS

To test the rapidness of our method, we aim to test it on a massive and diverse dataset of typical real-life meshes. We selected a subset of 16 diverse categories containing 10458 meshes from ShapeNetCore [43], which is presented in Table 1. N represents the number of meshes for each category, W represents the number of each categories of training meshes in the initialization stage of the OS-ELM, and W' represents the number of each categories of training meshes in the online incremental learning stage of the OS-ELM. We ensure that the number of training meshes is 5% of the total number of meshes for each category.

The visualization results for 16 types of meshes provided by ShapeNetCore and segmented using our method are shown in Fig. 10, in which different segmented regions are indicated by different colors. At the same time, our segmentation results are compared with the method presented by Xie *et al.* [9], in which the segmentation regions with differences are indicated by dashed circles and shown in Fig. 11.

TABLE 1. Categories of the experimental dataset.

Category	*N*	*W*	*W'*
Airplane	1500	75	1425
Bag	83	4	79
Cap	55	3	52
Car	1500	75	1425
Chair	1500	75	1425
Earphone	69	3	66
Guitar	793	40	753
Knife	420	21	399
Lamp	1500	75	1425
Laptop	445	22	423
Motorbike	336	17	319
Mug	213	11	202
Pistol	307	15	292
Rocket	85	4	81
Skateboard	152	8	144
Table	1500	75	1425
Total	10458	523	9935

3) EXPERIMENTAL PARAMETERS

All of the experimental parameters used in our tests are listed in Table 2.

B. SEGMENTATION VISUALIZATION

The visualization results for 16 types of meshes provided by ShapeNetCore and segmented using our method are shown in Fig. 10, in which different segmented regions are indicated

FIGURE 10. We use our method to obtain the segmentation results for more than 10000 meshes in 16 shape categories from ShapeNetCore. We denote the number of meshes in each category in parentheses.

by different colors. At the same time, our segmentation results are compared with the method presented by Xie *et al.* [9], in which the segmentation regions with differences are indicated by dashed circles and shown in Fig. 11.

When using the method proposed by Xie et al., it is necessary to predefine different regions of the mesh segmentation result in advance, which results in worse effects for regions with sharp features. In contrast, our method adopts the Gaussian curvature threshold as the basis for determining the feature points of the segmentation boundary. Therefore, we pay more attention to the influence of sharp features, which makes our method sensitive to tiny regions with sharp features. As shown in Fig. 11, our method can accurately distinguish between shell and blade regions in Airplane, tire and wheel in Car, lampshade and light source in Lamp, panel and keyboard in Laptop, etc. In contrast, the method proposed by Xie et al. can only perform mesh segmentation according to the predefined regions.

In addition, Gaussian curvature is only a kind of geometric features to the vertex of the mesh, but the current unsupervised algorithms always use different clustering methods to segment and label the triangular patches. If we combine Gaussian curvature with unsupervised methods, we should construct a complex mathematical model contains both geometric features of vertex and triangular patch. Although it may improve the quality of segmentation, the computational complexity is bound to increase.

However, it must be emphasized that the Xie et al. method was combined with the accurate classification of triangular patches via the ELM and the generation of smooth boundary via graph cuts such that the accurate segmentation effect was guaranteed. Meanwhile, although the segmentation results generated by our algorithm exhibit tiny zigzags near the boundaries, there is almost no difference between the method presented by Xie et al. and ours.

TABLE 2. Experimental parameters of the mesh segmentation tests.

Symbol	Description	Value
n	Equal number of frequency histogram [32]	800
H	Number of shape descriptors before dimensionality reduction [6]	242
L	Number of shape descriptors after dimensionality reduction [38]	40
R	Number of hidden layer nodes in OS-ELM [38]	81
φ	Ray angle [39]	5°
a, b	Internal energy coefficients [41]	0.5, 0.5
k	Neighborhood [41]	6

C. TIME COMPLEXITY

The training time and the segmentation time to segment 16 types of meshes are listed in Table 3, which contain six methods: JointBoost presented by Kalogerakis *et al.* [6]; Projective Convolutional Networks presented by Kalogerakis *et al.* [7]; the ELM presented by Xie *et al.* [9]; spectral clustering presented by Tong *et al.* [21]; our method without dimensionality reduction via PCA; and ours with dimensionality reduction via PCA. T_t represents the training time (unit: h), and T_s represents the segmentation time (unit: second). Additionally, we present two histograms that represent the training time and the segmentation time in Fig. 12 and Fig. 13, respectively.

