Research Article Vol. 37, No. 11 / November 2020 / Journal of the Optical Society of America A 1711

Journal of the Optical Society of America A

OPTICS, IMAGE SCIENCE, AND VISION

Simplification method for 3D Terracotta Warrior fragments based on local structure and deep neural networks

Guohua Geng,[1] Jie Liu,[1] Xin Cao,[1,4] Yangyang Liu,[1] Wei Zhou,[1] Fengjun Zhao,[1] Linzhi Su,[1] Kang Li,[1,5] and Mingquan Zhou[2,3]

[1]School of Information Science and Technology, Northwest University, Xi'an, Shaanxi, China
[2]College of Information Science and Technology, Beijing Normal University, Beijing, China
[3]Engineering Research Center of Virtual Reality and Applications, Ministry of Education, Beijing Key Laboratory of Digital Preservation and Virtual Reality for Cultural Heritage, Beijing Normal University, Beijing, China
[4]e-mail: xin_cao@163.com
[5]e-mail: likang@nwu.edu.cn

Received 17 June 2020; revised 12 August 2020; accepted 2 September 2020; posted 2 September 2020 (Doc. ID 400571); published 7 October 2020

The emergence of the three-dimensional (3D) scanner has greatly benefited archeology, which can now store cultural heritage artifacts in computers and present them on the Internet. As many Terracotta Warriors have been predominantly found in fragments, the pre-processing of these fragments is very important. The raw point cloud of the fragments has lots of redundant points; it requires an excessively large storage space and much time for post-processing. Thus, an effective method for point cloud simplification is proposed for 3D Terracotta Warrior fragments. First, an algorithm for extracting feature points is proposed that is based on local structure. By constructing a k-dimension tree to establish the k-nearest neighborhood of the point cloud, and comparing the feature discriminant parameter and characteristic threshold, the feature points, as well as the non-feature points, are separated. Second, a deep neural network is constructed to simplify the non-feature points. Finally, the feature points and the simplified non-feature points are merged to form the complete simplified point cloud. Experiments with the public point cloud data and the real-world Terracotta Warrior fragments data are designed and conducted. Excellent simplification results were obtained, indicating that the geometric feature can be preserved very well.

https://doi.org/10.1364/JOSAA.400571

1. INTRODUCTION

Terracotta Warriors, known as one of the world's wonders, have become an important channel for spreading Chinese culture. In 1974, some fragments of Terracotta Warriors were discovered near Xi'an, Shanxi province. After two years of investigation and excavation, more figures of horses and warriors, which consisted of war chariots, infantrymen, cavalrymen, and other armed servicemen, were discovered. As many Terracotta Warriors have been predominantly found in fragments, the traditional restoration of cultural relics not only consumes a lot of manpower but also causes secondary damage to the Terracotta Warriors [1,2]. With the rapid development of laser scanner technology, the Terracotta Warrior fragments can be restored and preserved in a digital way, and the restoration work can be done virtually. However, the raw point cloud collected by scanning devices usually contains a lot of redundant points. It requires excessive amounts of main memory and much more time for subsequent processing in later applications such as classification, registration, 3D reconstruction and so on. Therefore, reducing the complexity of the raw point cloud is a critical issue in the restoration of Terracotta Warriors.

According to the principle of reduction, the traditional point cloud simplification methods can be generally classified into two categories: polygonal mesh-based methods and point cloud-based methods. The former methods simplify the point cloud by converting the point cloud to polygonal mesh model, simplify the model, and then reduce the points based on some rules. Wei *et al.* [3] represented a method for mesh simplification, which was based on visual saliency weighting. The method described the local feature values of the surface through the relationship between Voronoi poles and the sample points. Li *et al.* [4] proposed a uniform simplification algorithm for scattered point cloud data. The algorithm was based on the open-source C++ programming library point cloud library (PCL). First, a

k-nearest neighborhood voxel grid was built by voxel grid class in the PCL. Then, the k-nearest neighborhood distance was calculated and the normal was estimated. Then the barycenter of each small voxel grid was established, which replaced all point cloud in the voxel grid to achieve point cloud simplification. Finally, the simplified point cloud was reconstructed and displayed with a triangular mesh by the greedy projection triangulation class. Sun *et al.* [5] introduced the medial mesh, a discrete representation of the medial axis, to simplify the point cloud. The medial mesh is a two-dimensional (2D) simplicial complex coupled with a radius function that provides a piecewise linear approximation to the medial axis. The main disadvantages of these polygonal mesh-based methods are that they are highly complex and time-consuming. In contrast, point cloud-based methods can consume the point cloud directly. Since there is a lack of topological structures, it is a difficult task to remove redundant points while preserving the appearance properties, features, and contour of the raw point cloud in the simplified point cloud. Huang *et al*. [6] proposed a simplification method with a geometric feature reservation. First, points were distributed into uniform grids. Bounding spheres were searched in the relevant bounding sphere. Second, a specified function was defined to measure the curvature of each point so that feature points could be extracted and reserved. Finally, non-feature points in the bounding spheres were simplified according to the threshold of normal vectors, the inner product. Wang *et al*. [7] presented a feature points detection algorithm based on curvature and density. First, the feature parameter of each point was calculated. Then, the density of data was defined by using an octree. It was divided by the maximum distance from model center to data points and applied as the feature threshold to determine the feature points. The feature points were recognized when its density parameter was bigger than the threshold. Chang *et al*. [8] proposed a k-means clustering algorithm based on boundary reservation to simplify the point cloud. The algorithm first used the k-dimension (kd-tree) to initialize the center of mass, and then used the XY boundary extraction algorithm to preserve the boundary integrity. Finally, the cluster was subdivided according to the curvature level, so that the necessary points were kept in the high-curvature region, and some were kept uniform in the low-curvature region. Shi *et al.* [9] extended an adaptive simplification of the point cloud using k-means clustering. An automatic recursive subdivision scheme was designed to pick out representative points and remove redundant points. To maintain the integrity of the border, an automatic boundary cluster detection algorithm was developed. The method was mainly impacted by two factors: user-defined space interval and normal vector tolerance. By adjusting the two parameters, each level of detail and amount could be obtained. Chen *et al.* [10] presented a method for point cloud simplification. The proposed method used the position of points, the normal vector, and the curvature to detect the feature points and used Gauss map clustering for sharp feature detection. It calculated the dynamic simplification ratio in different regions, and used a lower simplified rate in the feature region and a high simplified rate in the flatness region to retain an appropriate density and avoid producing blank areas and holes. Ji *et al.* [11] defined the detail feature points simplified algorithm (DFPSA). A k-neighborhood search method based on distance and density was proposed to find the k-neighborhood of each point as accurately as possible. Feature points and non-feature points were reduced in different ways. Li *et al.* [12] proposed a synthetic down-sampling method for point cloud simplification. A coarse-to-fine feature extraction manner was designed with normal vectors deviation and k-means clustering methods, which could concentrate more sample points in regions of the high curvature. Moreover, the directed Hausdorff distance approach was employed for sampling in an edge-preserving manner. Chen *et al.* [13] defined an algorithm of extracting feature points based on the multiple parameters hybridization method. Based on the feature parameters, the characteristic threshold and the feature discriminant parameter could be defined and figured out. Feature points were recognized by the defined rule. The proposed algorithm not only extracted the steep feature points but also identified the boundary points. The experiment verified that the algorithm was not effective in thin-wall models. Markovic *et al.* [14] proposed a method for feature-sensitive simplification of the 3D point cloud that was based on ε insensitive support vector regression (ε-SVR). By the flatness property of ε-SVR, the effective recognition of points in high-curvature areas of scanned lines was exploited. Besides, the points in the vicinity of sharp edges were effectively detected by the method without additional processing.

