Point-based Simplification Algorithm

Pai-Feng Lee[1], Bin-Shyan Jong[2]
Department of Information Management, Hsing Wu College[1]
Dept. of Information and Computer Engineering Engineering, Chung Yuan Christian University[2]
093094@mail.hwc.edu.tw[1], bsjong@ice.cycu.edu.tw[2]

Abstract: - This study presents a novel, rapid, and effective point simplification algorithm based on a point cloud without any normal and connectivity information. This study is initiated with a scattered sampled point set in three dimensions, and the final output is a triangular mesh model, which is simplified according to a restrictive criteria. The proposed method reduces the number of calculations required to establish the relation between triangulation and the connectivity. Due to the continuous development of computer graphics technology, diversified virtual reality applications are being increasingly adopted. Recently, the efficient and vivid portrayal of 3D objects in the real world in virtual scenes has become a crucial issue in computer graphics. A triangular mesh is one of the most popular data structures for representing 3D models in applications. Numerous methods currently exist for constructing objects using surface reconstruction. The data required for the sampled points are generally obtained from a laser scanner. However, the extracted sampled points are frequently affected by shape variation. The number of triangles created increases with the number of points sampled from the surface of a 3D object, which helps in the reconstruction of the correct model. Nevertheless, subsequent graphics applications, such as morphing or rendering, increase the computation costs. Appropriate relevant points should be chosen so as to retain the object features and reduce the storage space and calculation costs.

Key-Words: - point simplification, discrete shape operator, feature extraction, curvature, torsion.

1 Introduction

Due to recent advances in computer graphics field, the efficient and realistic representation of 3D models has become an important topic. However, displaying realistic scenes and models in a virtual environment typically depends on triangular mesh models. In general, the creation of a triangular mesh model includes the steps of acquisition, processing, and rendering. The first step is the acquisition of the 3D point information of a physical object. By using a 3D laser scanner, the 3D digital contents can be obtained. In the processing step, 3D models are processed further, which includes spectral processing, Boolean operation, free-form deformation, morphing, interactive painting, brush painting, and so on. Finally, some rendering techniques are adopted to display the processed models.

Although point-based rendering techniques have been developed, mesh-based rendering techniques continue to have wider applications. To obtain a good rendering result, a point cloud model must be reconstructed as a triangular mesh model. However, the number of triangles in the model increases with the quality of the model. Certainly, the calculation cost will also increase with the storage space costs of the models. To reduce the computation cost, many surface simplification methods are used to decrease the number of triangles. In other words, the steps involved in 3D digital content creation include surface reconstruction and surface simplification, and the former is executed before the latter.

The number of triangles created increases with the number of points sampled from the surface of a 3D object, which helps to reconstruct the correct model. However, subsequent graphics applications increase the computation costs. Appropriate relevant points should be chosen so as to retain the object features and reduce the storage space and costs.

2. Related Works

2.1 Surface Simplification

In addition, to produce a 3D model that provides good rendering, numerous triangles are frequently used to represent the models in the scenes. Consequently, the storage space and computation costs of the model increase. Hence, the achievement of a reduction in the number of triangles while maintaining the contour of the objects has recently become a major issue in this field. For example, the vertex pair contraction proposed by Garland *et al.* [5] has been identified as one of the best simplification methods in recent years. The method has decimation operations that are generally arranged in a priority queue according to an error matrix that quantifies the errors caused by decimation. Simplification is performed iteratively to reduce any smoothing of point pairs that has been caused by the decimation operation. This greedy technique can obtain the simplified model with a minimum error from the original model. However, this simplification algorithm needs a triangular mesh and connectivity in advance. In other words, the algorithm is burdened with a large number of computations before simplification. Consequently, this process is prohibitively expensive. Neither can the vertex pair contraction effectively retain the outlook of the object in terms of low resolution nor can it maintain the required detailed characteristics in terms of the detailed features of the object.

This study proposes an efficient point simplification method to retain the physical features of the model. The discrete shape operator *(DSO)* is adopted to extract the feature vertices of the models, and these vertices are postponed to simplify. The proposed point simplification method improves the quadric error metric [5] of the vertex pair contraction; hence, it not only effectively simplifies the model and maintains the features of the object model, but due to the consideration of the point cloud it also decreases the pre-processing time cost associated with the reconstruction.

