RAIRO-Oper. Res. 53 (2019) 487–504
https://doi.org/10.1051/ro/2018082

RAIRO Operations Research
www.rairo-ro.org

AN EFFICIENT SIMPLIFICATION METHOD FOR POINT CLOUD BASED ON SALIENT REGIONS DETECTION

Abdul Rahman El Sayed[1], Abdallah El Chakik[2,*], Hassan Alabboud[3] and Adnan Yassine[1,4]

Abstract. Many computer vision approaches for point clouds processing consider 3D simplification as an important preprocessing phase. On the other hand, the big amount of point cloud data that describe a 3D object require excessively a large storage and long processing time. In this paper, we present an efficient simplification method for 3D point clouds using weighted graphs representation that optimizes the point clouds and maintain the characteristics of the initial data. This method detects the features regions that describe the geometry of the surface. These features regions are detected using the saliency degree of vertices. Then, we define features points in each feature region and remove redundant vertices. Finally, we will show the robustness of our method *via* different experimental results. Moreover, we will study the stability of our method according to noise.

Mathematics Subject Classification. 46N10, 62H35, 65D18.

Received October 23, 2017. Accepted September 19, 2018.

1. Introduction

3D content has an important role in many domains such as computer vision, architectural and industrial design, scientific visualization, animation films, and medical imaging. However, the performance of 3D scanning devices [14] has been enhanced year by year and modern scanners generate complicated and dense sampled point clouds with considerable redundancy. These point clouds are then converted into continuous surface representation such as polygonal meshes for further processing using reconstruction algorithms. Thus, the redundancy of point clouds must be removed to minimize the complexity of reconstruction algorithms. Point clouds simplification is considered as the process of eliminating duplicated and redundant points that does not affect the characteristics of the initial form [12]. Point clouds simplification algorithms can be classified into clustering, coarse-to-fine, and iterative methods.

The first category (clustering methods) covers algorithms that aim to subdivide the input point clouds into several surface patches (group of points), based on a certain criterion, then replace every patch by its representative point (the centroid for example). In this context, Pauly *et al.* [23] proposed two different strategies, inspired by the methods of mesh simplification [4, 27], for the construction of patches. The first is incremental

Keywords. 3D point clouds simplification, 3D point clouds segmentation, 3D object processing, points clouds, saliency.

[1] Normandie Univ., UNIHAVRE, LMAH, FR CNRS 3335, ISCN, 76600 Le Havre, France.
[2] Beirut Arab university, Tripoli, Lebanon.
[3] Business and Administration Faculty, Lebanese university, Tripoli, Lebanon.
[4] Normandie Université, UNIHAVRE, ISEL, 76600 Le Havre, France.
*Corresponding author: a.alshakik@bau.edu.lb

Article published by EDP Sciences

and uses an algorithm of regions growth. The second is hierarchical, and bases itself on a binary partition of the space. The number, as well as the size of patches, are controlled by means of the curvature. The authors mention that the methods of clustering are fast and effective regarding the resource memory, but leads to point clouds with significant quadratic error.

Coarse-to-fine approaches extract randomly a set of points from the original point cloud then define implicitly a distance function using a 3D Voronoï diagram. This distance function is then used to refine the point cloud [20].

Iterative methods correspond to algorithms whose objective is to iteratively reduce the number of points of the original point clouds by using a decimation operator as the one proposed in [31]. In these methods, each point of the input cloud is assigned by a weight quantifying its importance among its direct neighbors. Then, points with the lowest weights are removed, and the list of neighbors of the remaining points and their respective weights are updated. This last step is repeated until the number of points set by the user is reached.

In this paper, we propose to divide point cloud into patches and compute a saliency map in order to cluster the point cloud into set of feature regions. Then the non-features points are removed to refine the point cloud.

1.1. State-of-the-art

In this subsection, we discuss some available approaches for 3D point cloud and mesh simplification by mention their defects.

Lee *et al.* [11] introduced a simplification method based on the geometry information and points normal using 3D grids. Pauly *et al.* [23] presented and studied different approaches for surface simplification of 3D objects from unstructured point clouds. In [19,20], Moaning and Dodgson used the idea of progressive intrinsic farthest point sampling of a surface in point clouds. They presented a uniform simplification coarse-to-fine algorithm with user-controlled density.

In order to compute the weight of 3D models features, Lee *et al.* [13] suggested a novel simplification method by adopting the Discrete Shape Operator. Based on features extraction, Peng *et al.* [24] proposed a new simplification algorithm for unstructured point clouds described by its unit normal vectors.

In [30], Song *et al.* proposed a global clustering simplification method for point cloud. It consists of finding a subset of the original input data set according to a specified data reduction ratio. Then, they obtained a global optimal result by minimizing the geometric deviation between the input point sets and the simplified ones. The drawbacks of this method is the time complexity and the approximated point-to-surface distances may give non-accurate values.

In [18], Miao *et al.* proposed a curvature-aware adaptive re-sampling method to simplify point clouds based on an adaptive mean-shift clustering pattern. The usage of adaptive mean-shift clustering is to generate a simplification non-uniformly distributed result. However, it is difficult to incorporate the simplified geometric error in their approach and the simplification rate is low.

Finally, in [28], Shi *et al.* presented a new method for simplification "cluster subdivision" based on k-means clustering approach according to two factors: user-defined space interval and normal vector tolerance. Thus, this method is very sensitive to user-defined parameters

1.2. Contributions

In this method, the point cloud is represented by a weighted graph, and the saliency degree of vertices is calculated. Then, the graph is segmented into feature regions that maintain the characteristics of the surface shape where each region is a set of similar vertices in term of saliency degree. Secondly, we define the set of interest points located on the boundary of each region by detecting collinear points. Moreover, we define the set of interest point located into region according to a saliency threshold value. Finally, we maintain the interest points and remove non-featured points. Our method is illustrated in the flowchart (Fig. 1).

1.3. Paper organization

The remainder of this paper is organized as follows. Section 2 presents the 3D point clouds surface modeling, Section 3 details the approach used in our proposed method to compute Saliency. Section 4 presents the Feature

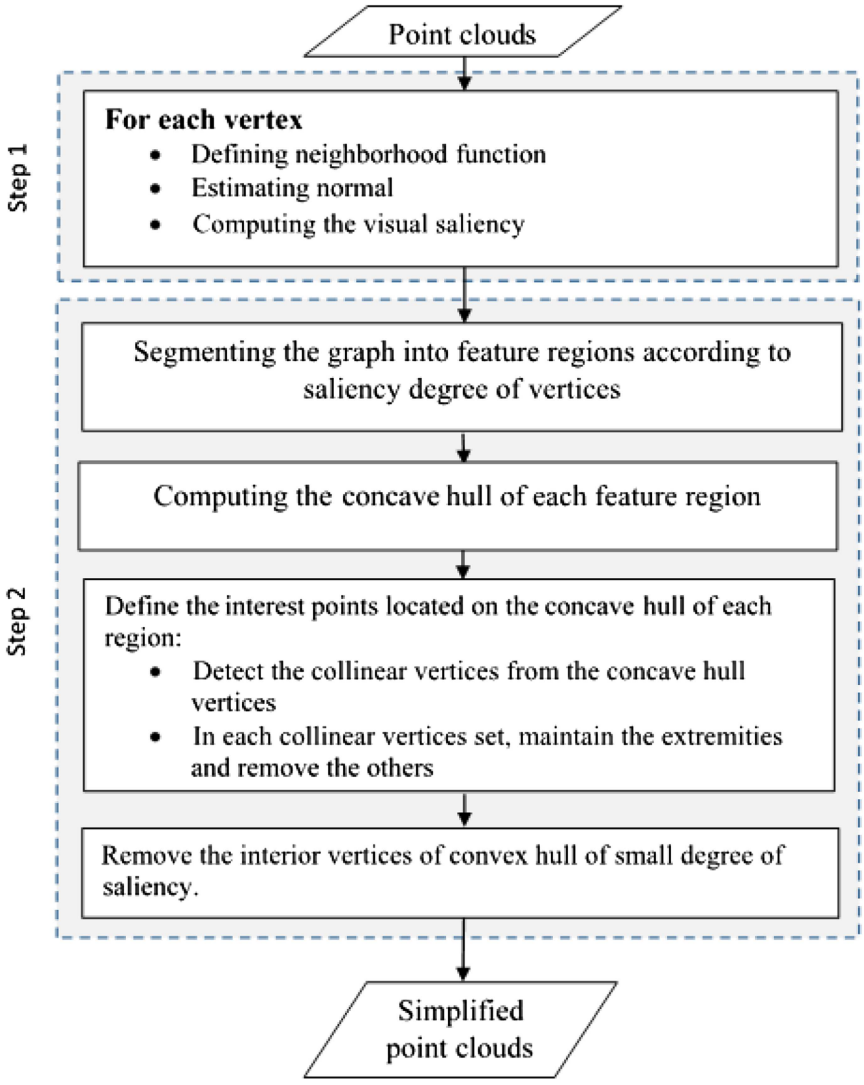

FIGURE 1. Flowchart of our method.