In recent years, deep learning has gained significant progress in the field of point cloud processing. The first deep learning architecture for consuming the point cloud was presented by Qi *et al.* [15] and named as PointNet. The basic idea of PointNet was to learn the spatial encoding of each point and then aggregate all individual point features to a global point cloud signature. PointNet provided a unified approach to several 3D recognition tasks including object classification, part segmentation, and semantic segmentation. However, PointNet could not capture local structures induced by the metric space points live in, limiting its ability to recognize fine-grained patterns and generalizability to complex scenes. To learn richer local structures, many specialized neural modules have been subsequently and rapidly introduced. These notable extension modules can be generally categorized as neighborhood features pooling [16–19], graph-based convolution [20–25], and kernel-based convolution [26–30]. A hierarchical neural network that applied PointNet recursively on a nested partitioning of the input points is proposed in Ref. [16] and named as PointNet++, which learned deep point cloud features efficiently and robustly. Li *et al.* [17] proposed the novel SO-Net that performed hierarchical feature extraction for point cloud by explicitly modeling the spatial distribution of input points and systematically adjusting the receptive field overlap. The SO-Net guaranteed invariance to the order of input points, by the network design and permutation invariant SOM training. It outperformed other deep learning-based approaches in the field of point cloud classification and shape retrieval. Based on PointNet, Yang *et al.* [20] proposed an auto-encoder (AE) that is referenced as FoldingNet. On the encoder side, a graph-based enhancement was enforced to promote local structures on top of PointNet. A novel folding-based decoder deformed a canonical 2D grid onto the underlying 3D object surface of a point cloud, achieving low reconstruction errors even for objects with delicate structures. Wang *et al.* [19] proposed a new neural

network module dubbed EdgeConv suitable for CNN-based high-level tasks on point cloud, including classification and segmentation. EdgeConv acted on graphs dynamically computed in each layer of the network. EdgeConv incorporated local neighborhood information and could be stacked and applied to learn global shape properties. Hua *et al.* [25] presented a convolutional neural network. At the core of the network was pointwise convolution, a new convolution operator that could be applied at each point of a point cloud. The network could yield competitive accuracy in both semantic segmentation and object recognition task. A recent work by Dovrat *et al.* [31] proposed a task-specific sampling method. The network, termed S-NET, consumed a point cloud and output a down sampled point cloud that was optimized for a particular task. The generated point cloud was not guaranteed to be a subset of the input. Therefore, in a post-processing step, the clouds were replaced by their nearest neighbor points in the raw point cloud, which yielded a subset of the input. Lang *et al.* [32] introduced a novel differentiable relaxation for point cloud sampling. They employed a soft projection operation that represents output points as a weighted average of points in the input. During training the network, the soft projection operation replaced the regression of optimal points in the ambient space with multiple classification problems in local neighborhoods of the input. In summary, the variety of deep learning applications for point cloud expanded substantially, and these methods have gained encouraging results compared to traditional point cloud simplification methods.

In this work, a novel method for simplifying the Terracotta Warrior fragments is presented. Firstly, an algorithm for extracting feature points is proposed. By constructing a kd-tree, the k-nearest neighborhood of each point is established, and multi-parameters for describing the features of each point are designed and calculated. The multi-parameters include four parts (see Section 2.A for details). Based on the four parameters, the feature points and non-feature points can be extracted. Secondly, several subsets of non-feature points are generated based on uniform sampling method and the k-Nearest Neighbor (kNN) method. A deep neural network is then constructed to simplify these subsets. Finally, the simplified subsets and feature points are merged into the complete point cloud. The flowchart of this method is shown in Fig. 1.

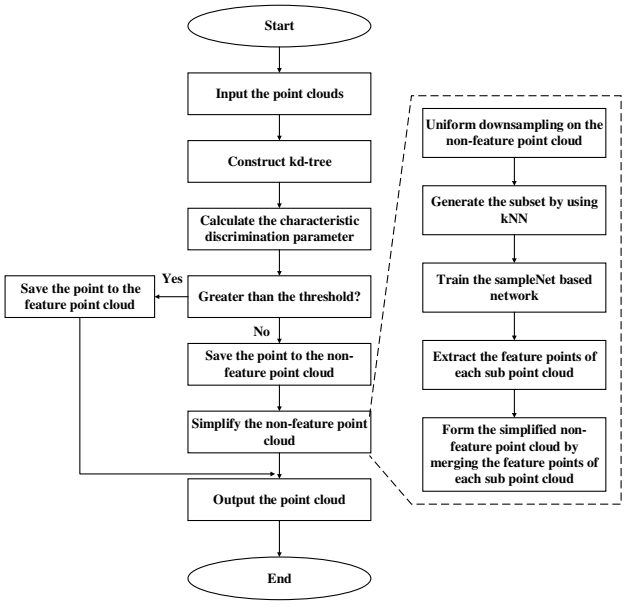

Fig. 1. Flowchart of the proposed method.

The remainder of this paper is organized as follows. In Section 2, the related theories, network architecture, and our simplification method are described. The experimental results and the contrasts with other related methods are provided in Section 3. Finally, discussion and a conclusion are given in Section 4.

2. METHODS

A. Extraction of Feature Points

In order to retain the feature points entirely and accurately, a method to measure the importance of each point is proposed. Based on the method, the feature points can be judged. Given the raw point cloud as $R = \{p_i(x_i, y_i, z_i)\}$, $(i = 1, 2, \ldots, N)$, where p_i denotes the ith point, and N denotes the size of R. As the topological relationship between the scattered points is not obvious, the method needs to construct the k neighborhood of points. At present, common methods for the kNNsearch include the spatial grid method, the octree method, and the kd-tree method. The kd-tree method can quickly establish the search path of each data point. It not only consumes less computer memory, but also has strong adaptability. After comparative considerations, the kd-tree method is used to establish the k neighborhood of points. Four characteristic parameters to reflect the importance of one point are defined: the average distance from one point to its k-nearest neighborhood points (abbreviated as AVD), the curvature and the average normal vectors deviation angles between the point and its neighboring points (abbreviated as ANVDA), and the distance from one point to the centroid of its neighborhood (abbreviated as DCON). Based on the four characteristic parameters, a feature discriminating parameter and a feature threshold are defined, and the point where the feature discriminating parameter is greater than the threshold is the feature points.

