Int J Adv Manuf Technol (2010) 50:235–252
DOI 10.1007/s00170-009-2484-y

ORIGINAL ARTICLE

An improved mesh simplification method using additional attributes with optimal positioning

Han Kyun Choi · Hyun Soo Kim · Kwan Heng Lee

Received: 10 November 2008 / Accepted: 14 December 2009 / Published online: 6 January 2010

Abstract The use of highly detailed models is continuously increasing in computer-aided design (CAD) design and computer graphics field as the technology of range scanners advances. Real-time rendering and manipulating applications are also increasing to support applications in various areas such as collaborative design and scientific visualization. Although graphics hardware technology has been improved rapidly, more attributes such as color, material property, texture coordinate, and curvature are added to CAD models, and it becomes a challenge to handle and render such heavy models. Consequently, the models with complex mesh need to be approximated to improve the efficiency of rendering and manipulation and to reduce computation time. A considerable amount of work has been done regarding geometry preservation, but relatively little research has been performed to preserve both geometry and additional attributes. We present a feature sensitive simplification method using curvature color as an additional attribute. We also use curvature color filtering and optimal positioning methods after edge collapse to preserve feature more sensitively. Our method is applied to several models, and the performance is demonstrated by comparing it with other methods.

H. K. Choi (✉) · H. S. Kim · K. H. Lee
Gwangju Institute of Science and Technology(GIST),
1 Oryong-dong, Buk-gu, Gwangju 500-712,
Republic of Korea
e-mail: korwairs@gist.ac.kr

H. S. Kim
e-mail: hskim@gist.ac.kr

K. H. Lee
e-mail: khlee@gist.ac.kr

Keywords Mesh simplification · Curvature · PN triangle

1 Introduction

The use of complex polygonal models has been steadily increased in various application areas such as scientific visualization, finite element analysis, collaborative design, and computer-aided design (CAD)/computer-aided manufacturing (CAM). Especially, applications in real-time rendering of polygonal models have been increasing rapidly. Although the capability of graphics hardware has improved significantly, it is still limited in handling heavy polygonal models. Moreover, these models become more complex due to additional attributes such as color, curvature, and material property that are frequently used for some applications like photo realistic rendering and physics-based simulation. So, over the past few years, many different methods for surface simplification have been proposed to address these problems.

A common approach for simplification methods is to define an error metric $C(\mathbf{v})$ that can be calculated by estimating the error between the original and the approximated model. But the error metric $C(\mathbf{v})$ cannot guarantee the error bound and preservation of features. In this research, the curvatures are used as an additional attribute to preserve features. First, we calculate various curvatures and represent them as a color attribute on each vertex. Then the image processing technique using the vertex color is applied to reduce noise while preserving features. Finally, these filtered values are applied as color attributes for the error metric $C(\mathbf{v})$.

Additionally, the optimal positioning method is used to reduce the accumulated error during the process of simplification.

1.1 Nomenclature

The brief introduction of nomenclature is described in Table 1.

Table 1 Nomenclature

Terminology	
QEM	Quadric error metric (Section 2.2.2)
Extended QEM	QEM with attributes (Section 2.2.2)
Curvature color	Color mapped curvature (Section 3.1)
PN triangle	Point normal triangle (Section 3.3.1)
Notation	
$\mathbf{v}_i$	i-th vertex of mesh
$\mathbf{f}_i$	i-th triangle of mesh
$\mathbf{n}$	Normal vector
$N_1(\mathbf{v}_1)$	One-ring neighboring vertices of $\mathbf{v}_i$
$\mathbf{e}(\mathbf{v}_i, \mathbf{v}_j)$	An edge connecting vertices $\mathbf{v}_i$ and $\mathbf{v}_j$
$\mathbf{A}$	$\mathbf{n}\mathbf{n}^T$ (Section 2.2.2)
A	Area of triangle mesh
$C(\mathbf{v})$	Simplification error metric
A_{voroni}	Voronoi region
α_{ij}, β_{ij}	Two angles opposite the edge $\mathbf{e}(\mathbf{v}_i, \mathbf{v}_j)$
$K(\mathbf{v}_i)$	Mean curvature normal
A_{mixed}	Mixed region
$Q(\mathbf{v})$	Quadric error on vertex $\mathbf{v}$
κ_1, κ_2	Maximum and minimum curvature
$\kappa_{\text{H}}, \kappa_{\text{G}}$	Mean curvature and Gaussian curvature
$\Delta(\mathbf{v}_i)$	$\kappa_H^2(\mathbf{v}_i) - \kappa_G(\mathbf{v}_i)$
$\mathbf{u}$	Pixel at the coordinate $\mathbf{u} = (x, y)$
$I(\mathbf{u})$	Input pixel value on $\mathbf{u}$
$\hat{I}(\mathbf{u})$	Output pixel value on $\mathbf{u}$
$N(\mathbf{u})$	Neighboring pixel of $\mathbf{u}$
$W_{\text{c}}(x)$	Closeness smoothing filter
σ_{c}	Standard deviation of closeness
$W_{\text{s}}(x)$	Similarity smoothing filter
σ_{s}	Standard deviation of similarity
$W_\kappa(x)$	Curvature smoothing filter
σ_κ	Standard deviation of curvature
$W_{\text{E}}(x)$	Edge length smoothing filter
σ_{E}	Standard deviation of edge length
$\mathbf{b}_{ijk}$	Coefficients of control points
$\mathbf{n}_{ijk}$	Coefficients of the normal component
ϕ	Normal variance
R	Curvedness
γ	Compactness
P_1	Cost for edge collapse $e(\mathbf{v}_i, \mathbf{v}_j)$
P_2	Cost for edge collapse $e(\mathbf{v}_j, \mathbf{v}_i)$
P_{total}	$P_1 + P_2$
W_ϕ	Weight for normal variance
W_R	Weight for curvedness
W_γ	Weight for compactness

2 Related work and background

2.1 Related work

Many researchers have worked on mesh simplification problems. Schroeder et al. proposed the simplest simplification method called the vertex decimation [1]. It estimates the approximation error iteratively using the distance between the vertices and average planes. Some of the vertex decimation methods use more accurate error metric, for example, the localized Hausdorff error [2, 3]. However, these methods delete all associated triangles of a vertex, so we need to perform additional retriangulation.

Recently, the iterative contraction algorithm has become the most popular method, which is relatively stable compare to vertex decimation. It continuously collapses vertex pairs ($\mathbf{v}_i$, $\mathbf{v}_j$) and generates new vertices ($\mathbf{v}_{new}$) at each iteration until the desired error is satisfied. Edge collapse [4] is originally proposed by Hoppe et al. and further developed by many researchers [5]. There are two issues which need to be solved in the edge collapse-based simplification method. It is important to come up with the error metric $C(\mathbf{v})$ that is used to determine the priority of collapse. The other is to calculate the location of the newly generated vertex $\mathbf{v}_{new}$ as a result of edge collapse.

An enhanced quadric error metric (QEM) was developed by Garland and Heckbert [6]. It defines the error as the squared distance from a vertex to the planes sets. This method is fast and accurate. However, it requires 4×4 symmetric matrix to be present for each vertex. After an edge collapse, the vertex position minimizing the quadric error is found where the gradient of the quadric error metric equals zero. QEM has been used for improving various simplification methods [7, 8].

The mesh models often have other attributes in addition to the coordinate values on each vertex such as texture coordinate, vertex color, and normal. QEM has been extended to accommodate these attributes [9–11]. More generalized method was proposed by Garland and Zhou for simplifying simplicial complexes of any type embedded in Euclidean spaces of any dimension [12].

An image-based approach was proposed by Cohen et al. that used an error caused by pixel operation in image space [13]. It also accounts for multiple attributes and decouples a surface position from color and curvature information and storing the latter quantities in the texture and normal maps. The geometry is simplified by a common simplification method, while the texture and normal maps are filtered by graphics hardware. The

quality of resulting models is visually very close to their original ones. Lindstrom and Turk used an image-based metric that compares rendered images of the original model and those of the simplified model. Geometric criteria were not considered in their method, but visual quality was improved significantly [14].