Table 3 shows that for the methods of JointBoost, Projective Convolutional Network and ELM, the average times spent on machine learning training are 7.6 h, 10.3 h and 4.4 h, respectively. Compared with our method, for which the average time is only 2.1 h, these methods require 72.4%, 80.0% and 52.3% more time, respectively. In addition, using the methods of JointBoost, Projective Convolutional Network and ELM for mesh segmentation, the average times are 38.5 s, 48.1 s and 21.9 s, respectively. Compared with the average time of 12.8 s using our method, these methods require 66.7%, 73.2% and 41.3% more time, respectively. Although the average training time and segmentation time of spectral clustering are only 1.7 h and 7.5 s, respectively, which is obviously superior to the other five methods, the segmentation consistency still should be improved compared with the supervised methods. The test results show that the proposed incremental learning mechanism of the OS-ELM has obvious advantages in terms of improving the training speed.

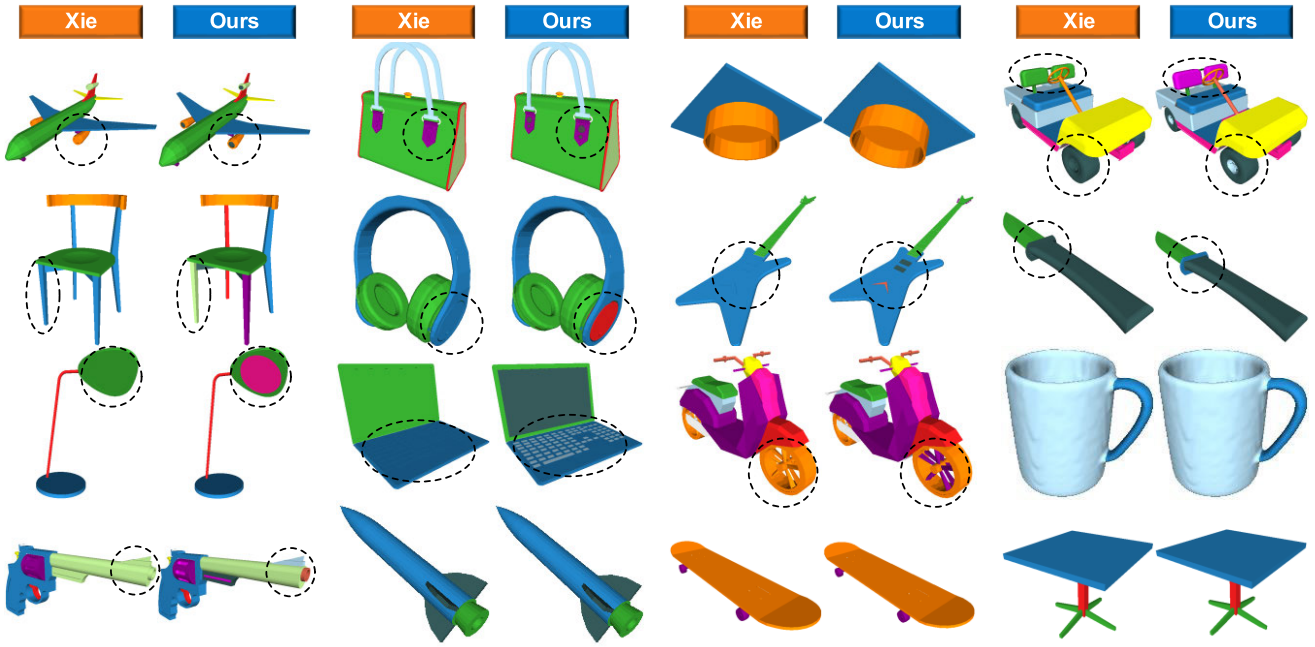


FIGURE 11. Sixteen segmentation results are compared with results from the method presented by Xie et al., in which the segmentation regions with differences are indicated by dashed circles.

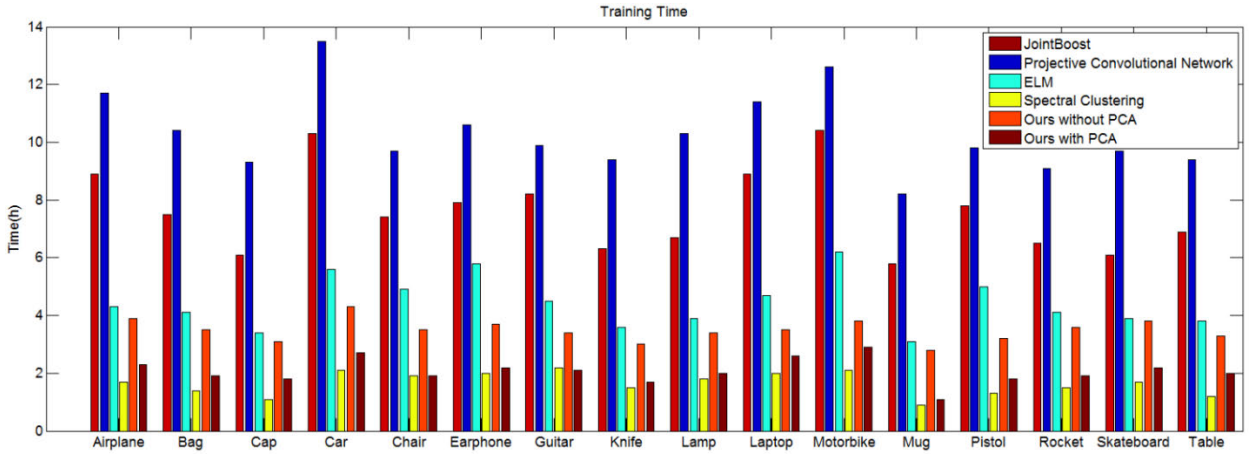


FIGURE 12. Training times of the six methods to segment 16 types of meshes.