1. AVD

The AVD from point p_i to its k-neighboring points is calculated as follows:

$$d(p_i) = \frac{1}{k}\sum_{i=1}^{k} |q_j - p_i|, \tag{1}$$

where $q_j (j = 1, 2, \ldots, k)$ denotes the jth point of the k-neighboring points of point p_i. When the neighborhood points of point p_i are more densely distributed, the smaller the AVD value and the greater the probability that point p_i is a feature point and should be retained. Conversely, it should be noted that a large AVD value means that point p_i locates in a flat region. Therefore, the point with a large AVD value can be regarded as a redundant point and should be removed.

2. Curvature and ANVDA

The principal component analysis (PCA) algorithm is used to obtain a normal estimation for each point p_i in point cloud R. Based on the covariance matrix, the local surface properties of point-based surfaces can be estimated. The centroid of the neighbors q_j of point p_i is defined as follows:

$$u = \frac{1}{k}\sum_{j=1}^{k} q_j. \tag{2}$$

The 3×3 covariance matrix R is given by

$$\mathrm{M} = \frac{1}{k}\sum_{j=1}^{k}(q_j - u)(q_j - u)^{\mathrm{T}} = \frac{1}{k}\begin{bmatrix} q_1 - u \\ q_2 - u \\ \vdots \\ q_k - u \end{bmatrix}^{T}\begin{bmatrix} q_1 - u \\ q_2 - u \\ \vdots \\ q_k - u \end{bmatrix}. \tag{3}$$

Assuming λ_0, λ_1, and λ_2 are the eigenvalues of M, and $\lambda_0 \le \lambda_1 \le \lambda_2$. v_0, v_1 and v_2 are the eigenvectors corresponding to λ_0, λ_1, and λ_2, respectively, then, the eigenvector v_0 corresponding to the smallest eigenvalue λ_0 is taken as the surface normal vector n_{pi} at p_i.

The minimum eigenvalue λ_0 measures the change of the local surface, so the change of λ_0 can be used to estimate the curvature. The local curvature at p_i can be provided by Eq. (4):

$$c^i_{\mathrm{cur}} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{4}$$

The value of c^i_{cur} reflects the curve of the surface. For points belonging to an ideal plane, $\lambda_0 = 0$ and $c^i_{\mathrm{cur}} = 0$ are defined. The larger the value of c^i_{cur}, the region that the variance of the curvature changes more where p_i locates in and the greater the probability that the point p_i is a feature point.

The normal vector of the point reflects the tangent plan where the point is located. If the normal vectors of two points are the same, it implies that they locate in the same region. In contrast, if the difference between the two normal vectors is large, it means that they are more likely to be in different tangent planes. Therefore, the average of the sum deviation angles between n_{pi} and n_{qj} is denoted as follows:

$$\omega_{\mathrm{pi}} = \frac{1}{k}\sum_{j=1}^{k}\arccos\frac{n_{\mathrm{pi}} \cdot n_{qj}}{|n_{\mathrm{pi}}| \times |n_{qj}|} \quad (i = 1, 2, \ldots, N), \tag{5}$$

where $\omega_{\mathrm{pi}} \in [0, \pi]$. The larger the value of ω_{pi}, the curved surface fitted by point p_i, and the more convex its k-neighboring points, the greater the probability that p_i is a point in a sharp region.

As shown in Fig. 2, point q_j with the green color is the k-neighboring points of p_i and the vectors n denote the normal of each point, where $k = 4$. The region in which the variance of the curvature changes greatly where point p_1 locates in is sharp. The deviations of the angles between point p_1 and its k-neighboring points are large, as shown in Fig. 2(a). However, the deviations of the angles between point p_2 and its k-neighboring points are shown in Fig. 2(b). According to the deviations, the points can be clustered into sharp regional points (in red) and flat regional points (in yellow).

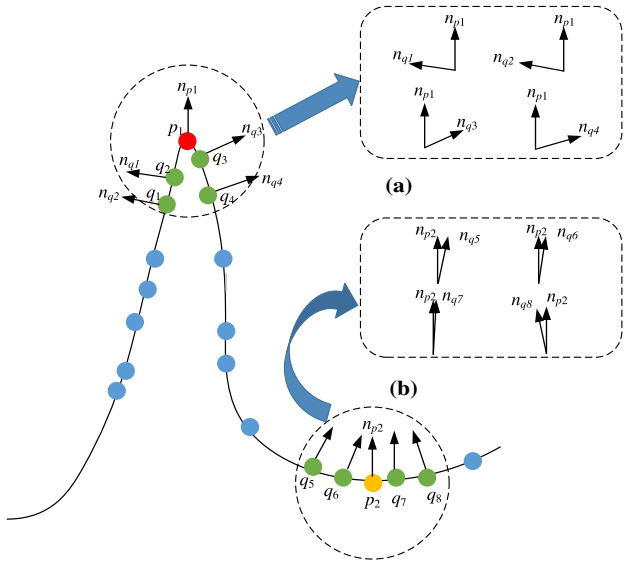

Fig. 2. Deviation angles ω_{pi} between point p_i and its k-neighboring points. (a) The deviations of the angles between the sharp point p_i and q_j. (b) The deviations of the angles between the flat point p_i and q_j.

3. DCON

The boundary points are extracted by the distance between point p_i and the centroid of its neighbors. The DCON value can be defined by Eq. (6).

$$d_{\mathrm{centroid}}(p_i) = \sqrt{(p_i - u)^2}, \tag{6}$$

where u is the centroid of the neighbors q_j of point p_i. The larger the value of $d_{\mathrm{centroid}}(p_i)$, the farther the distance between point p_i and the centroid of its neighbors. Point p_i can be judged as a boundary point. On the contrary, point p_i should be judged as an internal point.