regions detection and Feature points detection process. Section 5 shows some experimental results. Section 6 presents an evaluation of our method. Finally, Section 7 concludes this paper.

2. SURFACE MODELING

An undirected weighted graph $G = (V, E, W)$ consists of a finite set of vertices V, a finite set of edges $E \subset V \times V$, and an edge weight function $W : V \times V \to [0, 1]$ that represents the similarity measure between two adjacent vertices. Let (v_i, v_j) be the edge that connects two vertices v_i and v_j. Let us consider the general situation where a point cloud can be viewed as a weighted graph $G = (V, E, W)$. A vertex v_i is represented by its 3D coordinates $v_i = (x_i, y_i, z_i)^T$, its normal vector $N(v_i)$, and the directional vectors $x(v_i)$ and $y(v_i)$ that correspond to the 2D tangent plan estimated at vertex v_i on the point clouds. Each edge E has a weight ω associated to compute saliency.

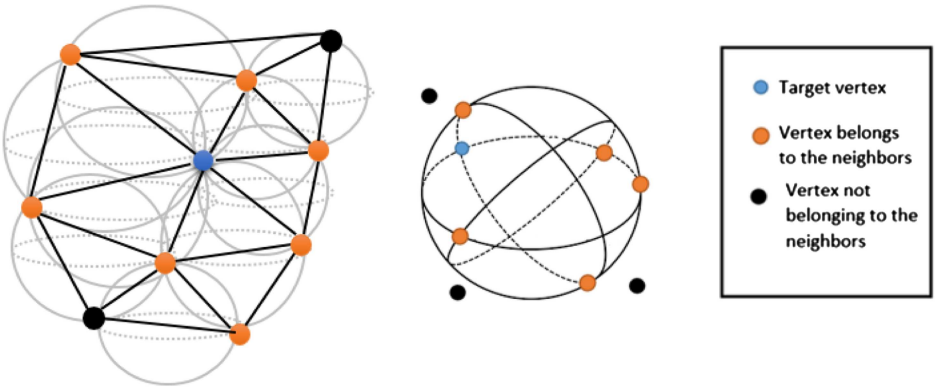

FIGURE 2. Illustration of a vertex neighboring.

2.1. Vertex neighbors

Based on the Delaunay triangulation approach of a given input point cloud P, we define the set of neighbors $D_n(v_i)$ for each vertex v_i by considering the set vertices located on the borders of circumscribed spheres as shown in Figure 2. In other word, $D_n(v_i)$ is defined as the corners (vertices) of triangles sharing v_i as shown in the following figure.

2.2. Normal estimation

To compute the normal $N(v_i)$ of a vertex v_i and the 2-directional vectors following the x- and y-axes, we use the Eigen-values of the covariance matrix $C(v_i)$ as proposed by [5]. To do so, we compute the gravity center and the associated covariance matrix at a vertex v_i.

- The gravity center is defined as:

$$\breve{v}_i = \frac{1}{\text{Card}(D_n(v_i))} \sum_{j \in D_n(v_i)} v_j \tag{2.1}$$

- Then, the covariance matrix is defined as:

$$C(v_i) = \sum_{j \in D_n(v_i)} (v_j - \breve{v}_i)(v_j - \breve{v}_i)^T \in \mathbb{R}^{3\times 3}. \tag{2.2}$$

Moreover, $c_0(v_i)$ and $c_1(v_i)$ that estimate receptively the minor and major principal direction are computed using the covariance matrix.

3. SALIENCY DETECTION

In this section, we present our proposed method to compute the saliency degree of a given 3D point cloud vertices. Firstly, we start by introducing the local patch descriptor for each vertex, then we define vertices deviation factors to calculate the saliency degree of these vertices.

3.1. Local patch construction

To define a patch for a given vertex v_i, we construct a square grid centered at v_i according to its tangent plane as shown in Figure 3. The patch length $L(v_i)$ is then defined as [16]:

$$L(v_i) = \max_{v_j \in D_n(v_i)} (||v_j - v_i||_2^2), \tag{3.1}$$

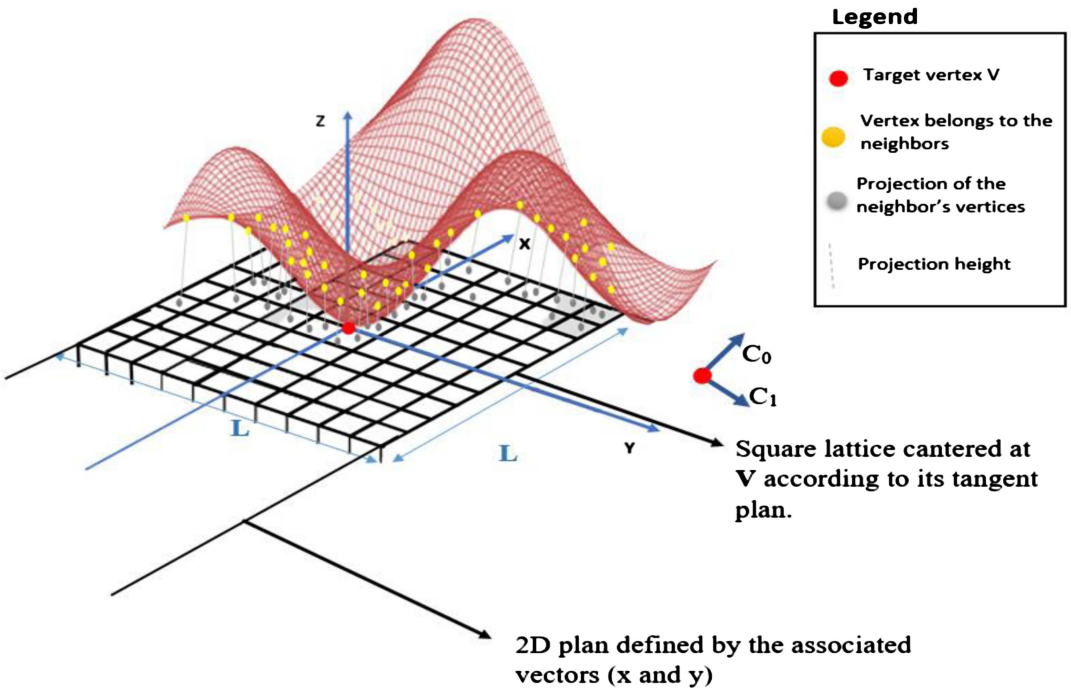

FIGURE 3. Illustration of the patch construction and vertices projection. $c_0(v)$ and $c_1(v)$ are the principal directions at v. L is the patch length.

where $||.||_2^2$ represents the Euclidean norm.