2.2 Point Cloud Simplification

Although reconstruction of a 3D mesh model from the sampled point cloud and performing simplification is the normal process, this sequence requires the establishment of a triangular mesh and connectivity prior to simplification. Therefore, point cloud simplification is an attractive approach. Point-based simplification is performed before reconstruction. If suitable relevant points can be extracted from a point cloud that represents the surface variation, then the number of calculations needed for reconstruction can be significantly reduced. Dey *et al.* [9] presented the first point cloud simplification approach, and adopted local feature sizes to detect the redundancy in the input point cloud and to ensure relevant point densities; this was accomplished by exploiting a 3D Voronoi diagram. Boissonnat and Cazals *et al.* [3] presented a coarse-to-fine point simplification algorithm that randomly calculates a point subset and constructs a 3D Delaunay triangulation. These algorithms must adopt pre-processing to retain the original surface data before simplifying the point set, and therefore require many computations.

Pauly *et al.* [7] applied four mesh-based simplification techniques to point cloud simplification. Alexa *et al.* [4] proposed the uniform simplification of the point set by estimating the distance from a point to the moving least square *(MLS)* surface. Moenning and Dodgson *et al.* [2] presented an intrinsic coarse-to-fine point simplification algorithm that guarantees uniform or feature-sensitive distribution. However, their method requires many computations and a large memory.

This study presents a rapid and effective point cloud simplification algorithm before discussing surface reconstruction and surface simplification. This study adopts the DSO to find the weight of the features of the 3D model, and it presents a novel method for extracting the relevant points for a dense input point set; it finally adopts the reconstruction algorithm proposed by Dey *et al.* [10] to generate the simplified model.

3. Discrete Shape Operator

If p is a point on a surface M, then for each tangent vector v to M at p, let $S_p(v) = -\nabla_v N$. Here, N denotes a unit normal vector field in the neighborhood of p on M, and S_p is called the shape operator of M at p and is derived from N, as shown in Fig. 1.

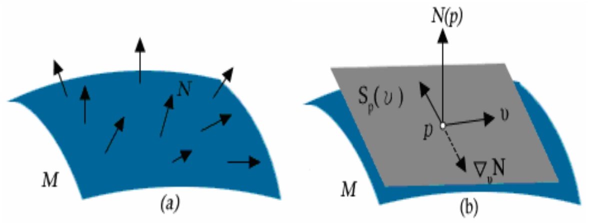


Figure 1. (a) Unit normal vector field of surface M and (b) shape operator for tangent vector v at p.

On the curve $r(s)$ that belongs to category C^3 based on the arc parameter s, the derivatives of the three unit vectors $\vec{T}$, $\vec{N}$, and $\vec{B}$ at each regular point concerned with the parameter s shall satisfy the following Frenet-Serret formulae [1]:

$$\begin{pmatrix} \frac{d\vec{T}}{ds} \\ \frac{d\vec{N}}{ds} \\ \frac{d\vec{B}}{ds} \end{pmatrix} = \begin{pmatrix} 0 & k & 0 \\ -k & 0 & \tau \\ 0 & -\tau & 0 \end{pmatrix} \cdot \begin{pmatrix} \vec{T} \\ \vec{N} \\ \vec{B} \end{pmatrix}$$

In the equation, $\vec{T}$ denotes the tangent vector of the curve; $\vec{N}$, the principal normal vector of the curve; $\vec{B}$, the binormal vector of the curve; k, the curvature of point p on the curve; and τ, the torsion of point p on the curve.

Two quantified functions, namely, curvature and torsion, can be extracted from the curve by using the curve theory. The curvature k denotes the variation of the tangent vectors, and the torsion τ denotes the variation in the binormal vectors. Hence, the value of the shape operator—$\sqrt{k^2+\tau^2}$—denotes the degree of local surface variation, and it can be adopted as a criterion for the model's feature area.

The curvature k is estimated as the sum of the differences between the tangent vectors at the sampling point p and all its adjacent points divided by the individual distance d_i. The torsion τ of point p can be estimated from the relationship between the binormal vectors of point p and its adjacent points by summing the differences between the binormal vectors of point p and each of its neighboring points divided by the individual distance d_i. The calculations of the curvature and torsion are demonstrated in Figs. 2 and 3, respectively.