4. Discrimination of Feature Points

In this paper, the feature points include both the sharp regional points and the boundary points, as mentioned above. The feature discriminant parameter is defined as follows:

$$f_i = \frac{\alpha \cdot c^i_{\mathrm{cur}} + \beta \cdot \omega_{\mathrm{pi}} + \gamma \cdot d_{\mathrm{centroid}}(p_i)}{d(p_i)}, \tag{7}$$

where α, β, and γ are the parameters of c^i_{cur}, ω_{pi}, and $d_{\mathrm{centroid}}(p_i)$, respectively. In order to avoid having the parameters in the feature discriminant parameters differ greatly in different models, the threshold is defined as follows:

$$F_i = \frac{1}{N}\sum_{i=1}^{N}\phi_i, \tag{8}$$

where $\phi_i = \frac{c^i_{\mathrm{cur}} + \omega_{\mathrm{pi}} + d_{\mathrm{centroid}}(p_i)}{d(p_i)}$. Finally, the discrimination rule can be defined as follows:

$$\text{class}(p_i) = \begin{cases} \text{feature point} & f_i \geq F_i \\ \text{non} - \text{feature point} & \text{otherwise} \end{cases}. \quad (9)$$

B. Network Architecture for Simplifying Non-feature Points

After the extraction of the feature points according to the discrimination rule, the non-feature points can also be extracted and retained. To simplify these non-feature points, we construct a deep neural network based on SampleNet. An overview of the network is presented in Fig. 3. First, an AE [33] network is pre-trained on the input point set $P(N, 3)$ for the task of reconstruction. Then, a smaller set of samples $Q(M_Q, 3)$ is generated via the simplified network. The simplified network includes five convolution layers, a feature-wise max-pooling layer and three fully connected layers. As Q is not a subset of P, the point in Q is projected onto P by a differentiable relaxation of the nearest neighbor selection. The projected point set $G(M_G, 3)$ is a subset of the set P, and the size of G is the same as Q. Finally, the point set G can be seen as the simplified point set of P, and the AE's loss guarantees that G is the important point that is suitable for reconstruction.

Thus, the goal of the proposed method can be concluded as follows: given a point set and a simplified size M_G, the goal of simplification is to find a subset of G^* of M_G points that minimizes the task $T's$ objective function of H, while the task T is to reconstruct a point cloud from its simplified point cloud.

$$G^* = \arg\min_G H(T(G)), \quad G \subseteq P, \quad |G| = M_G \leq N. \quad (10)$$

However, some previous works did not deal with the non-differentiability of the simplified operation. To deal with a novel differentiable relaxation for simplifying the point cloud, a soft projection operation is employed and named as soft projection.

1. Soft Projection

The soft projection strategy is used to make the generated point a subset of the input point cloud. The training process of projection is illustrated in Fig. 4. A point q_i (in orange) is projected onto its k-neighboring points from the input set P (in blue). A softly projected point g_i (in red) is obtained from a weighted average of its local neighborhood. The operation is governed by the parameter t, which is minimized during the training process to obtain the nearest neighbor point p_5 (in yellow). Instead of 3D space coordinates, the projected point is represented in the weight coordinates of its k-nearest neighborhoods in the input point cloud.

The projected point $g\,(g \in G)$ is given by a weighted average of input points from P.

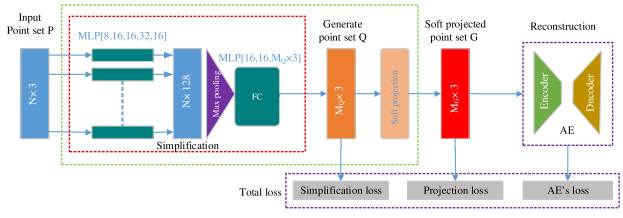

Fig. 3. Illustration of the proposed simplified method.

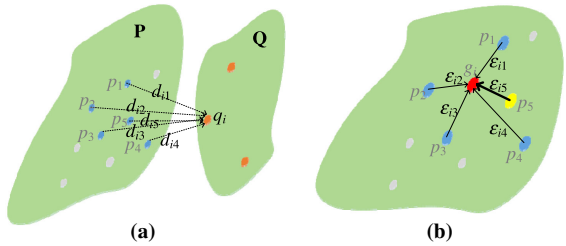

Fig. 4. Illustration of the simplified approximation. (a) Find k-neighboring points of q_i. (b) Project q_i onto local neighborhood from set P.

$$g = \sum_{i \in \text{Neig\_}p(qi)} \varepsilon_i p_i, \quad (11)$$

where Neig_$p(q_i)$ contains the indices of the k-neighboring points of q_i in P. The weights $\varepsilon_i(\alpha_i, t)$ obtained by the negative square of distance between q and its k-neighboring points, scaled with the parameter t. $\varepsilon_i(\alpha_i, t)$, is given by the following equation:

$$\varepsilon_i(\alpha_i, t) = \frac{\exp(-{\alpha_i}^2/t^2)}{\sum_{j \in \text{Neig\_}p(q)} \exp(-{\alpha_j}^2/t^2)}, \quad (12)$$

where α_i is the Euclidean distance between the point q and its k-neighboring points in set P. The distance α_i is given by Eq. (13).

$$\alpha_i = \sqrt{\sum_{i=0}^{k} (q - p_i)^2}. \quad (13)$$

Let $\varepsilon_i(\alpha_i, t)$ be a probability distribution over the points p_i, where g is the expectation value. The parameter t controls the shape of this distribution. In the limit of $t \to 0$, $\varepsilon_i(\alpha_i, t)$ will converge to a distribution centered at the nearest neighbor point, as shown in Eq. (14):

$$\lim_{t \to 0} \varepsilon_i(\alpha_i, t) = \begin{cases} 0, & \alpha_i \neq 0 \\ 1, & \alpha_i = 0 \end{cases}. \quad (14)$$

In the limit of $t \to 0$, the soft projection is replaced with sampling to obtain a sampled point cloud G^*. For each point $g^* \in G^*$, the point p_i with the largest projection weight is selected:

$$\lim_{t \to 0} \sum_{i \in \text{Neig\_}p(qi)} \varepsilon_i p_i = \arg\max_{\{pi\}} \varepsilon_i(\alpha_i, t) = g^*. \quad (15)$$

The point p_i is the nearest neighbor point of q_i. If several points in G^* might correspond to the same point p_i, the unique set of sampled points is taken. Then the method of iterative farthest point sampling (FPS), which is a sampling strategy applied in PointNet++, is used to complete it up to the size of G.

2. Loss Function

The network is trained with three terms:

$$D_{\text{total}} = D_{ae}(G) + \alpha_1 D_{\text{simply}}(Q, P) + \alpha_2 D_{\text{project}}. \quad (16)$$

The first loss term $D_{ae}(G)$ is used to optimize the approximated sampled set G to the task of reconstruction.

$D_{\text{simply}}(Q, P)$ ensures the simplified set to be close to the input and to be well spread over the input set P. Each point in Q should find its nearest points in the input point cloud. The last loss term D_{project} is used to approximate the sampling of points from the input point cloud by the soft projection operation.