In addition, the dimensionality reduction in the shape descriptor vector based on PCA reduces the training time and segmentation time by 40.7% and 30.0%, respectively, which indicates that the dimensionality reduction based on PCA reduces the computational complexity of the algorithm. It has a significant effect in improving the speed of our algorithm.

D. SEGMENTATION CONSISTENCY

We use the PSB to analyze the consistency of the segmentation results. This benchmark offers four quantitative metrics, i.e., the Cut Discrepancy (CD), Hamming Distance (HD), Rand Index (RI) and Consistency Error (CE), to measure the similarity between automatic segmentations and manual segmentations together with source code for computing these metrics. Smaller values indicate better segmentation results. We conducted an evaluation of our method on the benchmark by the automatic mode and using the default parameters, and a snapshot of our segmentation results in each category is illustrated in Fig. 14. The detailed quantitative results are presented in Table 4, in which OWTPCA represents our method without PCA, OWPCA represents our method with PCA, JB represents JointBoost [6], PCN represents Projective Convolutional Network [7], ELM represents the method presented by Xie *et al.* [9], and SC represents spectral clustering [21]. Based on the assumptions that people

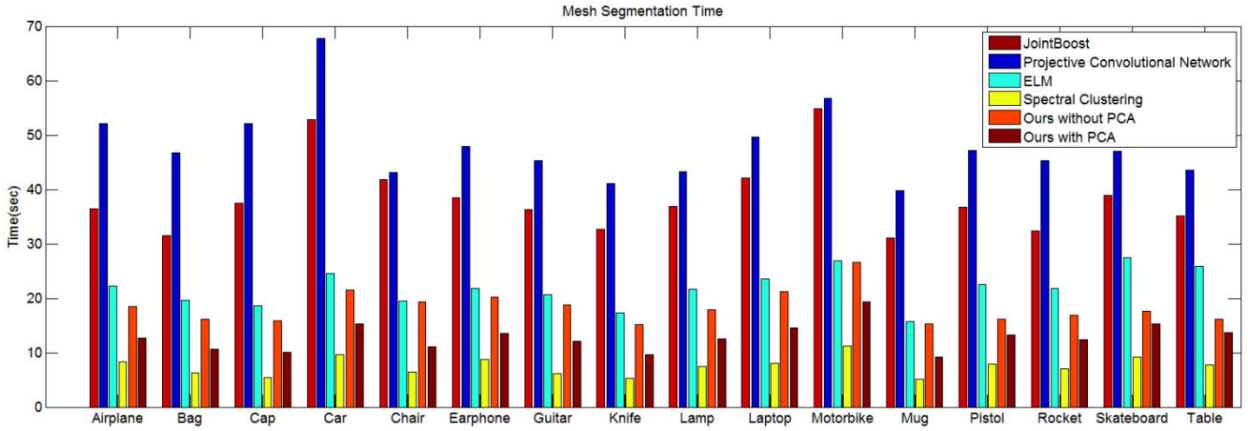


FIGURE 13. Segmentation times of the six methods to segment 16 types of meshes.

TABLE 3. Training time and segmentation time of 16 types of meshes with six different methods.