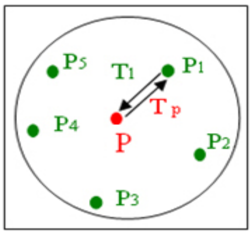


Figure 2. Diagram of curvature calculation, $k = \| (T_P - T_1)/d_1 + \dots\dots + (T_P - T_n)/d_n \|$.

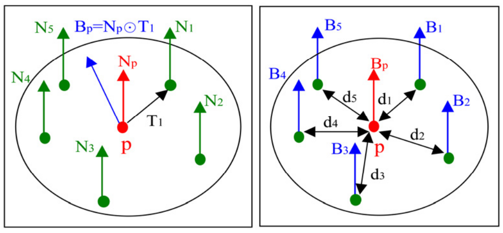


Figure 3. Diagram of torsion calculation, $\tau = \| (B_P - B_1)/d_1 + \dots\dots + (B_P - B_n)/d_n \|$.

A point cloud model consists of only point information. Therefore, to estimate the shape operator of the local surface in a point cloud model, the local sampling approach and the concept of discretization must be adopted. In this study, we adopted the local sampling approach proposed by Gopi *et al.* [6]. This approach can adaptively estimate the sampling region according to the point distribution, and it can also decrease the occurrence of oversampling. Figure 4 shows the feature extraction results of point cloud models. Greener vertices indicate that the features are flatter. On the contrary, redder vertices show that they have a high variation.

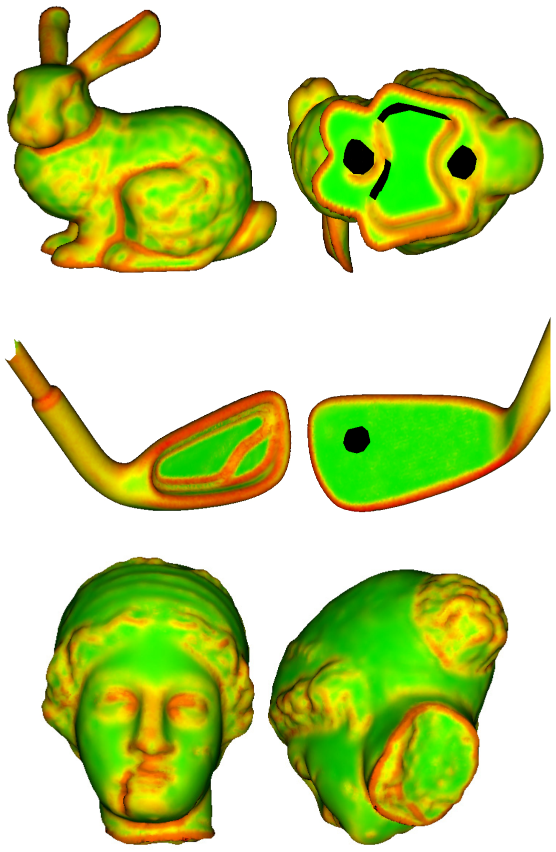


Figure 4. Detection of high-variation vertices using DSO for point cloud models.

The results show that our proposed approach can find the high-variation vertices. The feature weights (*DSO*) are used to preserve the features of the original model.

4. Point Simplification Algorithm

Our algorithm can be divided into the following steps.

4.1. To find neighboring points of each point

For efficiency, the point clouds are analyzed locally; hence, the neighboring points of each point are to be determined first. Here, the set of neighboring points of p is denoted as the region N_p. In this study, we adopted the sampling approach proposed by Gopi *et al.* [6]. Gopi estimates the sampling region of p from the distance s between a sampled point p and its closest point q. In other words, Gopi's method exploits the distance s to estimate the local density and derive the local area enclosed by ms, where m is a constant. The points within this range are regarded as adjacent points and are adopted for subsequent calculations, as shown in Fig. 5.

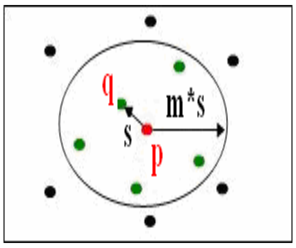


Figure 5. Taking *ms* adjacent points of *p*.