On the other hand, quite different method was proposed by David et al. [15]. In their method, the input model is divided into a set of nonoverlapping parts by solving discrete partitioning optimization problem. More compact representations are proposed by using either additional proxy types [16] or a new approximate error metric [17]. These methods can effectively reduce the accumulated error, but heavy computation is needed for the optimization of a set of local proxies.

We reviewed various methods of mesh simplification. These methods have been widely used in various CAD applications [18, 19], parameterizations [20–22], and computer graphics applications [23–25]. In order to satisfy various needs of mesh simplification applications, the methods must be able to handle multiple attributes included in the mesh model. They need to preserve features of a specified error level within a reasonable computation time even in the highly approximated models. In this study, we propose an improved simplification method that uses curvature values as additional attributes and also reduces the accumulated error by locally optimizing the position of new vertices during edge collapse.

2.2 Background

2.2.1 Gaussian curvature and mean curvature

In surface modeling, the curvature is a useful differential quantity of the discrete triangle mesh. Especially, the Gaussian curvature and the mean curvature are frequently used to define the properties of an arbitrary surface. The Gaussian curvature is the product of the two principle curvatures, while the mean curvature is the mathematical average of them. Usually, the mean curvature is calculated by using Laplace–Beltrami operator, and similarly, the Gaussian curvature is estimated by using Gauss–Bonnet theorem in discrete geometry. Then the principle curvatures are acquired using both of them. For curvature calculation, the barycentric finite-volume area is widely used for its simplicity. However, this method often gives unstable results for irregular polygonal models. We use more generalized curvature calculation method which uses both Voronoi finite-volume area and barycentric finite-volume area [26].

Voronoi region (see Fig. 1a) can be calculated by solving Eq. 1, and the mean curvature normal can be calculated by using Eq. 2. The mean curvature value is easily computed by taking the half the magnitude of the normal as described in Eq. 3.

$$A_{\text{voronoi}} = \frac{\sum_{j\in N_1(\mathbf{v}_i)}(\cot\alpha_{ij} + \cot\beta_{ij})\|\mathbf{v}_i - \mathbf{v}_j\|^2}{8} \tag{1}$$

where the A_{voronoi} is Voronoi region and α_{ij} and β_{ij} are two angles opposite to the edge $\mathbf{e}(\mathbf{v}_i, \mathbf{v}_j)$ that is shared by two triangles (see Fig. 1b).

$$K(\mathbf{v}_i) = \frac{\sum_{j\in N_1(\mathbf{v}_i)}(\cot\alpha_{ij} + \cot\beta_{ij})(\mathbf{v}_i - \mathbf{v}_j)}{2A_{\text{mixed}}}, \tag{2}$$

$$A_{\text{mixed}} = \sum_{j\in N_1(\mathbf{v}_i)} A_j$$

$$A_j = \begin{cases} A_j = \text{Voronoi region, if } \mathbf{f} \text{ is nonobtuse} \\ A_j = \text{area}(\mathbf{f})/2, \quad \text{if } \mathbf{f} \text{ is obtuse at } \mathbf{v}_i \\ A_j = \text{area}(\mathbf{f})/4, \quad \text{if } \mathbf{f} \text{ is obtuse at } \mathbf{v}_j \end{cases}$$

where $\mathbf{f}$ is triangle from one-ring neighborhood $N_1(\mathbf{v}_i)$ of the vertex $\mathbf{v}_i$.

$$\kappa_H(\mathbf{v}_i) = \frac{1}{2}\|K(\mathbf{v}_i)\|^2 \tag{3}$$

Similarly, the Gaussian curvature over the polygonal mesh can be calculated as expressed in Eq. 4. The value will be zero for any flat surfaces and cylindrical shapes.

$$\kappa_G(\mathbf{v}_i) = \frac{\left(2\pi - \sum_{j\in N_1(\mathbf{v}_i)}\theta_j\right)}{A_{\text{mixed}}} \tag{4}$$

2.2.2 Extended QEM

The original QEM uses the squared distance between a vertex and the planes [6]. The standard representation of a plane is $\mathbf{n}^T\mathbf{v} + d = 0$, where $\mathbf{n} = [a, b, c]^T$ is a unit normal vector and d is a scalar constant. The

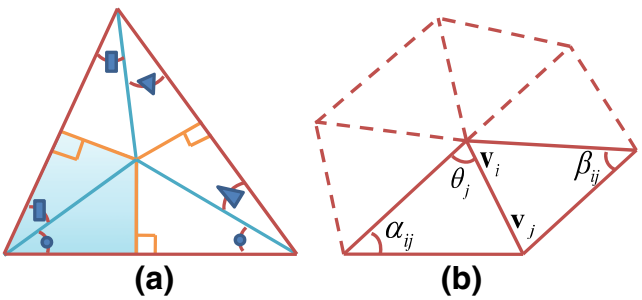

Fig. 1 Voronoi region and one-ring neighborhood: **a** Voronoi region on a nonobtuse triangle, **b** one-ring neighborhood and angles

squared distance between a plane and a given vertex $\mathbf{v} = [x, y, z]^T$ is expressed as $D^2(\mathbf{v}) = (\mathbf{n}^T\mathbf{v} + d)^2$. Therefore, the equation can be written as follows:

$$D^2(\mathbf{v}) = (\mathbf{v}^T(\mathbf{n}\mathbf{n}^T)\mathbf{v} + 2d\mathbf{n}^T\mathbf{v} + d^2) \quad (5)$$

The above equation defines the quadric $\mathbf{Q}$ as a triple $\mathbf{Q} = (\mathbf{A}, \mathbf{b}, c)$, where $\mathbf{A}$ is a 3×3 symmetric matrix, $\mathbf{b}$ is three-column vector, and c is a scalar. The quadric assigns a value $\mathbf{Q}(\mathbf{v})$ to every vertex by the second order equation as below.

$$\mathbf{Q} = (\mathbf{A}, \mathbf{b}, c) = ((\mathbf{n}\mathbf{n}^T), (d\mathbf{n}), d^2) \quad (6)$$

These three-dimensional quadric equations can be extended to n-dimension by using Gram–Schmidt orthogonalization [9]. It means that the vertex information of QEM can be extended from $[x, y, z]^T$ to $[x, y, z, r, g, b]^T$ to include color data and also to n-dimension (R^n). Extended QEM has an error metric similar to Eq. 6. However, the dimension of $\mathbf{A}$ is 6×6 symmetric matrix, $\mathbf{b}$ is a six-column vector, and c is a scalar value when the vertex color is used as an attribute value.

After the edge collapse, the optimal position $\mathbf{v}_{\text{new}}$ that minimizes $\mathbf{Q}(\mathbf{v}_{\text{new}})$ is determined. Since $\mathbf{Q}(\mathbf{v}_{\text{new}})$ is a quadric, finding its minimum is a linear problem. The minimum occurs where the gradient ($\nabla\mathbf{Q}(\mathbf{v}) = 2\mathbf{A}\mathbf{v} + 2\mathbf{b}$) is zero. By solving the gradient, the optimal position is calculated as $\mathbf{v}_{\text{new}} = -\mathbf{A}^{-1}\mathbf{b}$.

3 The proposed mesh simplification method

The mesh simplification proposed in this paper uses the curvature color. We first calculate the curvatures to detect features, but the initial values of them are inappropriate for direct use, so a filtering method is applied to the curvatures that are expressed as vertex color. Then the filtered curvature values are used as the error metric. Finally, the simplification operation with optimal positing is performed to reduce geometric error and to preserve features during edge collapse.

As shown in Fig. 2, the overall procedure consists of four steps: the calculation of curvatures, bilateral filtering of curvature colors, setting up the error metric with filtered curvature colors, and the simplification with optimal positioning.