Define the average nearest neighbor loss and the maximal nearest neighbor loss as follows:

$$D_{CD}(P, Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2^2, \tag{17}$$

$$D_{\max}(P, Q) = \max_{p\in P}\min_{q\in Q}\|p-q\|_2^2. \tag{18}$$

Then, simplification loss can be defined as follows:

$$D_{\text{simply}}(Q, P) = D_{CD}(Q, P) + D_{\max}(Q, P) + D_{CD}(P, Q). \tag{19}$$

The Chamfer loss is selected to construct the AE loss, which measures the squared distance between each point in point cloud G to its nearest neighbor in the other set G'.

$$D_{ae}(G, G') = \frac{1}{|G|}\sum_{g\in G}\min_{g'\in G'}\|g-g'\|_2^2 + \frac{1}{|G'|}\sum_{g'\in G'}\min_{g\in G}\|g-g'\|_2^2. \tag{20}$$

This is to drive every sample point g to be close to one point of the input set P. A projection loss is given by the following equation:

$$D_{\text{project}} = t^2. \tag{21}$$

3. EXPERIMENTS AND RESULTS

To evaluate the performance of the proposed simplification method, a series of experiments are devised and conducted. The data set includes the public point cloud data and the real-world Terracotta Warrior fragments data. It should be noted that the point cloud of the Terracotta Warrior fragments is scanned by trained students in our lab using a Creaform VIU handy scanner. The scan resolution was 3.91 mm, which favors speed but results in relatively low precision. The point clouds exhibit strong local imbalances in the sampling pattern and contain realistic noise that was the result of the scanning process. Thus, the Terracotta Warrior fragments data are denoised before using them to test the proposed method. The thickness of the fragments and the distribution of the feature points are also different. For example, armor and skirts pieces are thin-walled, while heads, hands, arms, and legs are non-thin-walled. The feature points of the head are concentrated on the eyes, nose, mouth, and ears, while the forehead, neck, and cheeks are relatively smooth with a few feature points. The armor contains straight and continuous curve features. However, the hand model is non-uniformly distributed. The back of the hand is relatively flat, and its feature points are obviously less than those of other parts. The coverage of the fragments selected in this paper is as comprehensive as possible. Some models used in this paper are shown in Fig. 5. All the experiments are conducted by a PC with the hardware of AMD Ryzen7 2700 (2.39 GHz), 16 GB memory, and NVIDIA RTX TITAN.

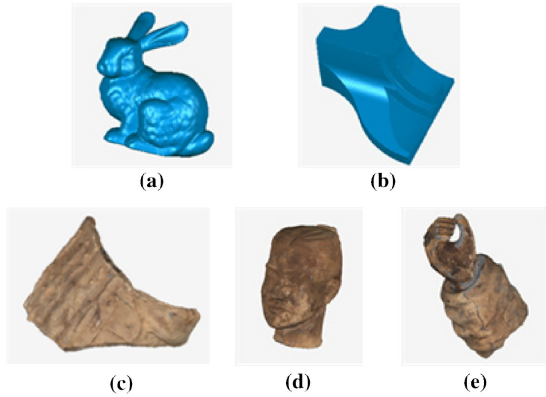

Fig. 5. Representative models used in this paper. (a) Bunny (35947 points), (b) Fandisk (53721 points), (c) G10-4-21 (36956 points), (d) G10-19-head (40859 points), and (e) G10-19-hand (26692 points).

A. Parameters Setting

1. Feature Points Extraction

The four parameters mentioned in Section 2.A are used to retain the feature points entirely and accurately. The parameter k, which determines the scale of the local regions, will be discussed. As shown in Fig. 6(a), the extracted feature points are relatively uniformly distributed, but the details of the head and ears are lost to a large extent. As shown in Fig. 6(c), there are too many points concentrated on the edges of the ears and legs, which are obviously redundant points. However, the head of the bunny does not have enough points. By comparison, we can find that the result of Fig. 6(b) is the best. The extracted feature points can well contain both sharp regional points and boundary points. Therefore, the series of the following experiments are all based on $k = 16$.

The parameters α, β, and γ are discussed as follows. A group of experiments have verified that the parameter β is more sensitive to the number of feature points. The results of different feature points corresponding to different values of parameters are shown in Table 1. In order to balance the number of point clouds after simplification, the number of retained feature points of the bunny is about 2000. $\beta = 0.4$ was chosen for this experiment. As shown in Fig. 7(a1), the distribution of the

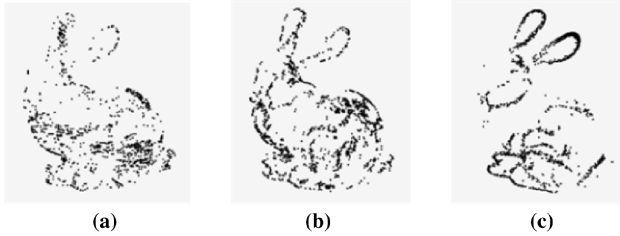

Fig. 6. Feature points extracted results of the bunny with different values of k-neighborhood. (a) $k = 8$, the number of feature points is 1479; (b) $k = 16$, the number of feature points is 1874; (c) $k = 32$, the number of feature points is 1583.

Table 1. Results of Different Feature Points under α, β, and γ for Bunny

α	β	γ	Feature Points	α	β	γ	Feature Points
10	0.35	1	108	20	0.4	3	2420
10	0.35	2	143	20	0.45	1	10100
10	0.35	3	174	20	0.45	2	10363
10	0.4	1	1718	20	0.45	3	10642
10	0.4	2	1874	30	0.35	1	497
10	0.4	3	2069	30	0.35	2	539
10	0.45	1	9484	30	0.35	3	574
10	0.45	2	9773	30	0.4	1	2718
10	0.45	1	10100	30	0.4	2	2892
20	0.35	1	277	30	0.4	3	3142
20	0.35	2	300	30	0.45	1	10653
20	0.35	3	345	30	0.45	2	10853
20	0.4	1	2340	30	0.45	3	11103
20	0.4	2	2361				

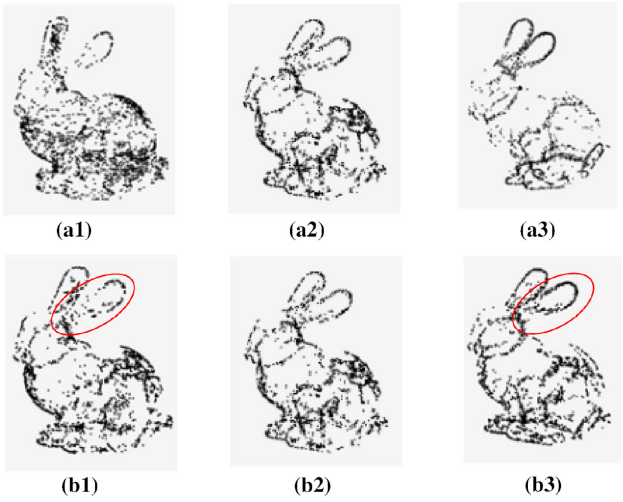

Fig. 7. Different results of feature points extraction. (a1) $\alpha = 10$, $\beta = 0.4$, and $\gamma = 2$; the number of feature points is 2413. (a2) $\alpha = 20$, $\beta = 0.4$, and $\gamma = 2$; the number of feature points is 2361. (a3) $\alpha = 30$, $\beta = 0.4$, and $\gamma = 2$; the number of feature points is 2272. (b1) $\alpha = 20$, $\beta = 0.4$, and $\gamma = 1$; the number of feature points is 2340. (b2) $\alpha = 20$, $\beta = 0.4$, and $\gamma = 2$; the number of feature points is 2361. (b3) $\alpha = 20$, $\beta = 0.4$, and $\gamma = 3$; the number of feature points is 2420.