Category	JointBoost		Projective Convolutional Network		ELM		Spectral Clustering		Ours without PCA		Ours with PCA	
	T_t	T_s	T_t	T_s	T_t	T_s	T_t	T_s	T_t	T_s	T_t	T_s
Airplane	8.9	36.4	11.7	52.1	4.3	22.3	1.7	8.4	3.9	18.5	2.3	12.7
Bag	7.5	31.5	10.4	46.7	4.1	19.7	1.4	6.3	3.5	16.2	1.9	10.6
Cap	6.1	37.4	9.3	52.1	3.4	18.6	1.1	5.5	3.1	15.9	1.8	10.1
Car	10.3	52.8	13.5	67.8	5.6	24.6	2.1	9.7	4.3	21.5	2.7	15.3
Chair	7.4	41.8	9.7	43.1	4.9	19.5	1.9	6.4	3.5	19.4	1.9	11.1
Earphone	7.9	38.5	10.6	47.9	5.8	21.8	2.0	8.7	3.7	20.2	2.2	13.6
Guitar	8.2	36.3	9.9	45.3	4.5	20.6	2.2	6.2	3.4	18.7	2.1	12.1
Knife	6.3	32.7	9.4	41.1	3.6	17.3	1.5	5.3	3.0	15.2	1.7	9.6
Lamp	6.7	36.9	10.3	43.2	3.9	21.7	1.8	7.5	3.4	17.9	2.0	12.5
Laptop	8.9	42.1	11.4	49.7	4.7	23.6	2.0	8.1	3.5	21.2	2.6	14.6
Motorbike	10.4	54.9	12.6	56.8	6.2	26.9	2.1	11.2	3.8	26.6	2.9	19.3
Mug	5.8	31.1	8.2	39.8	3.1	15.7	0.9	5.2	2.8	15.3	1.1	9.2
Pistol	7.8	36.7	9.8	47.2	5.0	22.5	1.3	7.9	3.2	16.2	1.8	13.2
Rocket	6.5	32.4	9.1	45.3	4.1	21.8	1.5	7.1	3.6	16.9	1.9	12.4
Skateboard	6.1	38.9	9.7	47.1	3.9	27.4	1.7	9.2	3.8	17.6	2.2	15.3
Table	6.9	35.2	9.4	43.6	3.8	25.9	1.2	7.8	3.3	16.1	2.0	13.7
Average	7.6	38.5	10.3	48.1	4.4	21.9	1.7	7.5	3.5	18.3	2.1	12.8

tend to segment objects consistently and that segmentation algorithms ought to mimic what people do, the manual segmentations are regarded as the ground truth. We use the results of manual segmentation of the ShapeNetCore model provided by Kalogerakis *et al.* [7] as the ground truth. To provide comparisons with the highly performing state-of-the-art methods, i.e., JointBoost [6], Projective Convolutional Network [7] and spectral clustering [21], we present the detailed Rand Index score statistics of each category in Table 4. Our method achieves average Rand Index errors of 10.0 and 9.9 with or without dimensionality reduction via PCA; it clearly outperforms the other four automatic methods and manual segmentation. In addition, the four metrics of PSB do not change greatly after dimensionality reduction in the shape descriptor vector via PCA, which indicates that the dimensionality reduction based on PCA has a minor impact on the segmentation result. Altogether, the test results demonstrate the advantage of the low computational complexity and segmentation consistency of our method.

E. COMPREHENSIVE EVALUATION

To comprehensively evaluate the abovementioned six segmentation methods, we used the entropy method [43] to perform a comprehensive evaluation according to the following 9 metrics: training time; segmentation time; cut discrepancy (CD); Hamming distance (HD); Hamming distance based on missing rate (HD-Rm); Hamming distance based on false alarm rate (HD-Rf); Rand Index (RI); global consistency error (GCE); and local consistency error (LCE). The entropy method fully considers the decisive role of metrics on the evaluation results, weakens the influence of subjective factors on the evaluation results, and helps to obtain more objective and true evaluation results. The values of 9 metrics of 6 segmentation methods are listed in Table 5, in which OWTPCA represents our method without PCA, OWPCA represents our