4.2. To find the normal of each point

To find the normal of each point *p*, the best-fit plane for N_p is calculated. In this study, it is calculated as the least-square fitting plane with perpendicular offsets. Let *d(q)* denote the perpendicular distance from a neighboring point *q* to the unknown plane that we are attempting to find. The least-square fitting plane with perpendicular offsets is the plane that minimizes $\sum_{i=1}^{k} d(q)^2$, i.e., it is the plane in which the sum of squares of the perpendicular distance from the neighboring points is a minimum (as illustrated in 2D in Fig. 6). The normal is then set to be the normal of the best-fit plane of N_p.

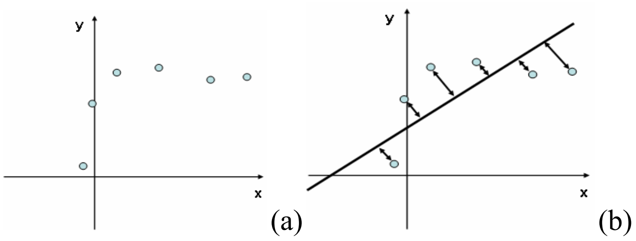


Figure 6. Least-square fitting plane: (a) Points in N_p and (b) a least-square fitting plane with perpendicular offsets corresponding to these neighboring points.

4.3. To determine the DSO of each point

After estimating the unit normal vector, this study employs the unit normal vector and N_p to estimate the discrete shape operator *(DSO$_{(p)}$)* at each sampled point *p*.

4.4. To determine the sequence of decimation

The vertex pair contraction [5] includes decimation operations that are generally arranged in a priority queue according to an error matrix that quantifies the errors caused by the decimation. However, this simplification algorithm requires a triangular mesh and connectivity in advance. In order to apply the strategy of Garland *et al.* [5], this study assumes the vertices q_i in the region N_p as the virtual edge $\overline{pq_i}$. We sort the vertices q_i of N_p and obtain the virtual triangles $\triangle pq_iq_j$. The proposed algorithm refers to Garland's study, and includes six major steps that can be described as follows:

1. **Calculate the quadric error metric and DSO$_{(p)}$ of each vertex *p*.**
   The quadric error metric is $Q_{(p)}$ = $DSO_{(p)} \cdot \Sigma K_m$, where m denotes the plane that contains the point p and K_m represents the 4 × 4 metric of the plane m.

2. **Choose each virtual edge pair to calculate the minimum error produced due to the simplification.**

3. **Choose the lowest error vertex pair as an object of the simplification.**

4. **Contract the vertex pair *(p_i, p_j)* into *p'* and calculate its *quadric error metric Q*, where *$Q_{(p')} = (Q_{(pi)} + Q_{(pj)})$*.**
   For a given contraction (p_i, p_j)→ p', p' is chosen as either p_i or p_j to obtain the minimal error.

5. **Update all the information of the vertices adjacent to *p_i* and *p_j*.**

6. **Repeat the previous steps until the required number of vertices are reached.**

The discrete shape operator (*DSO*) is calculated and considered as a weight to modify the quadric error metric. The experimental results demonstrate that the DSO helps to actually reduce the simplification error.

4.5. To reconstruct the simplified model

After simplifying the input point cloud, the algorithm presented by Dey *et al.* [10] is adopted to reconstruct the simplified model and the error is measured using the Metro tool [8]. The experimental results confirm that a good simplified model can be quickly obtained.

5. Results

The simplified models listed below are obtained using the proposed method. Table 1 shows the properties of the models, which are expressed in terms of the original vertices and DSO estimation time. Figure 7 shows the relative mean error [8] of the simplified models after Dey's reconstruction [10] at a different simplification percentage. Figure 8 presents the point distribution variation of simplified *Venus* models constructed with and without the use of the DSO. The feature-sensitivity distribution is generated by using the DSO. Figure 9 shows the error rate at different levels of the simplified models. Table 2 shows the execution time comparison of different percentage with simplified models, method of doing surface reconstruction first and doing point cloud simplification first. The doing surface reconstruction first method reconstructs the original model first, and then adopts the mesh base vertex-pair contraction [5], which simplifies to the required simplified percentage. The average execution time of reconstruction first method is *124.62* seconds. The average execution time of simplification first method is 4*3.23* seconds. The average improve time ratio is approximately *65.3%*. The improve time ratio = $(Time_R - Time_S) / Time_R$.