extracted feature points is relatively scattered when α takes a small value. The feature points are retained in some relatively smooth areas like the body of the bunny. On the contrary, there are fewer feature points on the parts of the head and right ear. As can be seen from Fig. 7(a3), the feature points retained on the head of the bunny are few. There are lots of redundant points on the contour of the ear. Comparing the three results in the upper row of Fig. 7, the result of Fig. 7(a2) is the best. So, we choose $\alpha = 20$. Parameter γ is used to control the extraction of edge points. As shown in Figs. 7(b1) and 7(b3), the feature points on the ears are either sparse or redundant. Therefore, $\gamma = 2$ is the right choice. Larger or smaller values of γ results have negative accuracy effects. Therefore, $\alpha = 20$, $\beta = 0.4$, and $\gamma = 2$ are chosen.

Table 2. Number of Feature Points for Different Models

Models	Total Number	β	Feature Points
Fandisk	53721	0.25	5523
G10-4-21	36956	0.32	4153
G10-19-hand	26692	0.32	3099
G10-19-head	40859	0.31	3412

According to the actual application project and experience, the algorithm parameters have been set as follows: $\alpha = 20$, $\beta = 0.4$, and $\gamma = 2$ are chosen. In the same way, the parameters for the other models are shown in Table 2. If the values of α, β, and γ in other models are exactly the same as those of the bunny, the proportion of feature points will be greatly different. To achieve a balance, the parameters α and γ are fixed, and the parameter β is fine-tuned.

2. Non-feature Points Simplification

For the non-feature points, the number of centroid points determines the simplification rate. The process of simplifying non-feature points is a self-training process. The five models referenced in Fig. 5 are input into the network, respectively. The neighborhood size group is the number of neighbors in P of a point $q \in Q$, on which q is softly projected. The parameter controls the local context in which q searches for an optimal point to simplify. As shown in Figs. 8(a) and 8(c), the reconstruction results have more holes in the bunny's ears. Figure 8(b) shows the best result with group = 8. The overall process of generating the data sets is as follows.

First, centroid points are obtained from the uniform sampling of the non-feature points by using Geomagic software (Geomagic Wrap). The number of centroid points for each model is defined as $N_{\text{non-cen}}$. Second, each centroid point can generate a subset by its 256 neighbors. It can obtain local features from the subset. Hence, the size of the non-feature points of each model is $N_{\text{non-cen}} \times 256 \times 3$. Each model is split into 85%–5%–10% for train-validation-test sets. The AE was trained to receive and reconstruct point clouds of 256 points. During the training of the network, the Adam optimizer with a learning rate of 0.0005 and mini-batches of 50 shapes is used, and the soft projection weights are computed with group = 8. The regularization weights in Eq. (16) are set to $\alpha_1 = 0.01$, $\alpha_2 = 0.0001$. The number of total epochs is set to 400, and the code is implemented with TensorFlow.

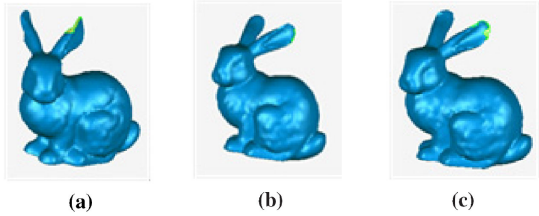

Fig. 8. Result of different neighborhood sizes. (a) group = 4, the number of points is 7098. (b) group = 8, the number of points is 6727. (c) group = 16, the number of points is 8580.

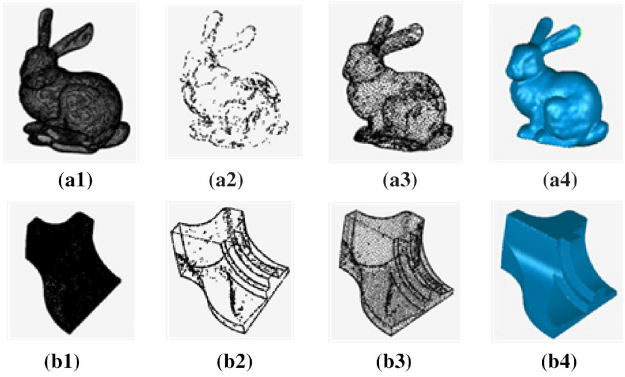

Fig. 9. Simplified results. (a1)–(a4) are the results of the bunny model, and the number of points of the simplified point cloud is 6885. (b1)–(b4) are the results of the Fandisk model, and the number of points of the simplified point cloud is 12362.

B. Simplification Results

Figure 9 shows the simplified results. In Fig. 9(a2), the areas of the ears, neck, mouth, and feet are dense areas of feature points. The feature points are well maintained. However, the body of the bunny is a flat region. The non-feature points can be well excluded. To reduce the holes that appear in the reconstructing point cloud, the total number of simplified results should be merged feature points with non-feature points, which is shown in Fig. 9(a3). It retains more points at the local detail feature parts, while fewer points are retained at the smooth position. In order to verify the validity of the algorithm, Geomagic software is used to fit the results of the point cloud simplification. The Fandisk model has some sharp edges and the normal vector changes greatly. The simplified result of the Fandisk model is shown in Fig. 9(b3). The contour points and some sharp regional points are well detected, and the sharp edges of the model are well preserved, while the sparse points in the flat areas are uniformly distributed.

The above two experimental results indicate that the proposed method has a good simplification effect in both flat- and high-curvature areas. The method is applied to the models of the Terracotta Warrior fragments as well. The results shown in Fig. 8 are the fragments of G10-4-21and G10-19-head. As a result, the number of the points in model of G10-4-21 is simplified to 7355, while that in G10-19-head is reduced to 7889. The simplification rates are 80.1% and 80.7%, respectively. From the zoom-in view of the simplified results in Figs. 10(a4) and 10(b4), it can be seen that the proposed method has achieved better results on the Terracotta Warrior fragments.

C. Comparison with Other Methods

In order to verify the feasibility and effectiveness of the simplification method proposed in this paper, we conduct two groups of experiments. The first group of experiments compares the proposed feature point extraction method with the traditional methods in Refs. [6,7]. The proposed simplification method is compared with two classical simplified algorithms, which are the k-means clustering simplification method [8] and the uniform simplification method [4] in the second group.