Then, a square lattice of n^2 cells is constructed with respect to the basis obtained from $c_0(v_i)$ and $c_1(v_i)$. The side length of each cell in the lattice is then $L(v_i)/n$. Finally, all vertices $v_j \in D_n(v_i)$ are projected on the tangent plane of v_i producing the set of projected vertices v'_j. Each cell of the patch is then filled with the average values of deviation factor of each vertex v_j according to v_i where is projected in the same cell.

3.2. Deviation factor

In order to measure the deviation factor between two vertices v_i and v_j, we compute the angle between their normal vectors shown in Figure 4 as following:

$$\propto_{(v_i,v_j)} = \arccos\left(\frac{||N(v_i)\cdot N(v_j)||_2^2}{\sqrt{(\sum_{k=1}^{3} N(v_i)(k) * N(v_i)(k)) * \sum_{m=1}^{3} N(v_j)(m) * N(v_j)(m)}}\right) \tag{3.2}$$

where v_i and v_j are two vertices. Then, we define the deviation factor between two vertices using the following equation:

$$\mathrm{DF}(v_i, v_j) = \frac{\propto (v_i, v_j)}{360} \times \lambda. \tag{3.3}$$

The deviation factor measures the percentage of deviation between two vertices where the maximal deviation is 360°. The constant λ can be considered hundred in order to bound the value of DF to this value. Note that if the value of the deviation factor between two vertices is almost zero, these two vertices are considered located in the same plane.

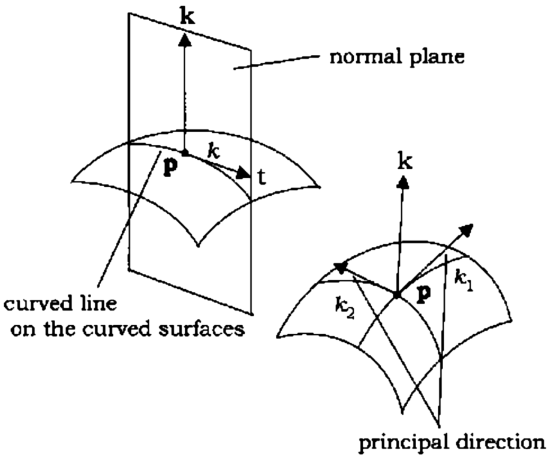

FIGURE 4. Vertices normal vectors.

3.3. Saliency computation

To compute the saliency at a vertex v_i, we measure the similarity between its local patch $P(v_i)$ and the patches associated to its neighbors. To do so, the saliency at a vertex v_i is calculated according to its patch saliency.

The weight function ω that represents a similarity measure between a vertex and its neighbors is defined as:

$$\omega(v_i, v_j) = \mathrm{e}^{-\frac{K(v_j)*|A(v_j)-A(v_i)|}{\sigma(v_j)*\propto(v_i,v_j)}}, \tag{3.4}$$

where $k(v_j)$ represents the mean curvature [9] at v_j, $A(v_j)$ is the average of the local patch descriptor *cells* elements of v_j, and $\sigma\,(v_i)$ is the scale parameter and can be calculated as:

$$\sigma\,(v_i) = \max_{v_k \in D_n(v_i)} (||v_k - v_i\,||_2). \tag{3.5}$$

The saliency degree of a given vertex is calculated according to its patch. Therefore, we propose to compute the patch saliency in term of the entropy of vertices and their neighbors. To do so, we define the probability $P_r(v_i)$ as:

$$P_r(v_i) = C_x^i/|\mathrm{V}|, \tag{3.6}$$

where C_x^i represent the number of $P(v_i)$ vertices having a deviation factor with v_i greater than x (x is a constant related to applications) and $|V|$ is the number of all $P(v_i)$ vertices as shown in Figure 5.

Then, the entropy that measures the dissimilarity of $P(v_i)$ is computed as following:

$$\mathrm{entropy}\,(v_i) = -P_r(v_i) \times \log_2 P_r(v_i). \tag{3.7}$$

The saliency of $P(v_i)$ is calculated as following:

$$\mathrm{Patch\_Saliency}(v_i) = \frac{\sum_{j\in|V|}\,|(\mathrm{entropy}(v_j) - \mathrm{entropy}(v_i))| \times L(v_j)}{|\mathrm{Patches}|}, \tag{3.8}$$

where |Paches| is the number of patches of v_i neighbors, and $L(v_j)$ is the Euclidian distance between the vertex v_j and its 2D projection on the tangent plane at vertex v_j.

Finally, the visual saliency SL at a vertex v_i is defined as following:

$$\mathrm{SL}(v_i) = \frac{\sum_{j\in|V|}\,\mathrm{Patch\_Saliency}\,(v_j) \times \omega_(v_i, v_j)}{|V|}. \tag{3.9}$$

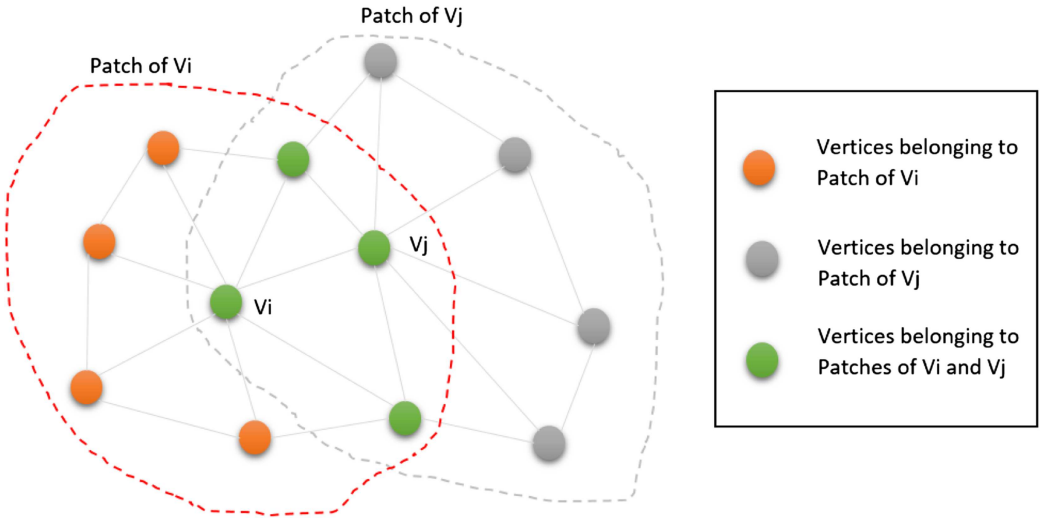

FIGURE 5. Vertices patches.

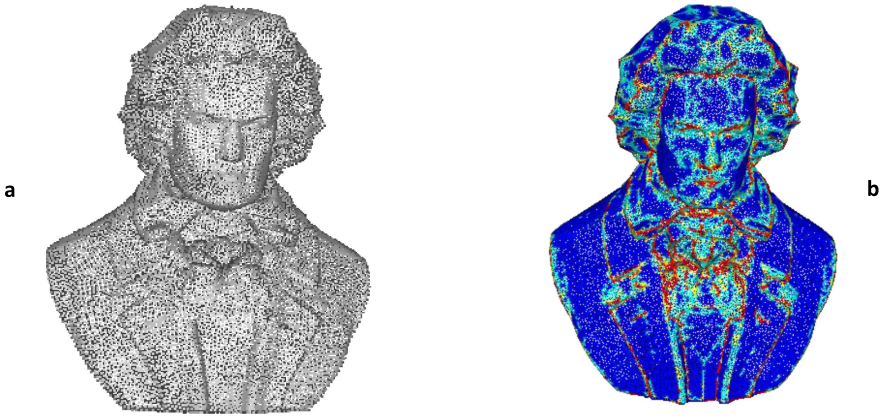

FIGURE 6. *Panel a*: the original 3D point clouds king_head. *Panel b*: displays its saliency using our method. The red areas on the 3D point clouds object represent the most salient regions and the blue areas are vertices of low saliency.

The saliency degree is bounded between [0, 1] where the degree 1 means that v_i is very similar to its neighborhood and 0 means the opposite (Fig. 6).