The priority of edge collapse is determined by QEM using the curvature color, and then the candidates for optimal position are generated around neighboring triangles of the collapsing edge. Finally, the optimal position is selected using the optimal positioning function. The optimal positioning process is iterated until the number of current vertex is equal to the desired number of vertices.

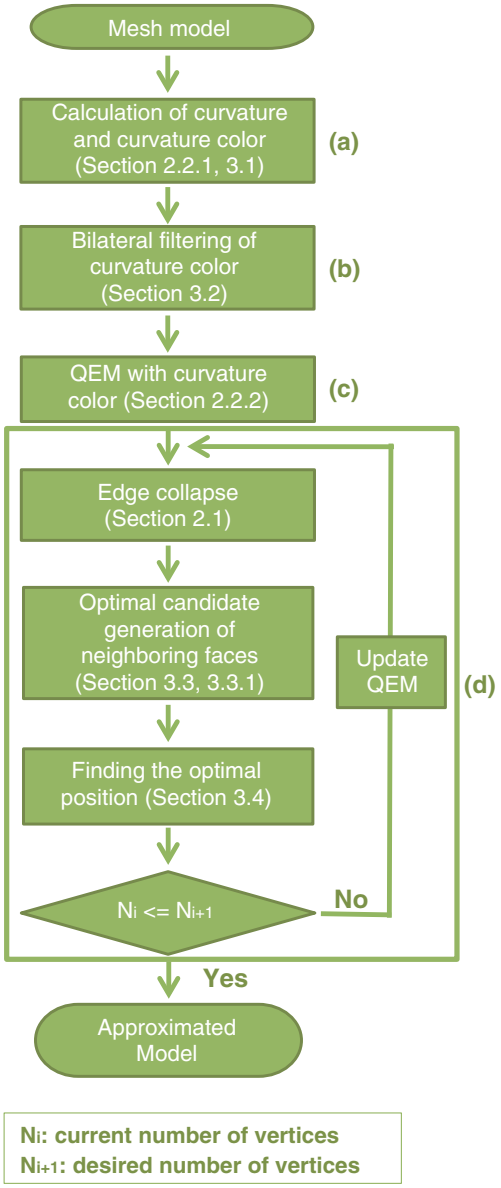

Fig. 2 The flowchart of the proposed method: **a** the calculation of curvature and converting it to color, **b** bilateral filtering of the curvature color, **c** estimation of the error metric using filtered curvature colors, **d** simplification with optimal positioning

The pseudocode for the entire process is described in Algorithm 1 for further understanding. In Algorithm 1, the procedure of the proposed method is explained under corresponding comments.

Accurate curvature value and its appropriate application in QEM are essential to guarantee the reliable determination of the collapsing priority. In this research, we use averaging Voronoi cells in the calculation of curvatures because they can accommodate

Algorithm 1 The pseudocode of the proposed method

```
//input
Vertex: v, Face f, Edge e, Mesh M
//One-ring neighborhood information of incident vertex v
V(v), F(v), E(v)
//Calculate curvatures and curvature color
//(Section 2.2.1, Section 3.1)
Curvature C(v): Calculate curvature(M, V(v))
Curvature color of the curvature C_m: Convert curvature to
color(C_v, v)
//Bilateral filtering of curvature color (Section 3.2)
for each vertex v do
  for each neighboring vertex V(v) incident on v do
    Update C_m: Bilateral filtering(v, V(v))
  end for
end for
for each vertex v do
  v[x, y, z, r, g, b]: Add curvature color to vertex(C_m, v)
end for
//Apply filtered color map of curvature associated vertex
//v[x,y,z,r,g,b] to extended QEM (Section 2.2.2)
QEM on vertex v: QEM(v)
V(f): vertices of face f
for each face f do
  for each neighboring vertex V(f) incident on f do
    QEM(v): Calculate extend QEM(f, v)
  end for
end for
//Determination of the edge collapse priority
//(Section 2.2.2)
for each vertex v do
  for each neighboring edge E(v) incident on vertex v do
    Edge collapse priority EC(e): Calculate edge collapse
    priority(e)
  end for
end for
//Determination of the optimal positioning
//three parameter: compactness, curvedness, normal varia-
tion
//(Section 3.2, 3.3, 3.4)
while Current number of vertices ≠ Desired number of
vertices do
  for each edge e of EC(e) do
    Optimal candidates OCV(e): Make PN Triangle(e)
    for each optimal position candidates v_opc of OCV do
      CVD(v_opc): Calculate curvedness(v_opc)
      CPT(v_opc): Calculate compactness(v_opc)
      NV(v_opc): Calculate normal variance(v_opc)
      OPC(v_opc): Calculate optimal positioning cost
      (CVD, CPT, NV)
    end for
    Collapse edge(e)
    Set optimal position(OPC(e))
    Update QEM(e)
    Update collapse priority(e)
  end for
end while
```

the consistent representation of the first- and second-order differential properties. Then the curvatures are normalized and categorized to assign appropriate color.

3.1 Curvature color

The proposed method uses Eqs. 3 and 4 to calculate Gaussian curvature κ_G and mean curvatures κ_H. The principle curvatures can be calculated using these two curvatures.

$$\kappa_1(\mathbf{v}_i) = \kappa_H(\mathbf{v}_i) + \sqrt{\Delta(\mathbf{v}_i)} \tag{7}$$

$$\kappa_2(\mathbf{v}_i) = \kappa_H(\mathbf{v}_i) - \sqrt{\Delta(\mathbf{v}_i)} \tag{8}$$

where $\Delta(\mathbf{v}_i)$ is $\kappa_H^2(\mathbf{v}_i) - \kappa_G(\mathbf{v}_i)$.

After the calculation of various curvatures, they are normalized and classified for coloring the vertices as shown in Fig. 3.

In this research, we can directly use the curvature value as an attribute value in the extended QEM. However, the range of curvature values differ significantly depending on the geometry of the model. Although normalized values are used, there are limitations when noisy data are used shown in Fig. 4e. Figure 4d–f shows that the curvature color can detect features better than gray-level curvatures especially for the scanned data that are often noisy. But for noiseless data as shown in Fig. 4a–c, both gray-level curvatures and the curvature color can detect features well. The use of the curvature color is emphasized in this research due to this finding. During simplification, however, many difficulties occur in directly applying the curvature color as the attribute value due to its discontinuity on the surface. The curvature color needs to be smoothed out in the smooth-shaped areas, while the features need to be preserved in highly curved regions. The details discussed in the sections below.

3.2 Feature-preserved smoothing of the curvature color by bilateral filtering

The curvature color is quite sensitive in representing the local features of the model because of its high resolution. Furthermore, they often show the blurred boundary at highly curved areas. Too many vertices tend to have the similar colors for the sharp features. On the other hand, the color often varies significantly for a smooth area because of the sensitiveness of curvatures. For successful simplification, most vertices need to maintain its continuity of the color while preserving the color of sharp features.