6. Conclusion and Further Works

We present a novel algorithm of point cloud simplification. The DSO indicates the outline of the local variations of a local surface, such as variations in the curvature and torsion, and enables the adjustment of the feature characteristics. In other words, the DSO can collect more surface information adaptively and it can also be adopted to extract the features of the model to retain the physical feature of the model under low resolution. The proposed simplification algorithm improves the vertex pair contraction algorithm. The theory proposed in this paper ensures the quality of the simplified model and decreases the storage space and calculation costs.

Directions for future work include the out-of-core implementation of the presented simplification method, the stabilization of data comprising noise, and the application of the DSO to other investigations.

Table 1. Properties of the model considered in the evaluation.

Model	Original Points	DSO Time
Femur	76794	9.79
Horse	48485	6.23
Santa	75781	11.22
Venus	50002	6.73

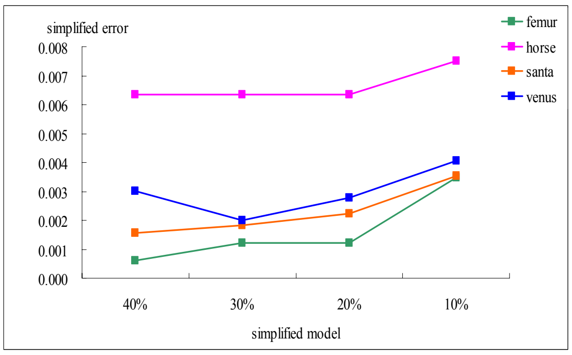


Figure 9. Error rate of the simplified models. Due to the retention of the physical features of point clouds, the proposed method can maintain a low error.

References

[1] B. O'Neill, "Elementary Differential Geometry", *Academic Press*, 1997.

[2] C. Moenning and N. A. Dodgson, "Intrinsic point cloud simplification", *In Proc. 14th GrahiCon*, Vol. 14, 2004.

[3] J.D. Boissonnat and F. Cazals, "Coarse-to-fine surface simplification with geometric guarantees", *EUROGRAPHICS'01, Conf. Proc,* 2001, pp. 490 - 499.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and T. Silva, "Point Set Surfaces", *In Proc. 12th IEEE Visualization Conf.*, 2001, pp. 21 - 28.

[5] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics", *Proceedings of SIGGRAPH 97*, Las Angeles, August 1997, pp 209 – 216.

[6] M. Gopi and S. Krishnan, "A Fast and Efficient Projection-based Approach for Surface Reconstruction", *15th Symposium on Computer Graphics and Image Processing,* 2002.

[7] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient Simplification of Point-Sampled Surfaces", *In Proc. 13th IEEE Visualization Conf.*, 2002, pp. 163 - 170.

[8] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces", *Computer Graphics Forum*, Vol.17, No.2, 1998, pp.167 – 174

[9] T. K. Dey, J. Giesen, and J. Hudson, "Decimating Samples for Mesh Simplification", *In Proc. 13th Canadian Conference on Computational Geometry,* 2001, pp. 85 - 88.

[10] T. K. Dey and S. Goswami, "Tight Cocone: A water tight surface reconstructor", Proc. 8th ACM Sympos. Solid Modeling Appl., 2003,pp. 127 – 134

Table 2. The execution time *(s)* of experimental results measured on a Pentium 4 *(3.0 GHz)* PC having *3 GB* of main memory. The table shows the comparison between performing the reconstruction or the point cloud simplification initially at a required simplification percentage.

Model	**execution time *(s)***	**40%**	**30%**	**20%**	**10%**
Femur	**reconstruct first (*$time_R$*)**	172.38	172.47	172.61	172.70
	point cloud Simplify first (*$time_S$*)	86.67	69.00	51.55	35.03
Horse	**reconstruct first (*$time_R$*)**	107.09	107.16	107.19	107.27
	point cloud Simplify first (*$time_S$*)	44.14	35.23	26.84	19.41
Santa	**reconstruct first (*$time_R$*)**	142.70	142.84	142.94	143.06
	point cloud Simplify first (*$time_S$*)	68.53	56.05	44.27	33.74
Venus	**reconstruct first (*$time_R$*)**	75.77	75.84	75.92	76.00
	point cloud Simplify first (*$time_S$*)	40.36	33.33	26.89	20.61