4. FEATURE REGIONS DETECTION

In this section, we present a method to detect feature regions followed by detecting feature points on a 3D point cloud using the saliency degree of vertices.

4.1. Segmentation

In order to detect the feature regions we firstly segment the 3D point cloud according to saliency degree of vertices using custom version of connected component labeling algorithm (Algorithm 1). Then, we define the

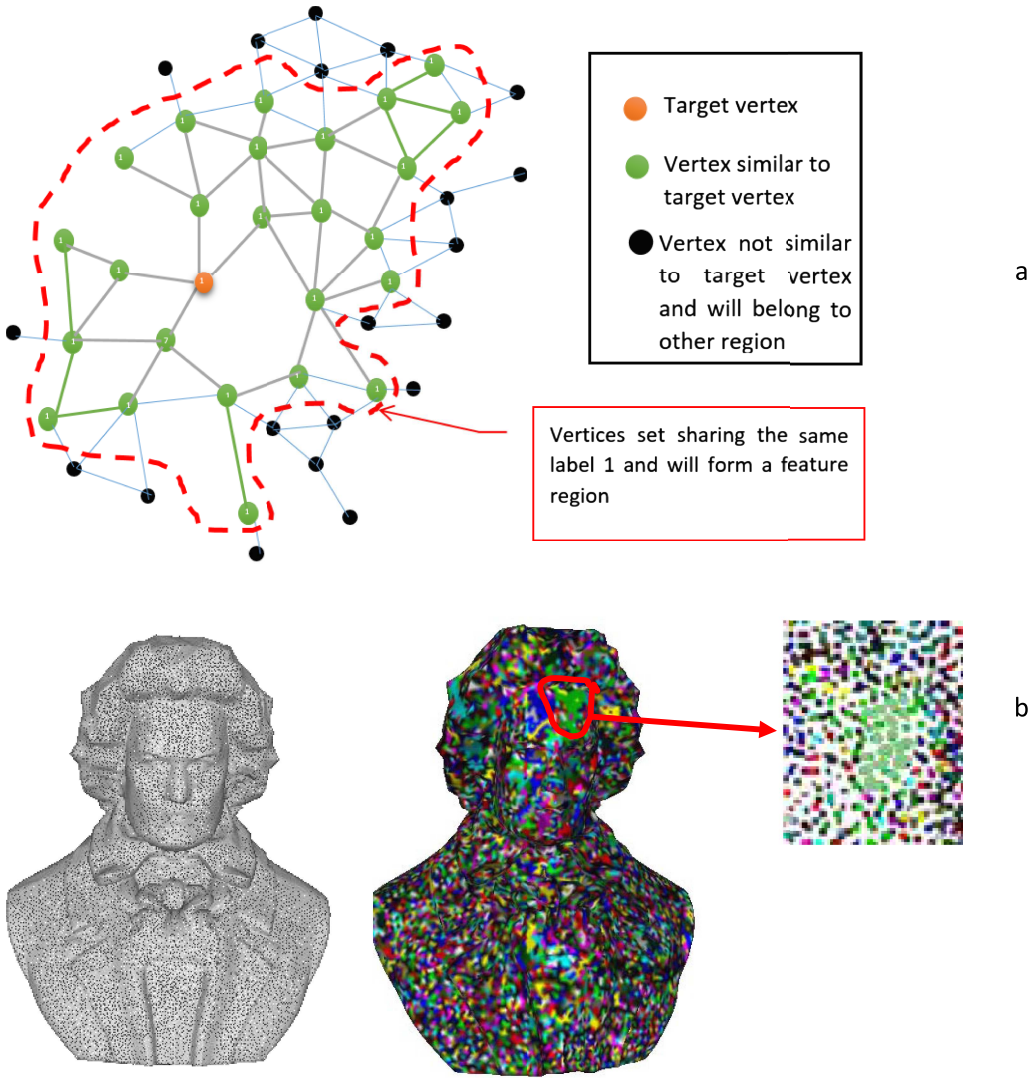

FIGURE 7. Feature regions: *panel a*: illustrates the step of our algorithm; *panel b*: the result of segmentation process.

set of interest points I_p in each segmented region that will represent finally the set of points to conserve in the simplified data.

Before describing the process of segmentation, let us define some functions, which will be used in our algorithm.

To measure the homogeneity of two vertices v_i and v_j according to their saliency degree, we define a saliency similarity function as following:

$$F(v_i, v_j) = \begin{cases} 1 \text{ if } |\mathrm{SL}(v_i) - \mathrm{SL}(v_j)| \leq \sigma \\ 0 \text{ otherwise} \end{cases}, \tag{4.1}$$

where $\mathrm{SL}(v_i)$ is the saliency degree of vertex v_i and σ is a parameter defining the point clouds density.

Let $C(v_i)$ be the set of vertices belonging to $D_n(v_i)$ and similar to v_i in term of saliency.

$$C(v_i) = \{v_j \in D_n(v_i) \,\mathrm{suchas} F(v_i, v_j) = 1\}. \tag{4.2}$$

Algorithm 1 describes the regions segmentation process of 3D point cloud. The algorithm steps and result illustrated in Figure 7.

Algorithm 1. This algorithm assigns to similar (in term of saliency) connected vertices the same label.

```
Algorithm ConnectedVerticesLabeling
Procedure Propagation
Input
      Vertex: start
      Integer: Current_Label

Output:
      Connected vertices have the same Label L

Let '[L1]' an empty list of vertices
Let '[L2]' an empty list of vertices

If (not Selected (start)) then
      Add start to '[L1]'
End if

while (true)
      Begin
      If ('[L1]' is ∅) then
            Quit Procedure
      End if
      For each (Vertex v_i in '[L1]')
            Begin

            Set v_i as Selected
            Assign Current_Label to v_i

               For each (Vertex v_j in N(v_i))
                Begin
                      If (not Selected (v_j)) then
                                          If (F (v_i, v_j) = 1) then
                                 Add v_i to '[L2]'
                           Remove duplicate ('[L2]')
                        End if
                                       End if
               End for
      End for
      Clear ('[L1]')
      Replace '[L1]' by '[L2]'
      Clear ('[L2]')
End While
End Procedure
Let '[L3]' a list of vertices // Vertices represented the point cloud
For each (Vertex v_i in '[L3]')
    Begin
       Propagation (v_i, k)
       k=k+1
End for
End algorithm
```

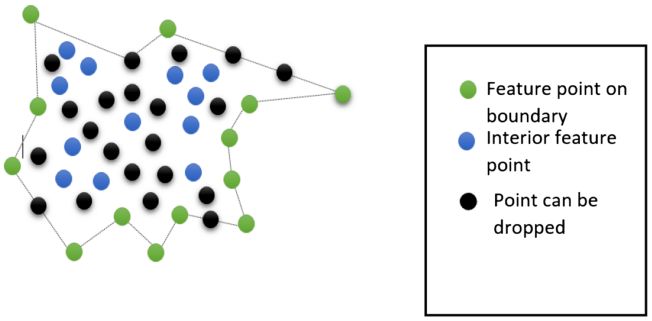

FIGURE 8. Illustration of distribution of interest points (internal and external on boundaries).

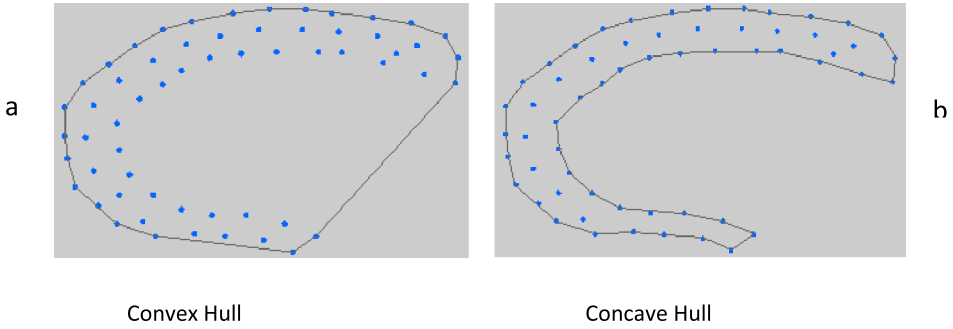

FIGURE 9. Illustration of feature region boundary [8]. The concave hull (*panel b*) of region vertices forms the boundary of feature region.