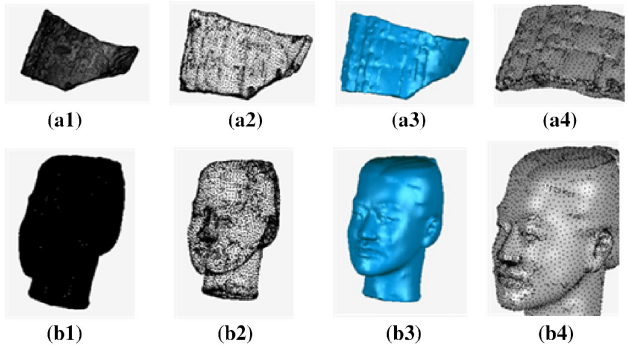

Fig. 10. Results of several Terracotta Warrior fragments.

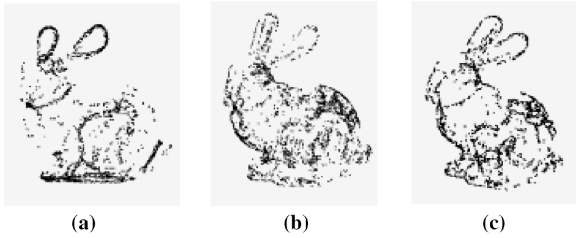

Fig. 11. Feature points extraction results of different methods. (a) The feature points extraction method in Ref. [6], the number of feature points is 2387. (b) The feature points extraction method in Ref. [7], the number of feature points is 2838. (c) The proposed feature points extraction method, the number of feature points is 2361.

The comparison results of different feature points extraction methods are shown in Fig. 11. As shown in Fig. 11(a), the method [6] only retains the contour points. The ears and legs leave too many points to be redundant. However, flat regions like the body and the head of the bunny have too few points and some features are blurred. Figure 11(b) retains more points at the smooth areas, such as the back. At the same time, the number of feature points on the contour of the right ear is significantly less. Holes appear in ears during point clouds reconstruction. Figure 11(c) shows that the ears, mouth, and feet of the bunny are all dense feature points, and these feature points are detected and retained. While in the body of the bunny, some points are retained to highlight the features of the model. The compared results show that the feature points extraction method in this paper is significantly better than the methods in Refs. [6,7].

The second group of comparative experiments is mainly based on the real-world Terracotta Warrior fragments data. For the fairness of the experiments, the simplification rate of each algorithm is set to around 80% by adjusting the relevant parameters.

Figure 12 shows the different simplified and reconstruction results of the G10-19-head and the G10-19-hand. Compared with Figs. 12(a3) and 12(a4), Fig. 12(a2) has a clearer contour on the narrow feature positions, such as the ears, eyes, nose, and mouth, and the redundant points in the forehead and cheeks are removed. In Fig. 12(a3), there are many points in the high-curvature regions, e.g. the nose and ear. However, the k-means clustering simplification method ignores the points with low curvature. It can be seen from Fig. 12(a4) that the result does not

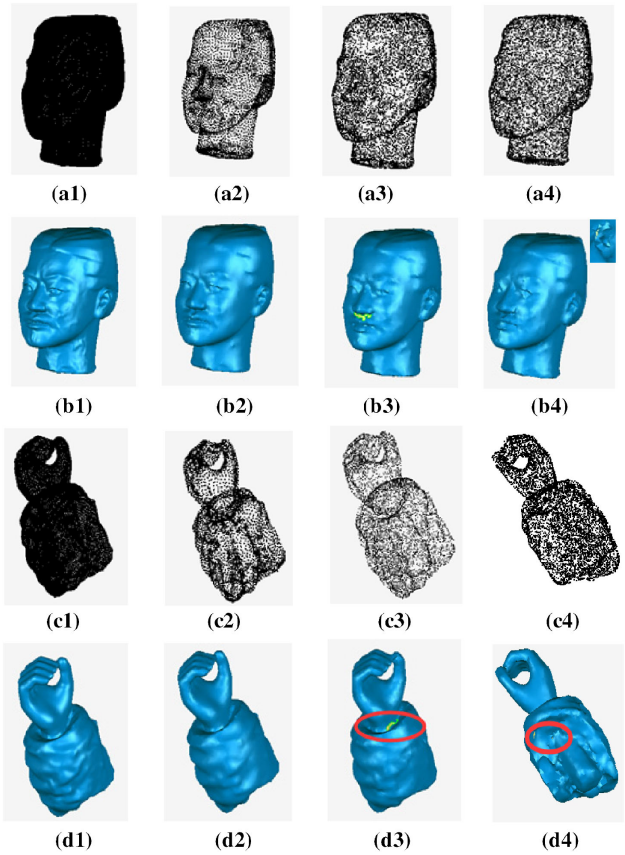

Fig. 12. Simplified and reconstruction results of G10-19-head and G10-19-hand. (a1) and (c1) are the raw point cloud, the number of points is 40859 and 26692; (a2) and (c2) are our simplification results, with the number of points of 7889 and 6586; (a3) and (c3) are the k-means clustering simplification method results, with the number of points of 7890 and 6593; (a4) and (c4) are uniform simplification method results, with the number of points of 7927 and 6580; and (b1)–(b4) and (d1)–(d4) are the reconstruction results.

consider any feature points; it just deletes certain ratio points evenly and ignores the curvature and density of the points. As a result, the detail features are seriously lost. The simplification result is worse. As shown in Figs. 12(b3) and 12(b4), the reconstruction results of the two methods both have some holes. For example, there are some holes at the area of the Terracotta Warrior's nose and right ear. From the Fig. 12(b3), the narrow feature location such as the nose and mouth fit incompletely. In Fig. 12(b4), the eyes of the model are blurred. Figure 12(b2) has the more completed and smoother surfaces. As to the simplified and reconstruction results of the G10-19-hand, the simplification rates of the three methods are all 75.3%. In Fig. 12(d3), we can see holes in the wrist of the hand model. And in the same area in Fig. 12(d4), uniform simplification has bad fitting effects. Compared with the other two methods, the proposed method has a clearer contour on the feature position.

To evaluate the accuracy of the simplified point clouds, the geometric error between the original and simplified point clouds should be measured. The quantitative errors are shown in Table 3, where Δ max and Δavg represent the maximum error and average error of the geometry that was proposed by Shi *et al.* [9]. The method proposed in this paper makes the difference between the simplified and the original point clouds smaller, and better preserves the characteristics of the models.