We applied bilateral filtering [27] to the curvature color by which we can improve the smoothness of the

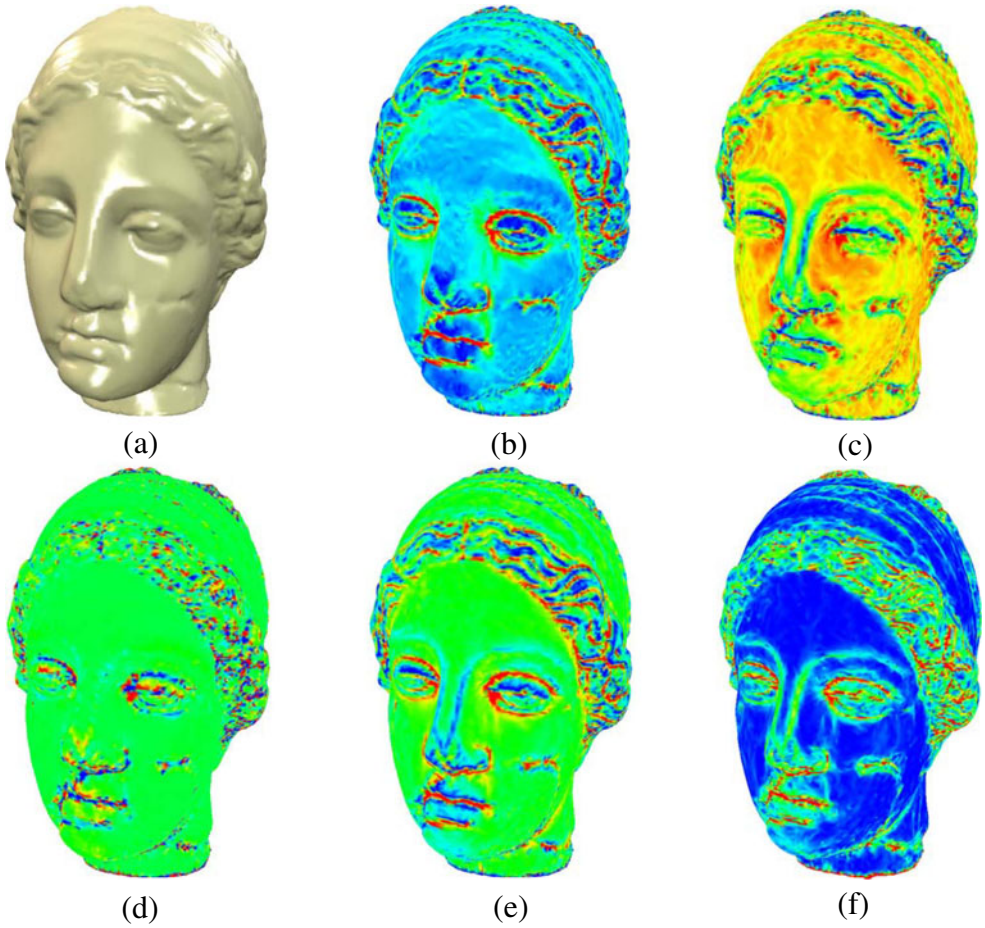

Fig. 3 Samples of different curvature colors: **a** an original model, **b** maximum curvature color, **c** minimum curvature color, **d** Gaussian curvature color, **e** mean curvature color, **f** curvedness color

color at the low curved area and the sharpness of the color for highly curved area as illustrated in Fig. 5. The bilateral filter has been used for generating smooth images while preserving features in image processing applications. The method is noniterative and simple. If the input image is $I(\mathbf{u})$, the bilateral filter at the coordinate $\mathbf{u} = (x, y)$ can be defined as follows:

$$\hat{I}(\mathbf{u}) = \frac{\sum_{\mathbf{p}\in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|) W_s(|I(\mathbf{u})-I(\mathbf{p})|) I(\mathbf{p})}{\sum_{\mathbf{p}\in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|) W_s(|I(\mathbf{u})-I(\mathbf{p})|)}$$

$$W_{\mathrm{c}}(x) = e^{-\frac{x_{\mathrm{c}}{}^2}{2\sigma_{\mathrm{c}}{}^2}},\ W_{\mathrm{s}}(x) = e^{-\frac{x_{\mathrm{s}}{}^2}{2\sigma_{\mathrm{s}}{}^2}} \tag{9}$$

where $N(\mathbf{u})$ is the neighbor pixel of $\mathbf{u}$. The closeness smoothing filter $W_{\mathrm{c}}(x)$ is a standard Gaussian filter with parameter σ_{c}, and the similarity weight is determined by the weight function $W_{\mathrm{s}}(x)$ with parameter σ_{s} to preserve features. Denote that pixel $\mathbf{u}$ depends not only on the spatial distance $\|\mathbf{p}-\mathbf{u}\|$ but also on the similarity $\|I(\mathbf{u})-I(\mathbf{p})\|$. The efficiency of the filter is determined by the parameter σ_{c} and σ_{s}. The parameter σ_{c} defines the size of the spatial neighborhood used to filter a pixel, and σ_{s} controls the weight of adjacent pixels due to color difference. In practice, these parameters are heuristically determined depending upon applications.

Some authors have used the bilateral filter on a surface model [28, 29]; likewise, the bilateral filter for the curvature color is proposed as follows:

$$\hat{\kappa}(\mathbf{v}_i) = \frac{\sum_{\mathbf{v}_j\in N_1(\mathbf{v}_i)} W_{\mathrm{E}}(\|\mathbf{v}_i-\mathbf{v}_j\|) W_\kappa(|\kappa(\mathbf{v}_i)-\kappa(\mathbf{v}_j)|)\kappa(\mathbf{v}_i)}{\sum_{\mathbf{v}_j\in N_1(\mathbf{v}_i)} W_{\mathrm{E}}(\|\mathbf{v}_i-\mathbf{v}_j\|) W_\kappa(|\kappa(\mathbf{v}_i)-\kappa(\mathbf{v}_j)|)}$$

$$W_\kappa(x) = e^{-\frac{x_\kappa{}^2}{2\sigma_\kappa{}^2}},\ W_{\mathrm{E}}(x) = e^{-\frac{x_{\mathrm{E}}{}^2}{2\sigma_{\mathrm{E}}{}^2}} \tag{10}$$

where $N_1(\mathbf{v}_i)$ is one-ring neighbor vertices of vertex $\mathbf{v}_i$. The edge length is used as the closeness smoothing filter $W_{\mathrm{E}}(x)$ with the parameter σ_{E}, and the similarity weight is described by $W_\kappa(x)$ with the parameter σ_κ. The curvature $\hat{\kappa}(\mathbf{v}_i)$ has more influence on the closer vertices and the one having similar curvature value.