4.2. Interest points detection

In this subsection, we describe the process of defining the set of interest points in each feature region. Note that an interest point can be located inside a feature region or on its boundaries as shown in Figure 8.

The boundary point set of a feature region is defined as its Concave Hull presented in Figure 9.

The set of interest points are defined amongst the set of boundary points as the extremities of collinear points sets. Figure 10 illustrates the detection of boundary feature points.

The set of collinear points is defined in the form of triplet points. Therefore $P_1(x_1, y_1, z_1), P_2(x_2, y_2, z_2)$, and $P_3(x_3, y_3, z_3)$ are collinear if, the area A of the triangle composed by P_1, P_2 and P_3 equals to zero. Let a, b and c be the sides of this triangle and defined as:

$$\begin{aligned} a &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\ b &= \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2} \\ c &= \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2}. \end{aligned} \tag{4.3}$$

The area of the triangle s is calculated using Heron's formula:

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \tag{4.4}$$

where s represents the half of triangle perimeter.

The set of interior points in each feature region are weighted by their saliency degree. Thus, we can classify these points as High salient or Low salient points. This classification is made according to a local mean saliency

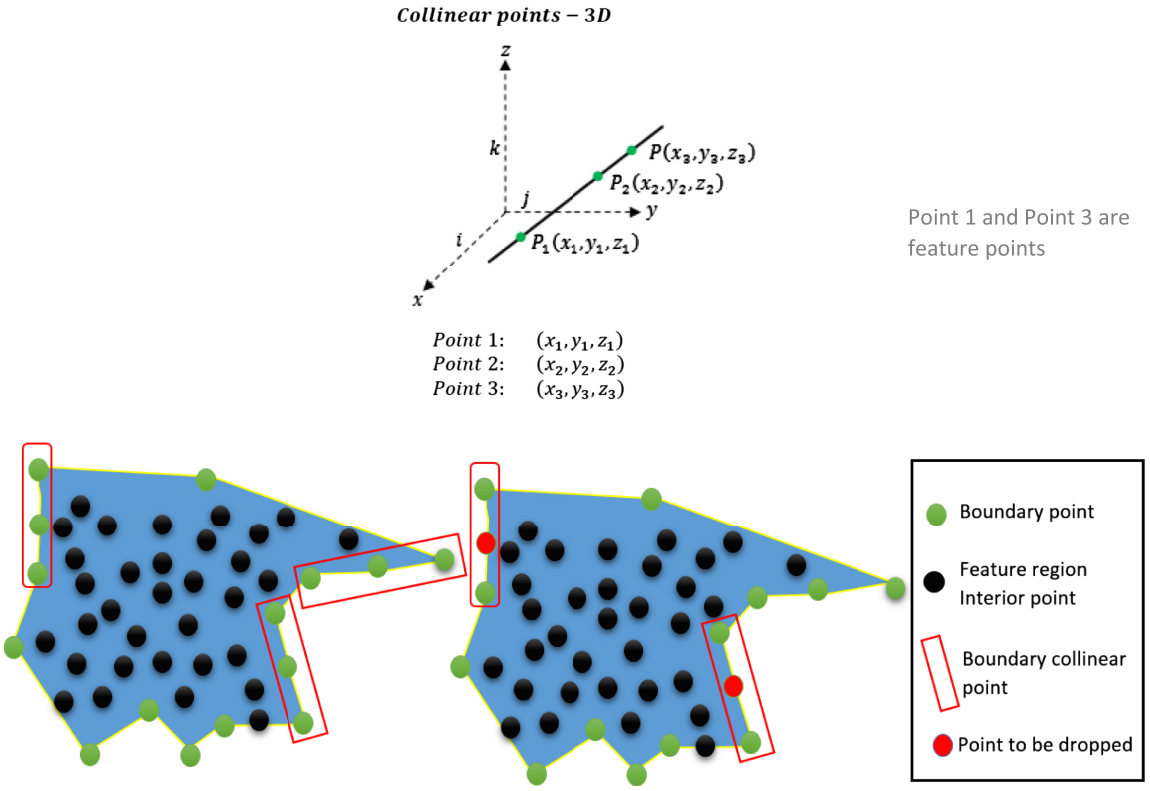

FIGURE 10. Boundary feature points. The extremities of each set of boundary collinear points (Green vertices) are the interest points.

measure (LMS) and a global mean saliency measure (GMS) as following:

$$\mathrm{LMS}\,(r) = \frac{\sum_{v_i \in r} \mathrm{SL}(v_i)}{|r|}, \tag{4.5}$$

$$\mathrm{GMS} = \frac{\sum_{v_i \in G} \mathrm{SL}(v_i)}{|G|}, \tag{4.6}$$

where r represents the set of points in each feature region,and $|G|$ the number of graph vertices.

To check whether a point is an interest point, we define the function $\mathrm{ITP}(v_i, r)$ as:

$$\mathrm{ITP}(v_i, r) = \begin{cases} 1 \text{ if } (\mathrm{GMS} - \propto) \leq \mathrm{SL}\,(v_i) \text{ or } (\mathrm{LMS}\,(\mathbf{r}) - \propto) \leq \mathrm{SL}(v_i) \\ 0 \qquad\qquad \text{otherwise} \end{cases}, \tag{4.7}$$

where $\propto$ is a threshold parameter.

Finally, the resulting simplified 3D point cloud is composed by all feature points classified as high saliency point. In some cases where the set of interior points is classified as low saliency points, the fact that produce gaps in the simplified 3D point cloud. To handle this case, we propose to simplify this set according to the following rules and illustrated in Figure 11.

- Compute the gravity center GCS of region high salient points,
- Compute the midpoint of segment connecting each boundary interest point with the gravity center GCS ,
- Maintain the nearest low salient point of each midpoint.

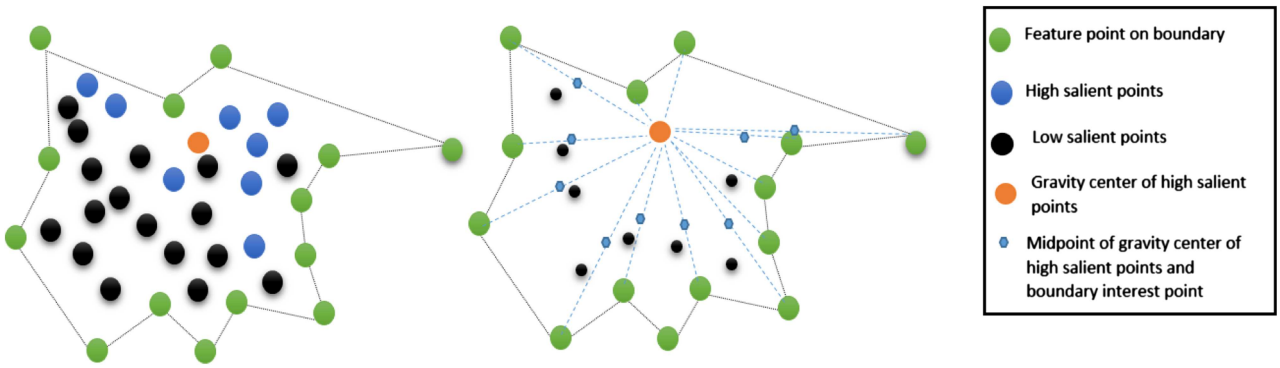

FIGURE 11. Region gaps treatment.

5. EXPERIMENTAL RESULTS

In this section, we show the efficiency and robustness of our proposed method on many colored 3D point clouds models and 3D meshes. In these experimental results, we present and discuss the influence of different parameters on our approach such as graph Delaunay-level k, the variable x in equation (3.6) that defines the deviation factor threshold in patches entropy calculation $\propto$ threshold parameter $\propto$ and the number of patch cells n.