4. DISCUSSION AND CONCLUSION

Point cloud simplification plays a very important role in the process of virtual cultural relics protection, which can reduce the time spent on subsequent tasks while reducing data storage space. One of the most important principles of point cloud simplification is to reduce the number of points as much as possible without affecting the reconstruction effect. In this research, a novel robust point cloud simplification method for cultural relics, especially 3D Terracotta Warrior fragments, is developed that can hold the geometric features and the shape of the potential surface. The main contributions are as follows: (1) an algorithm for extracting feature points and non-feature points is proposed, which is based on local structure; (2) a deep neural network is constructed to reduce the number of non-feature points; and (3) the proposed method can effectively reduce the scale of the point cloud and is applied to the simplification of the terracotta point cloud for the first time.

We have conducted a series of simulations and real data-based experiments, and the results show that the proposed method can simplify the point cloud more effectively than other methods. Compared to traditional methods, one of the main advantages is that the deep neural network can effectively learn the local features of the point cloud. However, there are still some shortcomings in this method. First, the parameters for feature points extraction are not adaptive. These parameters are determined by experience, which wastes a lot of time for manually adjusting the parameters. As for future work, the self-adaptive parameters should be developed to retain more suitable descriptive feature points. Second, the kNN used in the processing of feature points

Table 3. Simplification Results Comparison

Models		G10-19-head	G10-19-hand
Original points		40859	26692
Simplified points		7889	6586
K-means clustering simplification method [8]	Δ max (mm)	0.9114×10^{-3}	0.1135×10^{-2}
	Δ avg (mm)	0.2249×10^{-3}	0.2011×10^{-3}
Uniform simplification method [4]	Δ max (mm)	0.1309×10^{-2}	0.1051×10^{-2}
	Δ avg (mm)	0.2409×10^{-3}	0.1992×10^{-3}
Our method	Δ max (mm)	0.85794×10^{-3}	0.7432×10^{-3}
	Δ avg (mm)	0.20156×10^{-3}	0.1979×10^{-3}

and non-feature points cannot capture wider context information for each point. Learning more about richer structural features is also a future research direction.

In summary, the method proposed in this paper can simplify the point cloud effectively. The results of the first application on the 3D Terracotta Warrior fragments have demonstrated the effectiveness and practicability of the proposed method. We hope this work can provide a useful data preprocessing tool for archaeological work.

Funding. National Natural Science Foundation of China (61701403, 61673319, 61731015); China Postdoctoral Science Foundation (2018M643719); Young Talent Support Program of the Shaanxi Association for Science and Technology (20190107); National Key Research and Development Program of China (2017YFB1402103); Shaanxi Province Industrial Innovation Chain Project (2017ZDCXL-GY-03-01-01); Key R&D Projects in Shaanxi Province (2019ZDLGY10-01, 2019ZDLSF07-02); Scientific Research Program Funded by Shaanxi Provincial Education Department (18JK0767).

Disclosures. The authors declare no conflicts of interest.

REFERENCES

1. H. Gao and G. Geng, "Classification of 3D terracotta warrior fragments based on deep learning and template guidance," IEEE Access **8**, 4086–4098 (2020).
2. Y. Zhang, K. Li, X. Chen, S. Zhang, and G. Geng, "A multi feature fusion method for reassembly of 3D cultural heritage artifacts," J. Cult. Herit. **33**, 191–200 (2018).
3. N. Wei, T. Xu, K. Gao, and F. Dong, "Mesh simplification weighted by Voronoi poles feature computed saliency," J. Graph. **38**, 314–319 (2017).
4. R. Li, M. Yang, Y. Liu, and Y. Zhang, "A uniform simplification algorithm for scattered point cloud," Acta Opt. Sin. **37**, 1–9 (2017).
5. F. Sun, Y. Choi, Y. Yu, and W. Wang, "Medial meshes-a compact and accurate representation of medial axis transform," IEEE Trans. Vis. Comput. Graphics **22**, 1278–1290 (2015).
6. W. Huang, X. Peng, P. Wen, and X. Wu, "Simplification of scattered point cloud with geometric feature reservation," Comput. Eng. Appl. **45**, 168–170 (2009).
7. L. Wang and B. Yuan, "Feature point detection for 3D scattered point cloud model," Signal Process. **27**, 932–938 (2011).
8. J. Chang, L. Zhao, and H. Wang, "Research on k-means clustering point cloud reduction algorithm based on boundary reservation," Eng. Surv. Mapp. **027**, 60–65 (2018).
9. B. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using k-means clustering," Comput. Aided Des. **43**, 910–922 (2011).
10. Y. Chen and L. Yue, "A method for dynamic simplification of massive point cloud," in *IEEE International Conference on Industrial Technology (ICIT)* (IEEE, 2016), pp. 1690–1693.
11. C. Ji, Y. Li, J. Fan, and S. Lan, "A novel simplification method for 3D geometric point cloud based on the importance of point," IEEE Access **7**, 129029 (2019).
12. T. Li, Q. Pan, L. Gao, and P. Li, "A novel simplification method of point cloud with directed Hausdorff distance," in *IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (IEEE, 2017), pp. 469–474.
13. L. Chen, Y. Cai, and J. Zhang, "Feature point extraction of scattered point cloud based on multiple parameters hybridization method," Appl. Res. Comput. **34**, 2867–2870 (2017).
14. V. Markovic, Z. Jakovljevic, and Z. Miljkovic, "Feature sensitive three-dimensional point cloud simplification using support vector regression," Teh. Vjesn. **26**, 985–994 (2019).
15. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 652–660.
16. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems* (2017), pp. 5099–5108.
17. J. Li, B. M. Chen, and G. Hee Lee, "So-Net: self-organizing network for point cloud analysis," in *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 9397–9406.
18. H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: enhancing local neighborhood features for point cloud processing," in *IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 5565–5573.
19. Z. Zhang, B. Hua, and S. Yeung, "ShellNet: efficient point cloud convolutional neural networks using concentric shells statistics," in *IEEE International Conference on Computer Vision* (2019), pp. 1607–1616.
20. Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: point cloud auto-encoder via deep grid deformation," in *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 206–215.
21. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," ACM Trans. Graph. **38**, 146 (2019).
22. Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4548–4557.
23. C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *European Conference on Computer Vision (ECCV)* (2018), pp. 52–66.
24. C. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).
25. B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 984–993.
26. H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "SplatNet: sparse lattice networks for point cloud processing," in *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2530–2539.
27. H. Lei, N. Akhtar, and A. Mian, "Octree guided CNN with spherical kernels for 3D point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 9631–9640.
28. A. Komarichev, Z. Zhong, and J. Hua, "A-CNN: annularly convolutional neural networks on point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 7421–7430.
29. S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using geo-CNN," in *IEEE/CVF Conference on Computer Vision & Pattern Recognition* (2019).
30. H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: flexible and deformable convolution for point clouds," in *IEEE International Conference on Computer Vision* (2019), pp. 6411–6420.
31. O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).
32. I. Lang, A. Manor, and S. Avidan, "SampleNet: differentiable point cloud sampling," arXiv:1912.03663 (2019).
33. P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," arXiv:1707.02392 (2017).