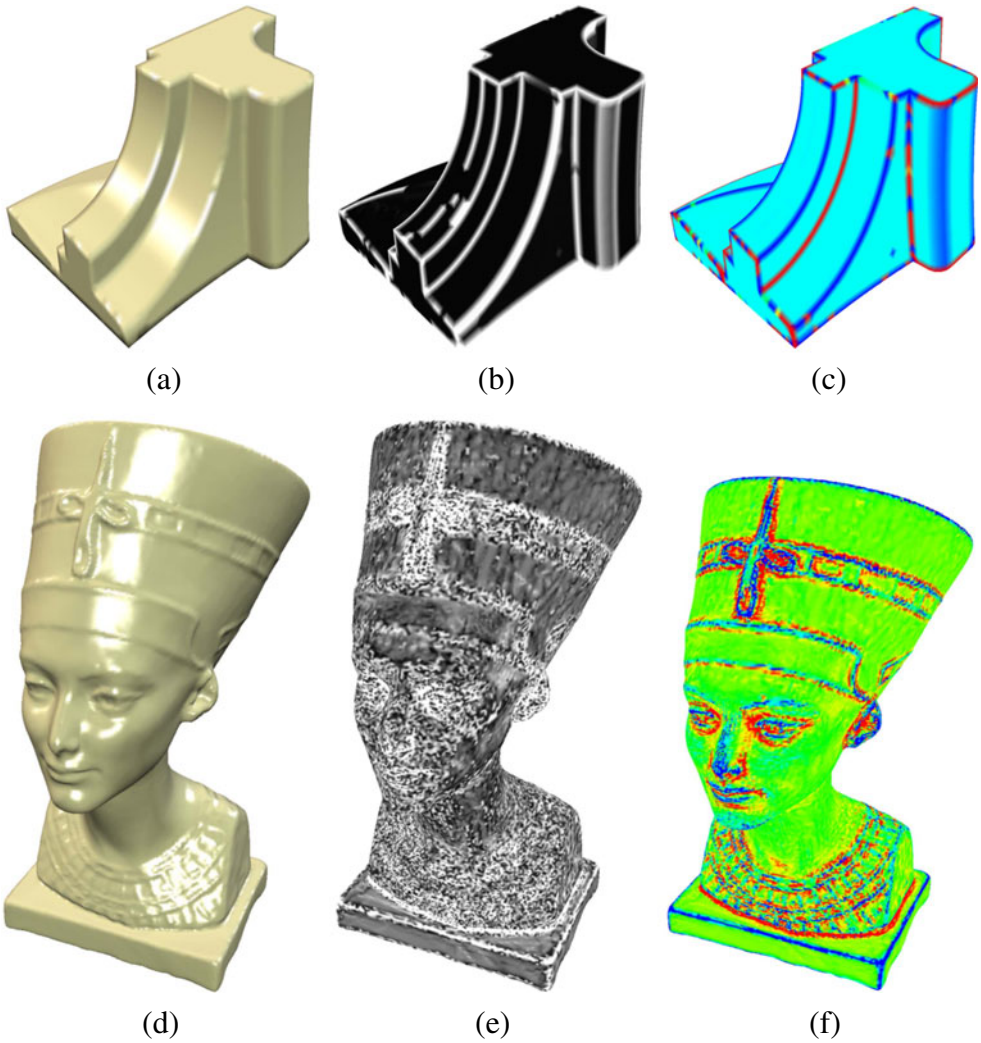

Fig. 4 Two examples showing difference between gray-level curvature and curvature color: **a** an original model, **b** gray-level Gaussian curvature, **c** Gaussian curvature color, **d** an original model, **e** gray-level mean curvature, **f** mean curvature color

The efficiency of the method also depends on two parameters σ_E and σ_κ. The initial value of σ_E is determined based on the edge length and the property of smoothing filter as $|2\sigma_E| > \|\mathbf{v}_i - \mathbf{v}_j\|$. Also the initial value of σ_κ is estimated as $|2\sigma_\kappa| > \|\kappa(\mathbf{v}_i) - \kappa(\mathbf{v}_j)\|$. Once the initial values of both parameters are determined, they are heuristically adjusted by the user depending on applications.

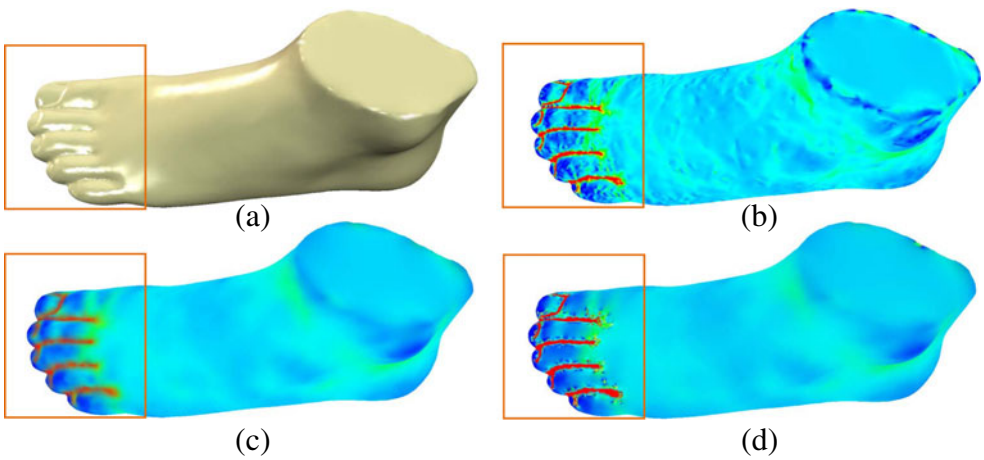

Fig. 5 Results of different smoothing methods: **a** an original model, **b** the color of maximum curvature, **c** Gaussian smoothing of maximum curvature, **d** bilateral smoothing of maximum curvature

The complex color feature can be expressed for the fine mesh when the vertex color is used. Consequently, it is required to approximate the color feature according to the simplification procedure. The filtering techniques can approximate color feature appropriately. However, the filtering often shows poor behavior in the simplification process. There are two regions, fine mesh around feature and coarse mesh in other regions to preserve a feature using the limited number of vertices. The feature preserving filtering of this kind often shows saw-like color pattern or inaccurate color calculation. Therefore, bilateral filtering is used only in the initial step, and the estimation of the curvature color is conducted to guarantee the feature preservation in the optimal positioning step (see Section 3.3.1).

3.3 Optimal positioning after edge collapse

Most simplification methods have concentrated on finding the cost function to determine the priority of decimation. However, the approximation error is continuously accumulated during the process of simplification. In highly simplified models, visual artifacts and geometric inconsistencies frequently occur due to the accumulated error. In this study, an optimal positioning method is used to reduce the accumulated error.

The overall procedure of optimal positioning is explained in Fig. 6. We use three steps to find the optimal position. As described in the figure, we first find neighbor triangles of an edge that is about to collapse. The candidates for the optimal position are generated by the vertices that are produced by subdividing the model using PN triangle [30]. The PN triangle is a curved point-normal triangle that will be described in the section below. Finally, the optimal position is calculated using

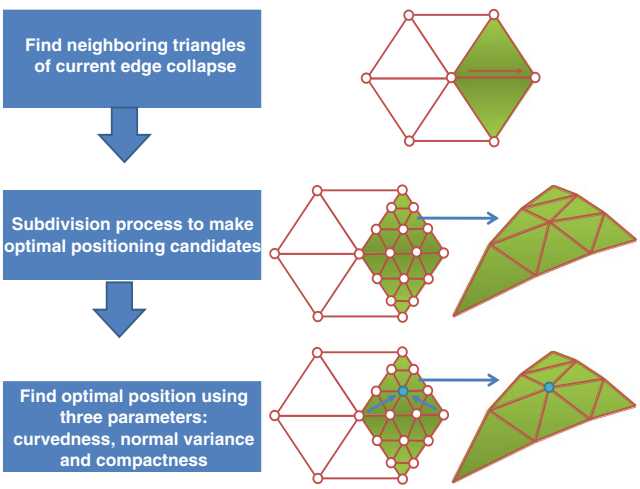

Fig. 6 The overall procedure of optimal positioning

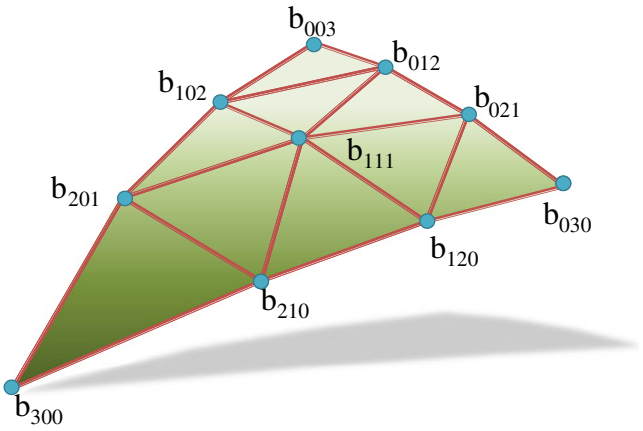

Fig. 7 The control points of a triangular Bézier patch arranged to form a control net(courtesy of [30])

three parameters that include the curvedness, normal variance, and compactness.

3.3.1 PN triangle and curvature color estimation

The geometry of a PN triangle is defined using a cubic Bézier patch. The patch matches the point and normal information at each vertex of a triangle. An interesting aspect of the curved PN triangle is the normal component of it. The geometry of a curved PN triangle is defined by a cubic patch b as shown in Fig. 7. The normal of a curved PN triangle can be expressed as a quadratic function of the position and normal coefficients $\mathbf{n}_{ijk}$ which is shown in Fig. 8.