Figure 12 presents the results of our simplification method applied on different point clouds. Images (a_1, a_2, a_3, a_4 and a_5) show the original 3D point clouds, images (b_1, b_2, b_3, b_4 and b_5) show the simplified point clouds with the following parameters values ($n = 8$, $x = 9$ and $\propto = 0$). Images (c_1, c_2, c_3, c_4 and c_5) show the simplified point clouds with ($n = 8$, $x = 9$ and $\propto = 30$). Images (d_1, d_2, d_3, d_4 and d_5) show the simplified point clouds with ($n = 8$, $x = 9$ and $\propto = 50$). One can notice that the increase of the simplification threshold decreases the number of points in a point cloud without affecting the shape. These results demonstrate that the features of models are well maintained by our method, even when the number of points is reduced.

In order to evaluate the quality of the simplified model generated by our method, we propose to measure the geometric error as the maximum error between the initial M and the simplified M' model, *i.e.* the hausdorff distance.

$$\triangle_{\max}(M, M') = \max_{q \in M} d(q, M'), \tag{5.1}$$

and the geometric average error:

$$\triangle_{\text{avg}}(M, M') = \frac{1}{||M||} \sum_{q \in M}^{d} (q, M'). \tag{5.2}$$

Figure 13 presents the influence of the deviation factor threshold x on the result. Image a shows the original 3D point cloud. Images b–e present the simplified point clouds with ($n = 8$, $x = 0$ and $\propto = 0$), ($n = 8$, $x = 5$ and $\propto = 0$), ($n = 8$, $x = 9$ and $\propto = 0$) and ($n = 8$, $x = 13$ and $\propto = 0$). One can see that the sharp curves and boundaries of regions that define the characteristics of shape surface are preserved although a minimum value of deviation factor threshold is used ($b, x = 0$) (Tab. 1).

Figure 14 presents the influence of the deviation factor threshold x and threshold parameter $\propto$ on the simplification rate. One can notice that the increase of the simplification threshold $\propto$ increase the simplification rate but the increase of the deviation factor threshold x decrease the simplification rate and in all cases the shape is not affected (Tab. 2).

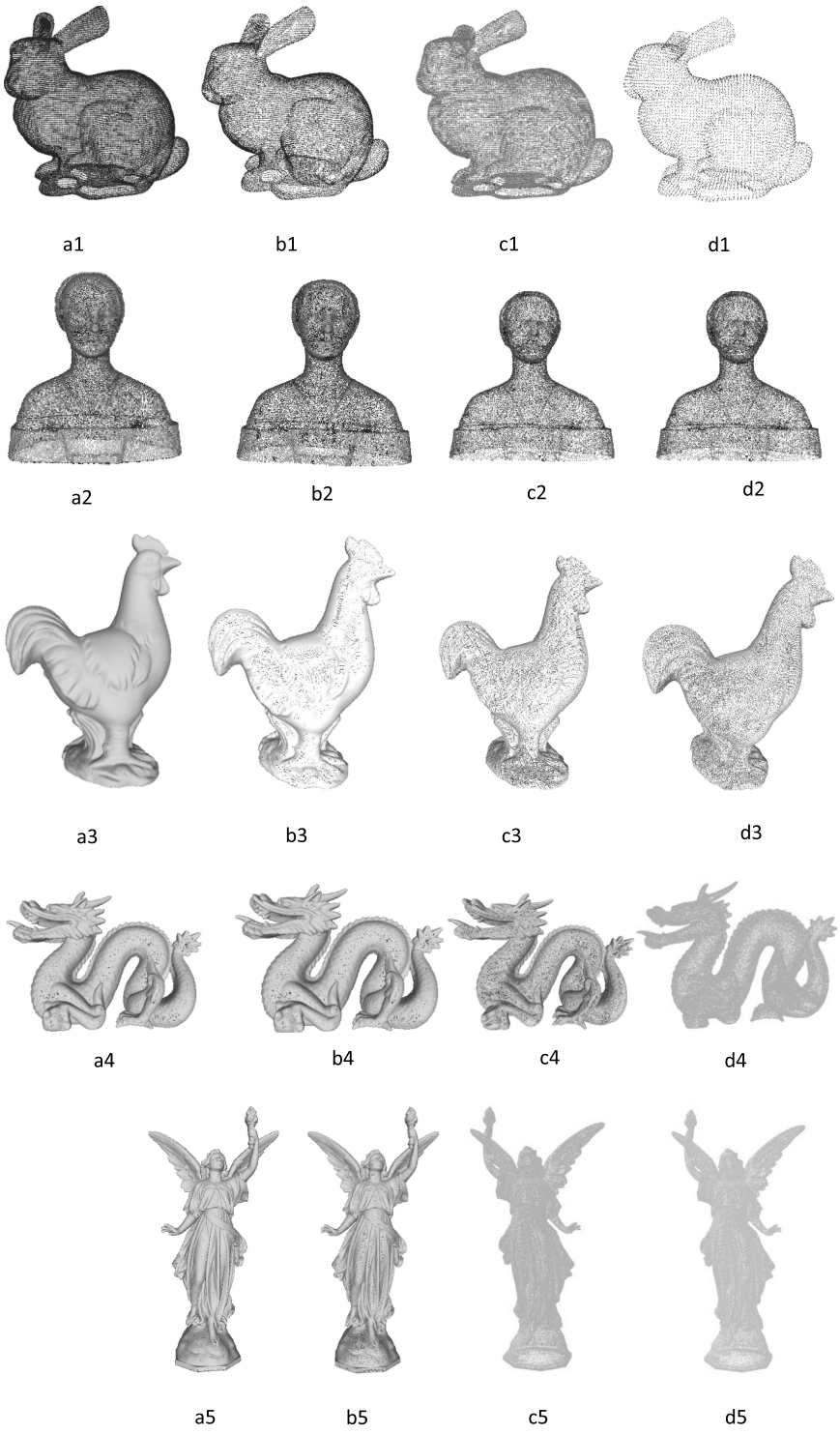

FIGURE 12. Results of simplification with different thresholds ($\propto = 0, 30$ and 50).

TABLE 1. This table summarizes the simplification rates and the evaluation of results showed in Figure 12.

Model	Number of original points	Number of simplified 3D points						Geometric error	
		$\propto = 0$	Rate	$\propto = 30$	Rate	$\propto = 50$	Rate	$\Delta_{\max}$	Δ_{avg}
Bunny	35947	30134	16%	27316	24%	16476	54%	0.005151	0.000265
Laurana	27861	18819	32%	17492	37%	16241	41%	0.003313	0.000014
Chicken_high	135142	120415	10%	118812	12%	100616	25%	0.050000	0.012888
Dragon	437645	387412	11%	185620	57%	33960	92%	0.044323	0.021243
Lucy	262909	242308	7%	212032	19%	106007	59%	0.064213	0.014888

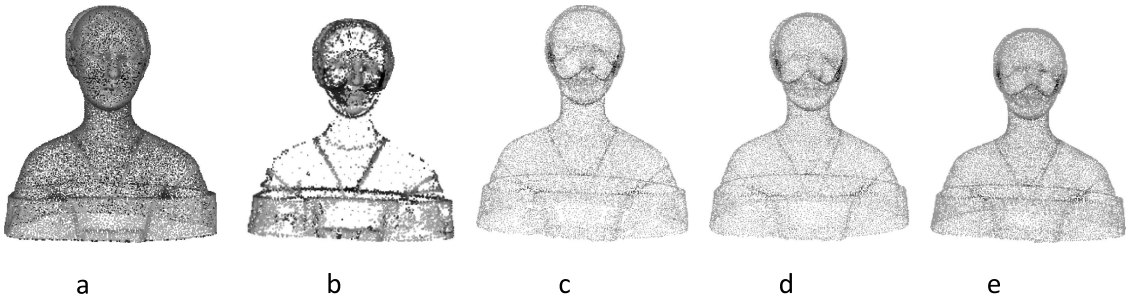

FIGURE 13. Contribution of the deviation factor thresholds ($x = 0$, 2, 5 and 9).