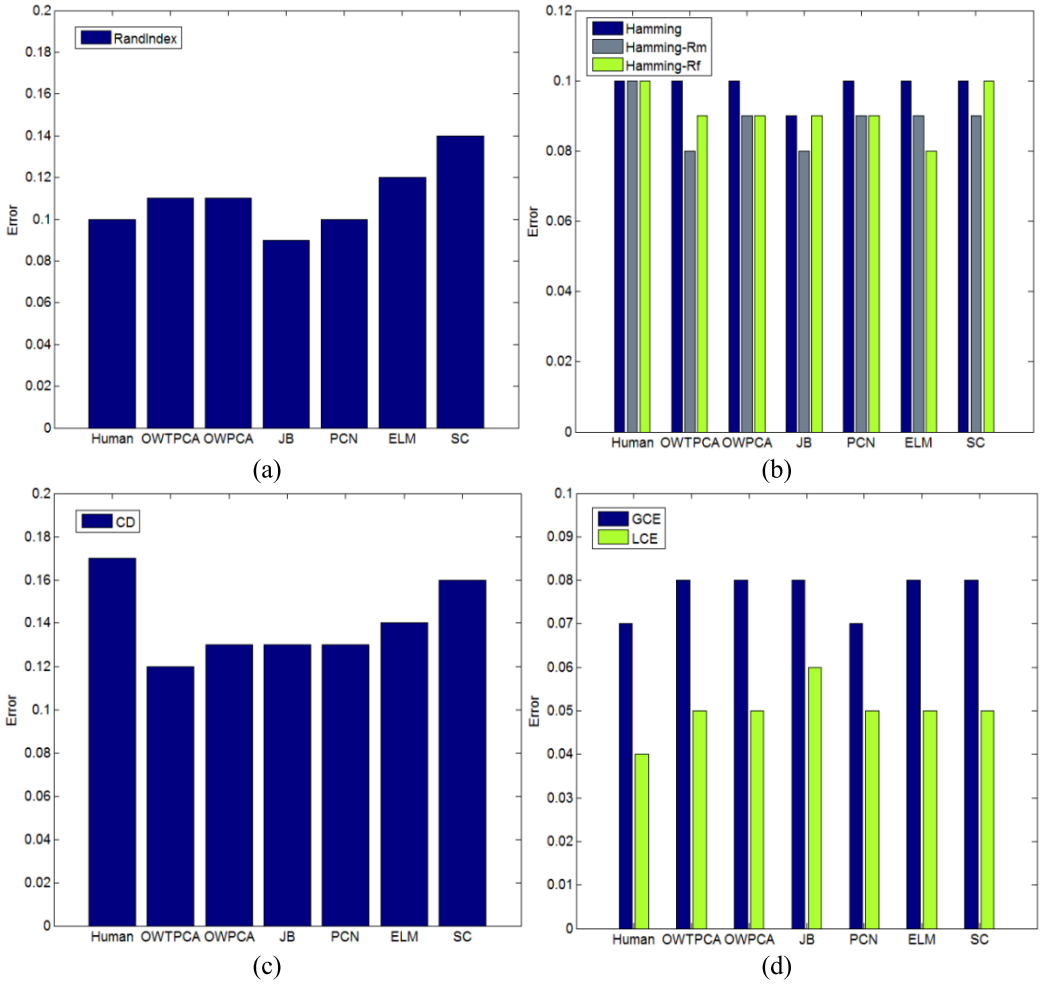


FIGURE 14. Quantitative evaluation of our method on the PSB [4].

TABLE 4. Rand index values of our method and some state-of-the-art methods across all 16 categories, where PSB [4] denotes the error included in the manual segmentations.

Method	Human	OWTPCA	OWPCA	JB	PCN	ELM	SC
Airplane	9.2	12.9	12.2	9.6	7.7	9.5	14.0
Bag	10.6	10.0	12.0	8.6	11.5	10.1	9.9
Cap	12.4	10.3	8.5	11.4	6.4	13.9	9.1
Car	9.5	10.4	10.2	10.1	6.1	10.3	14.7
Chair	8.9	7.2	9.0	11.7	11.1	12.9	14.7
Earphone	11.5	10.6	11.7	11.1	7.4	13.9	14.9
Guitar	12.3	9.0	7.9	10.6	9.8	10.8	12.9
Knife	12.7	7.6	7.2	9.0	8.4	10.8	14.0
Lamp	8.9	8.7	8.1	12.3	8.4	12.9	14.7
Laptop	9.0	11.9	8.6	10.5	8.8	8.9	14.3
Motorbike	8.1	11.6	13.1	8.8	9.2	13.8	13.5
Mug	13.6	12.0	13.2	10.8	11.4	11.0	11.7
Pistol	11.0	7.3	10.3	8.1	10.5	13.1	11.6
Rocket	10.8	11.8	10.4	10.1	6.5	9.9	9.6
Skateboard	12.0	9.2	7.5	9.8	11.3	9.9	13.1
Table	9.3	8.1	9.2	9.4	7.7	10.1	11.5
Average	10.6	9.9	10.0	10.1	8.9	11.4	12.8

method with PCA, JB represents JointBoost [6], PCN represents Projective Convolutional Network [7], ELM represents the method presented by Xie *et al.* [9], and SC represents spectral clustering [21].

The above nine metrics have negative influences, and the values of these metrics must be negatively normalized. The online mesh segmentation algorithm requires a lower time complexity, so the weights of the training time and the

TABLE 5. The values of 9 metrics of 6 segmentation methods.

Metrics	Training Time	Segmentation Time	CD	HD	HD-Rm	HD-Rf	RI	GCE	LCE
OWTPCA	3.5	18.3	0.12	0.1	0.08	0.09	0.11	0.08	0.05
OWPCA	2.1	12.8	0.13	0.1	0.09	0.09	0.11	0.08	0.05
JB	7.6	38.5	0.13	0.09	0.08	0.09	0.09	0.08	0.06
PCN	10.3	48.1	0.13	0.1	0.09	0.09	0.1	0.07	0.05
ELM	4.4	21.9	0.14	0.1	0.09	0.08	0.12	0.08	0.05
SC	1.7	7.5	0.16	0.1	0.09	0.1	0.14	0.08	0.05

TABLE 6. The entropy weights of the 9 metrics.

Metrics	Training Time	Segmentation Time	CD	HD	HD-Rm	HD-Rf	RI	GCE	LCE
Entropy weight	0.438	0.460	0.265	0.241	0.220	0.317	0.217	0.240	0.228

TABLE 7. The comprehensive scores of the six methods.