The connectivity information of the triangles is not needed in this method because we can calculate the position and the normal of arbitrary point using control points $\mathbf{b}_{ijk}$ and normal coefficients $\mathbf{n}_{ijk}$. The graphics processing unit implementation of this technique is relatively easy, which is available in the ATI SDK

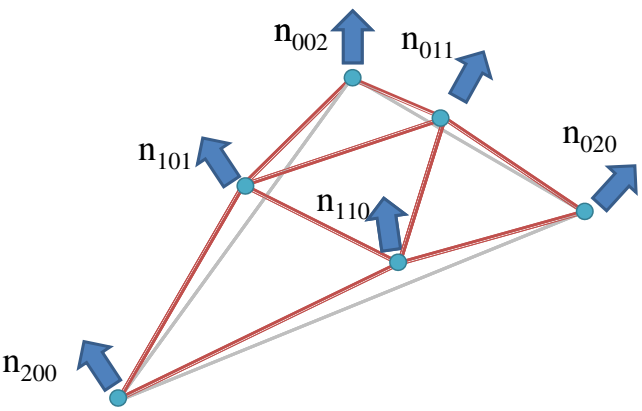

Fig. 8 The normal coefficients of a PN triangle (courtesy of [30])

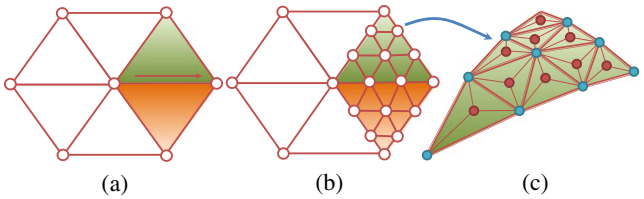

Fig. 9 PN-triangle subdivision: **a** edge collapse, **b** subdivision, **c** optimal position candidates

[31]. The curvature color of the optimal position candidates is estimated by bilinear interpolation using the curvature color of three vertices on the corresponding face [32].

3.3.2 The selection of an optimal position

The optimal position calculated by the extended QEM cannot guarantee the error bound ϵ. Moreover, the optimal position does not often exist when the 6×6 inverse matrix $\mathbf{A}^{-1}$ (see Section 2) is unable to be calculated or it contains imaginary values. Our proposed method uses multiple candidates for the optimal position to reduce these exceptions. If the PN patches are calculated, enormous number of candidates can be generated, but we only choose 32 points to reduce computation. The candidates are composed of points which are generated by subdivision and the center point of each subdivided triangle except for the rightmost and leftmost points with respect to the collapsed half edge (see Fig. 9c). Our method uses three parameters to estimate the optimal position among multiple candidates at each iteration (see Fig. 9). These parameters are used since they are sensitive to the change of geometry and relatively easy to compute.

3.3.3 Curvedness

The curvedness of a vertex can be defined using principle curvatures, but it can be defined as below using both Gaussian curvature and mean curvature since it contains the property of both.

$$R = \sqrt{{\kappa_{\mathrm{H}}}^2 - {\kappa_{\mathrm{G}}}^2} \tag{11}$$

where κ_{H} and κ_{G} are mean curvature and Gaussian curvature, respectively.

The curvedness value can also be expressed as vertex color in the same way as the curvature color as shown in Fig. 10. In most cases, the mean curvature and the curvedness represent overall features better than Gaussian curvature. The curvedness value changes smoothly except for highly curved areas, but the mean curvature value varies significantly in the smoothly changed curvature area.

The bilateral filtering is again applied to the curvedness color as shown in Fig. 11. The color map shows distinct difference between the curvedness color and the others. The curvedness color changes smoothly in most areas except for the sharp features, whereas the Gaussian curvature has dominant color only in highly curved areas, and the mean curvature shows distinguishable color difference in all areas.

In practice, the geometry and features are well preserved when the curvedness color is applied to the simplification method because the gentle color change areas lead to smooth results and the dominant color areas tend to preserve sharp features. In contrast, only sharp features are well preserved when the Gaussian curvature is applied, and both the smoothness and continuity are not guaranteed when the mean curvature is applied.

3.3.4 Compactness

The compactness is another measure that we use to determine the optimal position. We use this to improve the mesh quality by removing thin triangles. The thin triangles contain a very small angle with long sides such

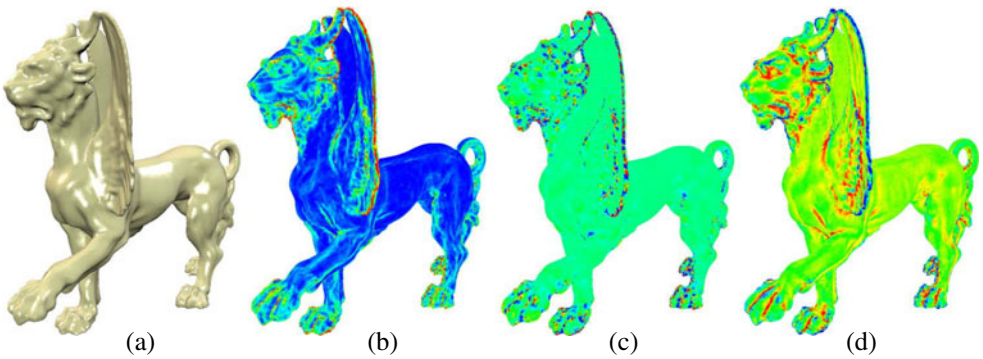

Fig. 10 The color representation of the curvedness: **a** an original model, **b** curvedness color, **c** Gaussian curvature color, **d** mean curvature color

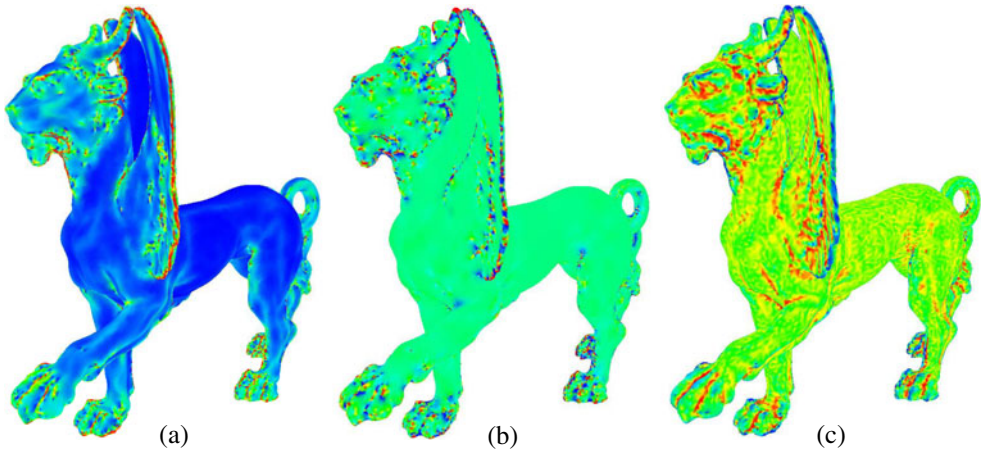

Fig. 11 The example of bilateral filtering of curvature colors and curvedness color: **a** the filtered curvedness color, **b** the filtered Gaussian curvature color, **c** the filtered mean curvature color

that they give undesirable effects for various applications. Guéziec defines the compactness as below [33].

$$\gamma = \frac{4\sqrt{3}A}{l_1^2 + l_2^2 + l_3^2} \tag{12}$$

where l is the length of edges and A is the area of the triangle. The compactness γ varies from 0 to 1 according to the shape of a triangle. It becomes close to 1 when the shape of the triangle is regular, and it approaches to 0 for the case of very bad thin triangles. In our proposed method, a threshold value for compactness is used to penalize candidate positions which generate a poor mesh. The threshold value is assigned heuristically by the user.

The proper compactness value varies according to the geometric characteristics and applications for a model. For a CAD model, it is desired to give relatively high compactness value. In contrast, a visualization model does not need to have high compactness values because it requires more faces to acquire similar visual results using high compactness models.

In the proposed method, the threshold of 0.3–0.4 is used for a CAD model and the threshold of 0.02–0.05 is used for a visualization model through various experiments. The geometric evaluation of the compactness is described in Frey and Borouchaki [34].

3.3.5 The normal variance

The normal variance is calculated by measuring the difference of the normal values of corresponding triangles before and after an edge collapse. It has been widely used due to its efficiency in preserving the geometry and removing nonmanifolds. Other methods usually skip the edge collapse and proceed to the next step when the value of normal variance exceeds the threshold. But the proposed method can contract edges without skipping the current step since it uses various candidates and it can avoid normal flipping.