TABLE 2. This table summarizes the simplification rates of results showed in Figure 13.

Model	Number of original points	Number of 3D simplified points							
		$x = 0$	Rate	$x = 5$	Rate	$x = 9$	Rate	$x = 13$	Rate
Laurana	27861	9764	64%	19636	29%	22894	17%	22929	17%

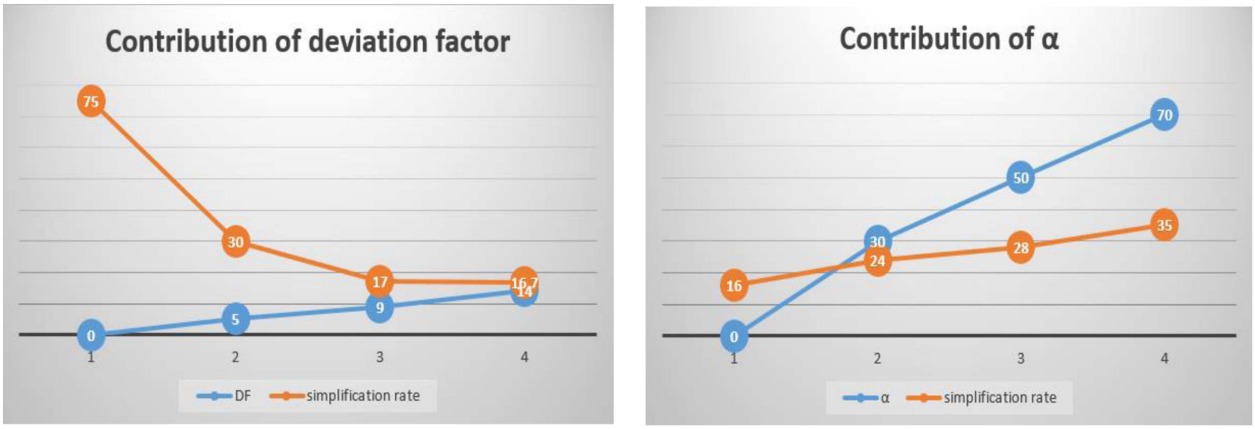

FIGURE 14. influence of the deviation factor thresholds (x) and thresholds ($\propto$) on simplification rate.

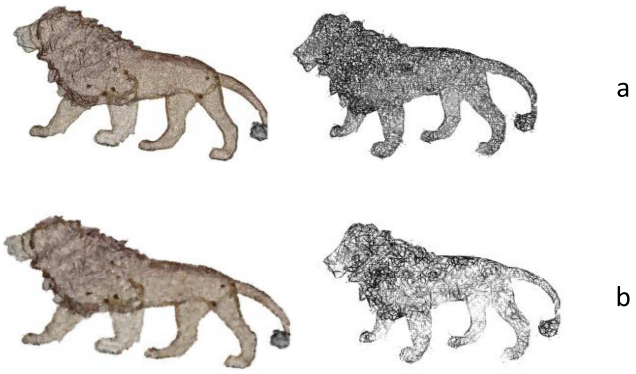

FIGURE 15. Results of simplification (original lion: (a), noisy lion: (b)) with threshold ($\propto = 0$).

Figure 15 presents the stability of our method according to noise. The first row (a) shows the simplification result of our method applied on an object of 18 3408 vertices with threshold parameter ($\propto = 0$). The simplified object has 152594 vertices. In contrast, by applying our method on a noisy object contains the same number of vertices approximatively (b), we can notice that the result is similar to the original object where the simplified object has 15 0584 vertices. In addition, the boundaries and the shape are not affected.

6. EVALUATION

In addition to the geometric average error measurement proposed in the previous section to evaluate the simplification quality, we compare our method with some related works in order to show its performance. First, note that our method has many advantages compared to some related work where the user can control the simplification rate. Furthermore, there are some cases where the simplification rate is very high [37] and some of sharp region points that conserve the characteristics of the surface shape will be removed resulting holes in the simplified 3D point clouds. In contrast, our method preserve the characteristics of the surface shape with a minimal configuration ($x = 0, \propto = 0$) due to the technique we use where we maintain internal and boundary feature points. Figure 16 shows the simplification results proposed by Liao *et al.* [15] where the authors mention that their proposed method works only with models whose shape is symmetrical or spherical and may produces holes in others. In contrast, Figures 12 and 13 shows that our method produce better results with models of different shapes (symmetric and no symmetric).

Table 3 shows a comparison of some results from [18,37] and our method. One can see that our method gives simplified models with lesser points than other methods and maintain the boundaries and ridges of 3D models.

Finally, by comparing the geometric error on simplification results (Bunny model) between [18] and our method (Tab. 1, first row), we can notice that our method is better.

TABLE 3. Comparison between our method and other works ([18,37]).

Model	#Original points	Article	#Points of simplified model (related work)	#Points of simplified model (our method)	Screen shot in corresponding article
Bunny	280 792	[18]	16 729	16 476	Not available
Dragon	437,645	[37]	34 861	33 960	Figure 17

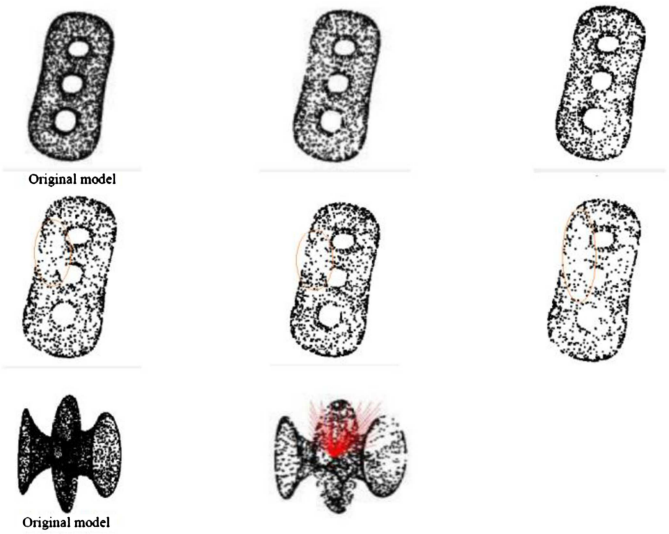

FIGURE 16. Simplification results proposed by [15].

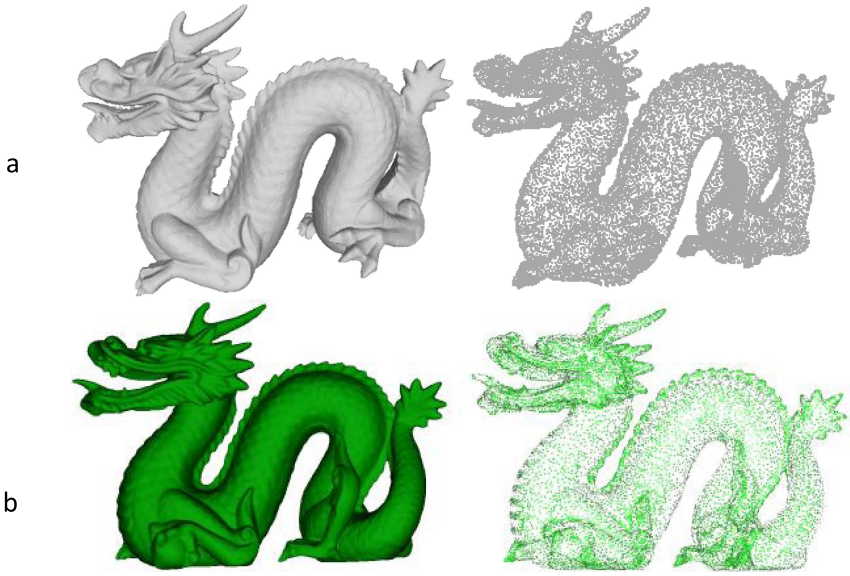

FIGURE 17. Dragon model: *panel a*: our method, *panel b*: method in [37].