Method	OWTPCA	OWPCA	JB	PCN	ELM	SC
Score	65.4	66.2	57.8	59.9	62.1	61.7

segmentation time are both 0.15, and the remaining weights are 0.10. The entropy weights of 9 metrics are calculated and reported in Table 6.

Finally, the comprehensive scores of the six methods are calculated according to the entropy weight vector and the values of 9 metrics; they are reported in Table 7, in which OWPCA represents our method with PCA, JB represents JointBoost [6], PCN represents Projective Convolutional Network [7], ELM represents the method presented by Xie *et al.* [9], and SC represents spectral clustering [21].

The above evaluation results demonstrate that under the premise of rapid mesh segmentation, our method obtains the highest score, has obvious advantages in terms of algorithm execution efficiency and rapidity, and can take into account segmentation visualization effect and segmentation speed; it is thus suitable for execution in the Web environment.

VII. CONCLUSION

In this paper, we present an online rapid mesh segmentation method based on an OS-ELM that aims to achieve online rapid mesh segmentation in the Web environment. The statistical method is used to obtain the Gaussian curvature threshold of the sample mesh, and the training sample set of the OS-ELM is collected. The online incremental learning mechanism of the OS-ELM is performed to increase the training speed. The Gaussian curvature threshold of the mesh to be segmented is extracted through the OS-ELM, and relevant processes of the boundary lines are executed to realize rapid mesh segmentation in the Web environment. Finally, we conduct some experimental tests on these segmentation results using the PSB and ShapeNetCore. This study has certain significance for implementing digital geometry processing of grids in the Web environment and for implementing digital geometry processing in the Web environment.

In the future, research should focus on two aspects. First, the segmentation consistency of our method should be improved through research regarding the segmentation boundary generation method according to multimetric fusion rather than a single metric. Second, the existing method performs better on the mesh with obvious boundary features. However, the segmentation consistency of mesh provided by ShapeNetCore with complex surfaces is not sufficiently satisfactory. A more general segmentation algorithm should be studied.

REFERENCES

[1] R. S. V. Rodrigues, J. F. M. Morgado, and A. J. P. Gomes, "Part-based mesh segmentation: A survey," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 235–274, 2018. doi: 10.1111/cgf.13323.

[2] R. Song, Y. Liu, R. R. Martin, and P. L. Rosin, "Mesh saliency via spectral processing," *ACM Trans. Graph.*, vol. 33, no. 1, p. 6, 2014. doi: 10.1145/2530691.

[3] F. Buonamici, R. Furferi, L. Governi, S. Lazzeri, K. S. McGreevy, M. Servi, E. Talanti, F. Uccheddu, and Y. Volpe, "A practical methodology for computer-aided design of custom 3D printable casts for wrist fractures," in *The Visual Computer*, 2019. doi: 10.1007/s00371-018-01624-z.

[4] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Trans. Graph.*, vol. 28, no. 3, p. 73, 2009. doi: 10.1145/1531326.1531379.

[5] M. Wencka and K. Walas, "Review of 3D objects segmentation methods," in *Automation 2017: Innovations in Automation, Robotics and Measurement Techniques* (Advances in Intelligent Systems and Computing), vol. 550, R. Szewczyk, C. Zielinski, and M. Kaliczynska Eds. Cham, Switzerland: Springer, 2017, pp. 595–604.

[6] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D mesh segmentation and labeling," *ACM Trans. Graph.*, vol. 29, no. 4, p. 102, Jul. 2010. doi: 10.1145/1778765.1778839.

[7] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3D shape segmentation with projective convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3779–3788.

[8] H. Benhabiles, G. Lavoué, J.-P. Vandeborre, and M. Daoudi, "Learning boundary edges for 3D-mesh segmentation," *Comput. Graph. Forum*, vol. 30, no. 8, pp. 2170–2182, Dec. 2011. doi: 10.1111/j.1467-8659.2011.01967.x.

[9] Z. Xie, K. Xu, L. Liu, and Y. Xiong, "3D shape segmentation and labeling via extreme learning machine," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 85–95, 2014. doi: 10.1111/cgf.12434.

[10] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang, "Projective feature learning for 3D shapes with multi-view depth images," *Comput. Graph. Forum*, vol. 34, no. 7, pp. 1–11, 2015. doi: 10.1111/cgf.12740.

[11] K. Guo, D. Zou, and X. Chen, "3D mesh labeling via deep convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 1, pp. 1–12, 2015. doi: 10.1145/2835487.

[12] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, p. 210, 2016. doi: 10.1145/2980179.2980238.

[13] L. Yi, L. Guibas, A. Hertzmann, V. G. Kim, H. Su, and E. Yumer, "Learning hierarchical shape segmentation and labeling from online repositories," *ACM Trans. Graph.*, vol. 36, no. 4, p. 70, 2017. doi: 10.1145/3072959.3073652.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660.