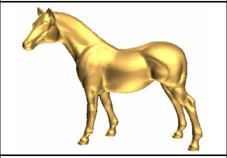
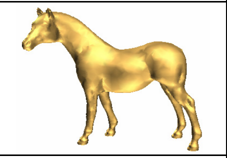
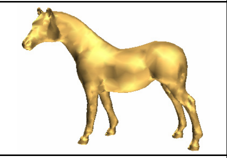
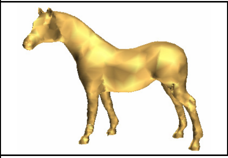
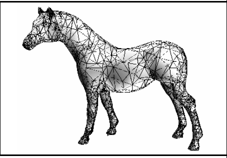
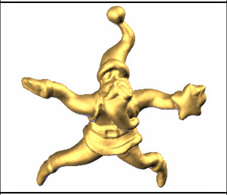
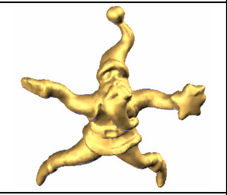
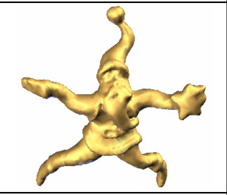
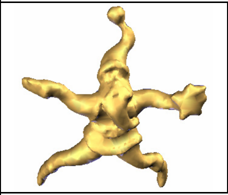
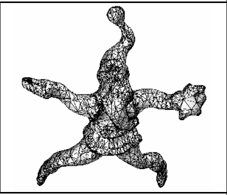
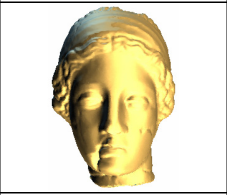
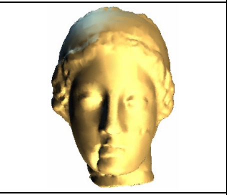
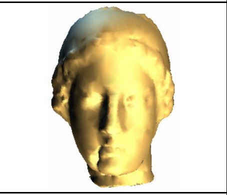
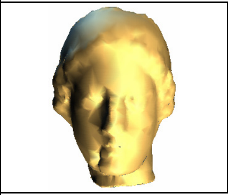
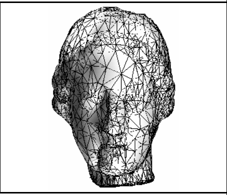
Model	**Original model**	**Simplified model**			
		20%	**10%**	**5%**	
Horse					
points	48485 points	9697 points	4848 points	2424 points	
error		0.006335	0.007514	0.016095	
Santa					
points	75781 points	15156 points	7578 points	3789 points	
error		0.002241	0.003527	0.005259	
Venus					
points	50002 points	10000 points	5000 points	2500 points	
error		0.002784	0.00405	0.005678	

Figure7. Reconstruction results for models at different simplification percentage.

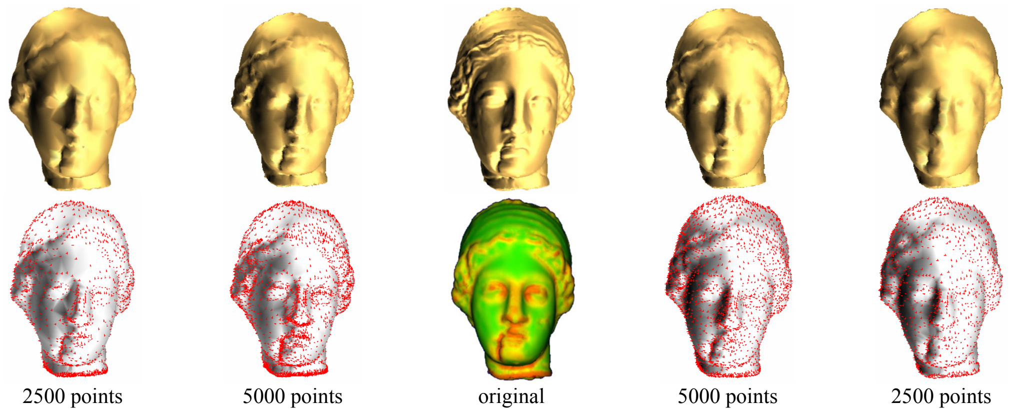


Figure 8. Point distribution in simplified *Venus* models. The results of using the proposed method obtained using DSO *(left half)* and without DSO *(right half)* are shown. The DSO for the point cloud model is shown in the middle. The result of the proposed method using DSO can be used to obtain the feature-sensitive point distribution.