7. Conclusion

In this paper, we have presented a novel approach for 3D point clouds simplification. The idea of our approach is to detect the feature regions that characterize the shape surface using the saliency degree measure followed by detecting and maintaining feature points in each feature region. To do so, we created a local patch descriptor for each vertex where the patch cell is filled with the average of the deviation factor of each vertex according to

the vertex neighbors that projected onto the same location associated to the cell. Next, the patch is used as a local descriptor for vertices and the saliency degree for each vertex is calculated.

In order to detect the feature regions, we presented an efficient algorithm based on the connected component labeling according to vertices saliency degree. Then, we detected feature points in these regions. Note that the user can control the degree of simplification by a resolution parameter where the simplification rate increases with the value of this parameter.

Finally, the robustness and the efficiency of our approach is demonstrated through some experimental results where our method can reduce the number of point clouds and maintain the shape characteristics without producing holes in the surface. Additionally, we compared our method with various state-of-the-art methods. In this context, we propose as perspective the use of feature points identified by our approach in many applications such as 3D objects matching.

References

[1] E. Altantsetseg, Y. Muraki, F. Chiba and K. Konno, 3D surface reconstruction of stone tools by using four-directional measurement machine. *Int. J. Virtual Reality (IJVR)* **10** (2011) 37–43.

[2] E. Altantsetseg, Y. Muraki, K. Matsuyama and K. Konno, Feature line extraction from unorganized noisy point clouds using truncated fourier series. *Visual Comput.* **29** (2013) 617–626.

[3] J.L. Bentley, Multidimensional binary search trees used for associative searching. *Commun. ACM* **18** (1975) 509–517.

[4] D. Brodsky and B. Watson, Model simplification through refinement. *Proc. Int. Conf. Graphics Interface, Quebec, Canada* (2000) 221–228.

[5] A. Chida, K. Matsuyama, F. Chiba and K. Konno, A rapid searching method of adjacent flake surfaces in stone implements by using sets of measured points for generating a joining material. *J. Soc. Art Sci.* **13** (2014) 107–115.

[6] P. Cignoni, C. Montani and R. Scopigno, A comparison of mesh simplification algorithms. *Comput. Graphics* **22** (1998) 37–54.

[7] P. Cignoni, C. Rocchini and R. Scopigno, Metro: measuring error on simplified surfaces. *Comput. Graphics Forum* **17** (1998) 167–174.

[8] Concave Hull, available at: http://ubicomp.algoritmi.uminho.pt/local/concavehull.html

[9] A. El Chakik, A. Elmoataz and X. Desquesnes, Mean curvature flow on graphs for image and manifold restoration and enhancement. *Signal Process.* **105** (2014) 449–463.

[10] R. Gal and D. Cohen-Or, Salient geometric features for partial shape matching and similarity. *ACM Trans. Graphics* **25** (2006) 130–150.

[11] K.H. Lee, H. Woo and T. Suk, Point data reduction using 3D grids. *Int. J. Adv. Manuf. Technol.* **18** (2001) 201–210.

[12] K.H. Lee, H. Woo and T. Suk, Data reduction methods for reverse engineering. *Int. J. Adv. Manuf. Technol.* **17** (2001) 735–743.

[13] P.F. Lee and B.S. Jong, Point-based simplification algorithm. *J. WSEAS Trans. Comput. Res.* **3** (2008) 61–66.

[14] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, The digital Michelangelo project: 3D scanning of large statues. In: *Proceedings of ACM SIGGRAPH, 1 July 2000* (2000) 131–144.

[15] C. Liao, X. Niu and M. Wang, Simplification of 3D point cloud data based on ray theory. *Comput. Model. New Technol.* **18** (2014) 273–278.

[16] F. Lozes, A. Elmoataz and O. Lézoray, Nonlocal processing of 3D colored point clouds. In: *21st International Conference on Pattern Recognition* (2012) 1968–1971.

[17] D.P. Luebke, A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graphics Appl.* **21** (2001) 24–35.

[18] Y. Miao, R. Pajarolac and J. Feng, Curvature-aware adaptive re-sampling for point-sampled geometry. *Comput. Aided Des.* **41** (2009) 395–403.

[19] C. Moenning and N.A. Dodgson, A new point cloud Simplification algorithm. In: *Proceedings of 3rd IASTED Conference on Visualization, Imaging and Image Processing* (2003) 1027–1033.

[20] C. Moenning and N.A. Dodgson, Intrinsic point cloud Simplification. In: *Proceedings of the 14th International Conference on Computer Graphic and Vision (GraphiCon), Moscow, Russia* (2004).

[21] G. Mullineux and S.T. Robinson, Fairing point sets using curvature. *Comput. Aided Des.* **39** (2007) 27–34.

[22] A. Nouri, C. Charrier and O. Lézoray, Multi-scale mesh saliency with local adaptive patches for viewpoint selection. *Signal Process. Image Commun.* **38** (2015) 151–166.

[23] M. Pauly, M. Gross and L.P. Kobbelt, Efficient simplification of point-sampled surfaces. In: *Proceedings of the Conference on Visualization'02*, IEEE Computer Society (2002) 163–170.

[24] X. Peng, W. Huang, P. Wen and X. Wu, Simplification of scattered point cloud based on feature extraction. In: *WGEC'09 Proceedings of the 2009 third International Conference Genetic and Evolutionary Computing, October 14–17* (2009) 335–338.

[25] H. P_Ster, M. Zwicker, J. Van Baar and M. Gross, Surfels: surface elements as rendering primitives. In: *SIGGRAPH'00 Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000) 335–342.

[26] Y. Qiu, X. Zhou, P. Yang and X. Qian, Curvature estimation of point set data based on the moving-least square surface. *J. Shanghai Jiaotong Univ. (Science)* **16** (2011) 402–411.

[27] E. Shaffer and M. Garland, Efficient adaptive simplification of massive meshes. In: *VIS'01: IEEE Transactions on Visualization'01, 21–26 October* (2001) 127–134.

[28] B.Q. Shi, J. Liang and Q. Liu, Adaptive simplification of point cloud using k-means clustering. *Comput. Aided Des.* **43** (2011) 910–922.

[29] P. Shilane and T. Funkhouser, Distinctive regions of 3D surfaces. *ACM Trans. Graphics* **26** (2007) 7.

[30] H. Song and H.Y. Feng, A global clustering approach to point cloud simplification with a specified data reduction ratio. *Comput. Aided Des.* **40** (2007) 281–292.

[31] H. Song and H.-Y. Feng, A progressive point cloud simplification algorithm with preserved sharp edge data. *Int. J. Adv. Manuf. Technol.* **45** (2009) 583–592.

[32] R.D. Toledo, B. Levy and J. Paul, Reverse engineering for industrial-environment cad models. In: *Proceedings of TMCE 2008, April 21–25, Kusadasi, Turkey* (2008).

[33] T. Varady, R. Martin and J. Cox, Reverse engineering of geometric models – an introduction. *Comput. Aided Des.* **29** (1997) 255–268.

[34] J. Wu, X. Shen, W. Zhu and L. Liu, Mesh saliency with global rarity. *Graphical Models* **75** (2013) 255–264.

[35] K. Yamahara, K. Konno, F. Chiba and M. Sato, A method of detecting adjacent flakes in stone tool restoration by extracting peeling surfaces. *Jpn. Soc. Archaeological Inf.* **17** (2011) 23–32.

[36] X. Yang, K. Matsuyama, K. Konno and Y. Tokuyama, Feature-preserving simplification of point cloud by using clustering approach based on mean curvature. *J. Soc. Art Sci.* **14** (2014) 117–128.

[37] Y. Yoshida, K. Konno and Y. Tokuyama, A distributed simplification method with PC cluster. *J. Soc. Art Sci.* **7** (2008) 113–123.