Normal flipping can occur in some cases of half edge collapse as shown in Fig. 12, but the proposed algorithm can preserved the consistency of the normal variance by collapsing the edge $\mathbf{e}(\mathbf{v}_1, \mathbf{v}_2)$ to an optimal position $\mathbf{v}_{new}$. Smooth models can be generated when a well-defined normal variance is used because it can reduce the difference of normal variance between a triangle and its neighbor triangles.

3.4 The optimal positioning function

As we mentioned before, an optimal positioning function uses three measures: normal variance, curvedness, and compactness. These measures are also frequently used as the constraints in the other research which do not consider optimization. The main difference of using of these constraints compared to the other research is that the other research methods use these to determine whether to make a contraction or not, while our method

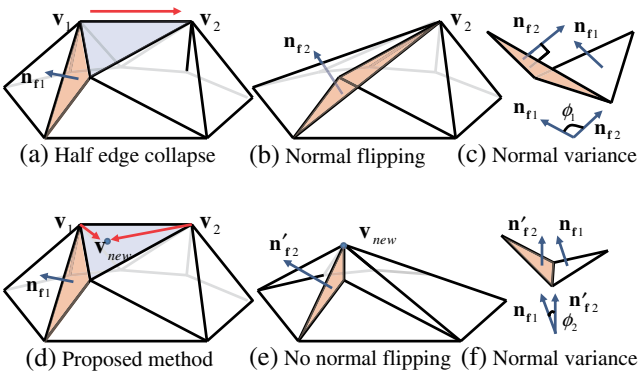

Fig. 12 The comparison of the normal variance and normal flipping between previous methods and the proposed method

uses these to find the optimal position among various candidates.

Consequently, the contractions are inevitably performed for the other methods even though the change of topology and geometry occurs when the model is highly simplified. Our proposed method can avoid these artifacts by using new vertices which minimize the destruction of geometry and topology. The optimal positioning function is defined using normal variance ϕ, curvedness R, and compactness γ. Three parameters are scalar values since they are normalized from 0 to 1. For example, the normal variance is angle value, and its unit is radian but it is normalized by dividing by the difference between its maximum and the minimum value. The optimal positioning function can be defined by linear combination of these parameters as follows:

$$\begin{aligned}
&P_1 = \textstyle\sum_i W_\phi(\phi_i - \bar{\phi})^2 + W_R(R_i - \bar{R})^2 + W_\gamma(\gamma_i - \bar{\gamma})^2 \\
&P_2 = \textstyle\sum_j W_\phi(\phi_j - \bar{\phi})^2 + W_R(R_j - \bar{R})^2 + W_\gamma(\gamma_j - \bar{\gamma})^2 \\
&P_1 = \text{Edge collapse}(\mathbf{e}(\mathbf{v}_i, \mathbf{v}_j)) \\
&P_2 = \text{Edge collapse}(\mathbf{e}(\mathbf{v}_j, \mathbf{v}_i)) \\
&P_{\text{total}} = P_1 + P_2 \\
&: \text{Normal variance } \phi, \text{ Curvedness } R, \text{ Compactness } \gamma
\end{aligned} \tag{13}$$

where $\bar{\phi}$, $\bar{R}$, and $\bar{\gamma}$ are the average of normal variance, curvedness, and compactness of collapsing edge, respectively, and W_ϕ, W_R, and W_γ are the weight of corresponding parameters. These weight factors are the coefficients for convex linear combination of these parameters. In this study, the initial value of each weight is set to 1/3. The user can provide different values for different applications while keeping the sum of these values equal to 1.

If a model is used only for visualization purpose, the user can lower the weight of compactness compared to the other two parameters. However, the control of the weight needs to be performed approximately within a certain level. For example, if the weight of the curvedness is given more emphasis than the others to reduce geometric error, the model cannot be used numerically because of the poorer quality and visual results affect numerical calculation according to the characteristics of discrete geometry. Similarly, extremely high weights for the other parameters can also produce anomalies. Practically, the largest weight a parameter should be less than twice the weight of the smallest parameter.

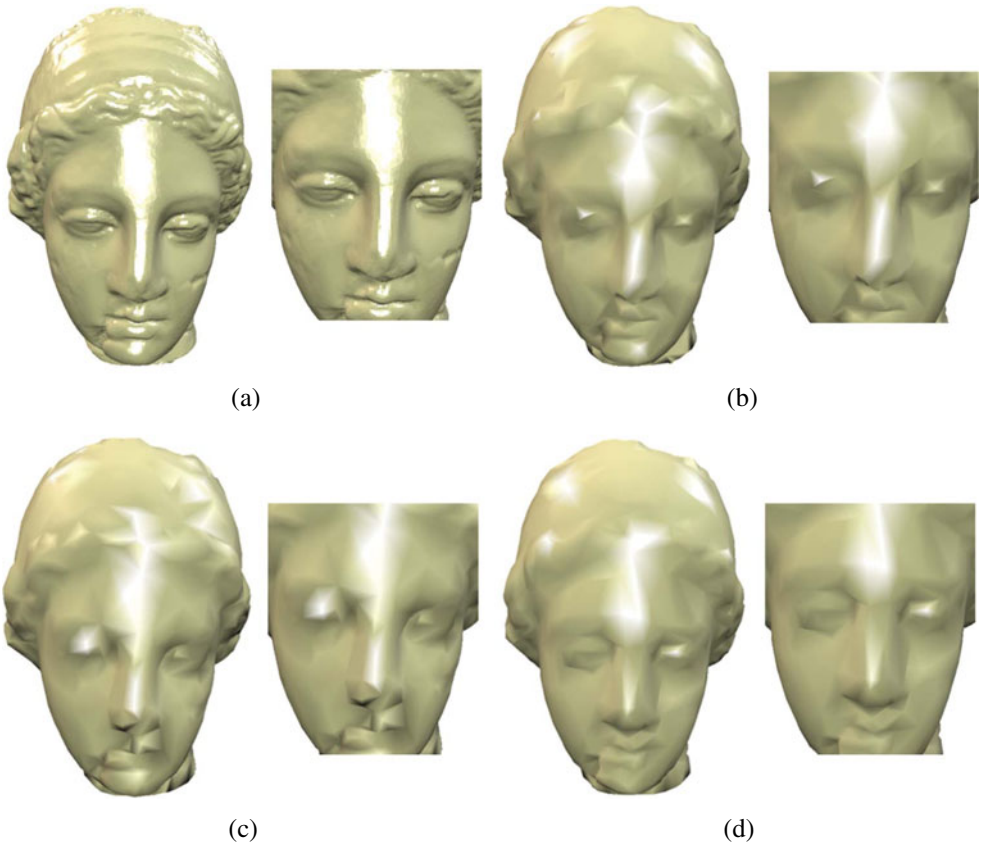

Fig. 13 Visual comparison of simplified IGEA models: **a** an original model (number of vertices, 50,000), **b** QEM (number of vertices, 1,500), **c** the roundness method (number of vertices, 1,500), **d** the proposed method (number of vertices, 1,500)

4 Experimental results

We demonstrated the performance of our method by comparing the results with other methods visually. We also compared the results using the normal distance between the original model and the approximated model. Figure 13 shows the visual comparison of our method with the results by QEM and the roundness method.

Both QEM and the roundness method are implemented using OpenMesh library [35]. The error metric of the roundness method is based on the triangle's roundness [36], and it is used as the constraints in the QEM method. For comparison, the IGEA model with 50,000 vertices is used as shown in Fig. 13. The model is reduced down to 1,500 vertices. As observed in the figure, the proposed method gives better results for the overall shape and for geometric features in highly curved areas such as the nose and the lips.

Another visual comparison is made using a foot model as shown in Fig. 14. This model is used to check the preservation of a smooth surface. The original foot model has 5,200 vertices, and it is simplified to 400 vertices by using the QEM, the roundness method, and the proposed method as well. The results are obtained by using the QEM; the roundness method shows relatively good results. However, the proposed method yields better results in preserving features.