[15] B. Graham, M. Engelcke, and L. van der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9224–9232.

[16] P. Luo, Z. Wu, C. Xia, L. Feng, and T. Ma, "Co-segmentation of 3D shapes via multi-view spectral clustering," *Vis. Comput.*, vol. 29, nos. 6–8, pp. 587–597, Jun. 2013. doi: 10.1007/s00371-013-0824-2.

[17] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, and L. Liu, "Unsupervised 3D shape segmentation and co-segmentation via deep learning," *Comput. Aided Geom. Des.*, vol. 43, pp. 39–52, 2016. doi: 10.1016/j.cagd.2016.02.015.

[18] P. Theologou, I. Pratikakis, and T. Theoharis, "Unsupervised spectral mesh segmentation driven by heterogeneous graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 397–410, Feb. 2017. doi: 10.1109/TPAMI.2016.2544311.

[19] X. Wang, B. Zhou, Z. Wang, D. Zou, X. Chen, and Q. Zhao, "Efficiently consistent affinity propagation for 3D shapes co-segmentation," *Vis. Comput.*, vol. 34, nos. 6–8, pp. 997–1008, Jun. 2018. doi: 10.1007/s00371-018-1538-2.

[20] H. Zhang, C. Wu, J. Deng, Z. Liu, and Y. Yang, "A new two-stage mesh surface segmentation method," *Vis. Comput.*, vol. 34, no. 11, pp. 1597–1615, Nov. 2018. doi: 10.1007/s00371-017-1434-1.

[21] W. Tong, X. Yang, M. Pan, and F. Chen, "Spectral mesh segmentation via $\ell 0$ gradient minimization," *IEEE Trans. Vis. Comput. Graph.*, to be published. doi: 10.1109/TVCG.2018.2882212.

[22] P.-A. Fayolle and A. Pasko, "Segmentation of discrete point clouds using an extensible set of templates," *Vis. Comput.*, vol. 29, no. 5, pp. 449–465, May 2013. doi: 10.1007/s00371-012-0749-1.

[23] Y. Fan, M. Wang, N. Geng, D. He, J. Chang, and J. J. Zhang, "A self-adaptive segmentation method for a point cloud," *Vis. Comput.*, vol. 34, no. 5, pp. 659–673, May 2018. doi: 10.1007/s00371-017-1405-6.

[24] D. Mejia, O. Ruiz-Salguero, J. R. Sánchez, J. Posada, A. Moreno, and C. A. Cadavid, "Hybrid geometry / topology based mesh segmentation for reverse engineering," *Comput. Graph.*, vol. 73, pp. 47–58, Jun. 2018. doi: 10.1016/j.cag.2018.03.004.

[25] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel, "Mesh segmentation driven by Gaussian curvature," *Vis. Comput.*, vol. 21, nos. 8–10, pp. 659–668, Sep. 2005. doi: 10.1007/s00371-005-0319-x.

[26] S. Huang, J. Sun, Y. Yang, Y. Fang, and P. Lin, "Multi-frame super-resolution reconstruction based on gradient vector flow hybrid field," *IEEE Access*, vol. 5, pp. 21669–21683, 2017. doi: 10.1109/access.2017.2757239.

[27] L. Liu, Y. Sheng, G. Zhang, and H. Ugail, "Graph cut based mesh segmentation using feature points and geodesic distance," in *Proc. Int. Conf. Cyberworlds (CW)*, Oct. 2015, pp. 115–120. doi: 10.1109/CW.2015.31.

[28] K. Saji, M. Umehara, and K. Yamada, "Coherent tangent bundles and gauss–bonnet formulas for wave fronts," *J. Geometric Anal.*, vol. 22, no. 2, pp. 383–409, Apr. 2012. doi: 10.1007/s12220-010-9193-5.

[29] N. Dyn, K. Hormann, S.-J. Kim, and D. Levin, "Optimizing 3D triangulations using discrete curvature analysis," *Math. Methods Curves Surf.*, vol. 1, pp. 135–146, Jan. 2001.

[30] D. Liu, M. Wang, and G. Shen, "A new combinatorial characteristic parameter for clustering-based traffic network partitioning," *IEEE Access*, vol. 7, pp. 40175–40182, 2019. doi: 10.1109/access.2019.2905618.

[31] J. Ylioinas, N. Poh, J. Holappa, and M. Pietikäinen, "Data-driven techniques for smoothing histograms of local binary patterns," *Pattern Recognit.*, vol. 60, pp. 734–747, Dec. 2016. doi: 10.1016/j.patcog.2016.06.029.

[32] Z. Gan, N. Zeng, F. Zou, J. Chen, M. Du, L. Liao, H. Li, and Y. Zhang, "Multilevel segmentation optimized by physical information for gridding of microarray images," *IEEE Access*, vol. 7, pp. 32146–32153, 2019. doi: 10.1109/access.2019.2900249.

[33] L. Wan, C. Zou, and H. Zhang, "Full and partial shape similarity through sparse descriptor reconstruction," *Vis. Comput.*, vol. 33, no. 12, pp. 1497–1509, Dec. 2017. doi: 10.1007/s00371-016-1293-1.

[34] Y. Cheng, J. Zhu, and X. Lin, "An enhanced incremental SVD algorithm for change point detection in dynamic networks," *IEEE Access*, vol. 6, pp. 75442–75451, Nov. 2018. doi: 10.1109/access.2018.2883647.

[35] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006. doi: 10.1016/j.neucom.2005.12.126.

[36] J. H. Huang, Z. L. Yu, and Z. H. Gu, "A clustering method based on extreme learning machine," *Neurocomputing*, vol. 277, pp. 108–119, Feb. 2018. doi: 10.1016/j.neucom.2017.02.100.

[37] J. R. Sendra and J. Sendra, "Computation of moore–penrose generalized inverses of matrices with meromorphic function entries," *Appl. Math. Comput.*, vol. 313, pp. 355–366, Nov. 2017. doi: 10.1016/j.amc.2017.06.007.

[38] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006. doi: 10.1109/TNN.2006.880583.

[39] M. Zhao, Y. Yuan, X. Zhang, Z. Shi, and Y. Wang, "A novel key frames matching approach for human locomotion interpolation," *Multimedia Tools Appl.*, vol. 77, no. 6, pp. 7779–7794, Mar. 2018. doi: 10.1007/s11042-017-4677-y.

[40] Q. D. Ho and M. S. Lee, "A Zone-Based Approach for Scalable Dynamic Traffic Grooming in Large WDM Mesh Networks," *J. Lightw. Technol.*, vol. 25, no. 1, pp. 261–270, Jan. 2007. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-25-1-261

[41] M. Jung and H. Kim, "Snaking across 3D meshes," in *Proc. 12th Pacific Conf. Comput. Graph. Appl. (PG)*, Oct. 2004, pp. 87–93. doi: 10.1109/PCCGA.2004.1348338.

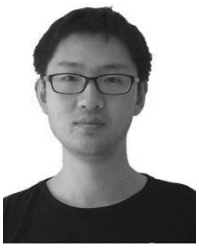
[42] C.-C. Wang, B. C. Jiang, and W.-C. Lin, "Evaluation of effects of balance training from using wobble board-based exergaming system by MSE and MMSE techniques," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 6, pp. 1745–1754, Nov. 2018. doi: 10.1007/s12652-017-0594-1.

[43] W. Wu, Y. Huang, R. Kurachi, G. Zeng, G. Xie, R. Li, and K. Li, "Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks," *IEEE Access*, vol. 6, pp. 45233–45245, 2018. doi: 10.1109/access.2018.2865169.

FEIYU ZHAO was born in Wuhan, Hubei, China, in 1992. He received the M.S. degree in mechanical engineering from Wuhan Polytechnic University, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree in mechanical engineering with the Wuhan University of Technology, Wuhan.

His research interests include the computer graphics, intelligent manufacturing, and CAD.

BUYUN SHENG was born in Huarong, Hunan, China, in 1964. He received the B.S., M.S., and Ph.D. degrees in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 1985, 1988, and 2001, respectively, where he is currently a Professor with the School of Mechanical and Electronic Engineering. He is the author of two books and more than 100 articles. His research interests include the digital and intelligent manufacturing, cloud manufacturing and service, computer graphics, and CAD/CAPP/PDM.

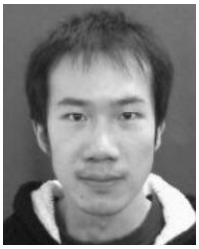


XIYAN YIN was born in Xiangyang, Hubei, China, in 1988. He received the B.S. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2011, where he is currently pursuing the Ph.D. degree in mechanical engineering.

His research interests include the digital manufacturing, intelligent manufacturing, and knowledge engineering.

HUI WANG was born in Nanchang, Jiangxi, China, in 1993. He received the B.S. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2015, where he is currently pursuing the Ph.D. degree in mechanical engineering.

His research interests include the scheduling, digital manufacturing, and project management.



XINCHENG LU was born in Guilin, Guangxi, China, in 1991. He received the M.S. degree in mechanical engineering from the Guilin University of Electronic Technology, Guilin, China, in 2018. He is currently pursuing the Ph.D. degree in mechanical engineering with the Wuhan University of Technology, Wuhan, China.

His research interests include the digital manufacturing, intelligent manufacturing, and CAD.

YUNCHENG ZHAO was born in Jingzhou, Hubei, China, in 1994. He received the B.S. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2017, where he is currently pursuing the M.S. degree in mechanical engineering.

His research interests include the digital manufacturing, intelligent manufacturing, and CAD.