The Nefertiti model has both complex geometry and flat regions as shown in Fig. 15. This model is chosen to test both the simple and complex geometry combined in a model. As shown in the figure, our proposed method yields better results not only for the sharp features but also in the flat areas.

The dragon model shown in Fig. 16 has extreme curved areas. This model is used to verify the stability of the proposed algorithm in handling very sharp features. As we observe from Fig. 16, the proposed method preserves sharp features better than the other methods.

The proposed method also can accommodate other attributes such as vertex color as shown in Fig. 17. In the figure, the original Hanbok model with 100,000 vertices has been reduced down to 2,000 vertices. All procedures are the same except using vertex color instead of the curvature color. Figure 17a, b shows the simplified results using QEM and the roundness method, respectively. As shown in the figure, the color feature of the model is not well preserved compared to extended QEM. The simplified result generated by extended QEM preserves color feature accurately, but the

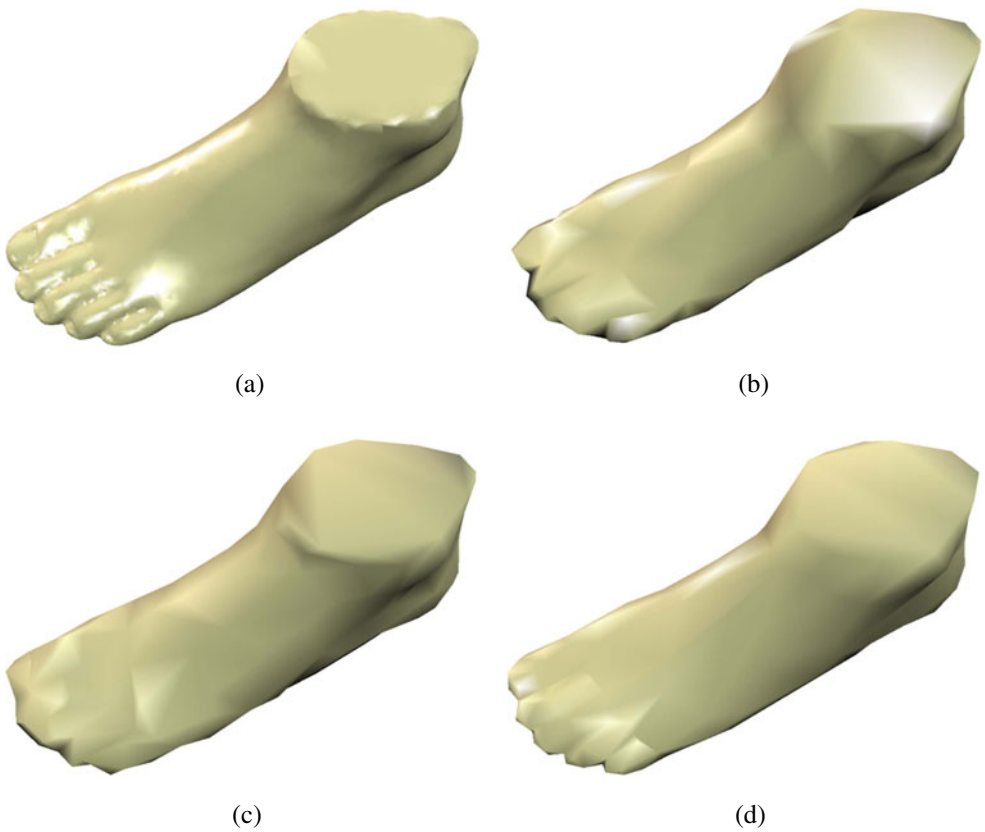

Fig. 14 Visual comparison of simplified foot models: **a** an original model (number of vertices, 5,200), **b** QEM (number of vertices, 400), **c** the roundness method (number of vertices, 400), **d** the proposed method (number of vertices, 400)

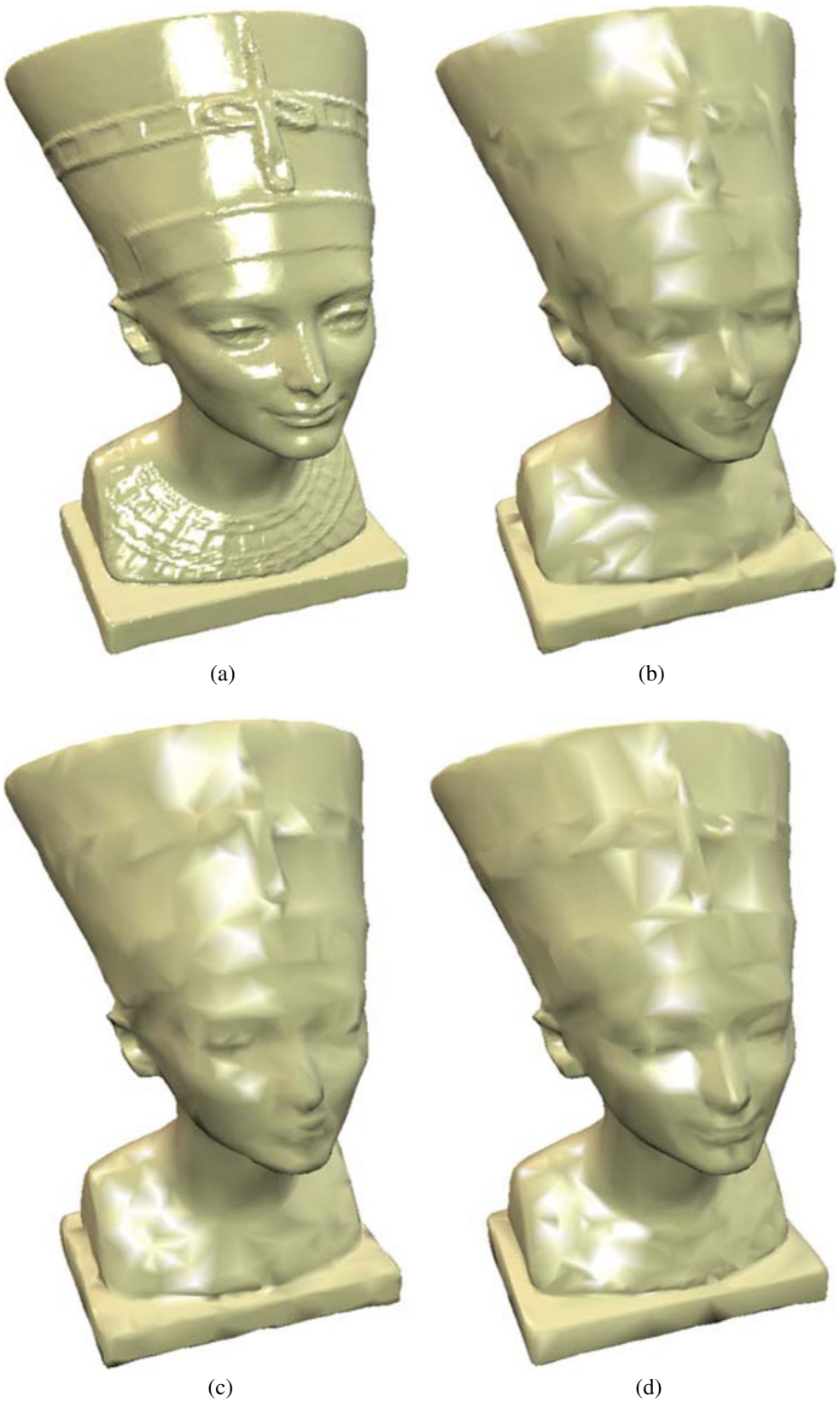

Fig. 15 Visual comparison of simplified Nefertiti models: **a** an original model (number of vertices, 90,000), **b** QEM (number of vertices, 2,000), **c** the roundness method (number of vertices, 2,000), **d** the proposed method (number of vertices, 2,000)

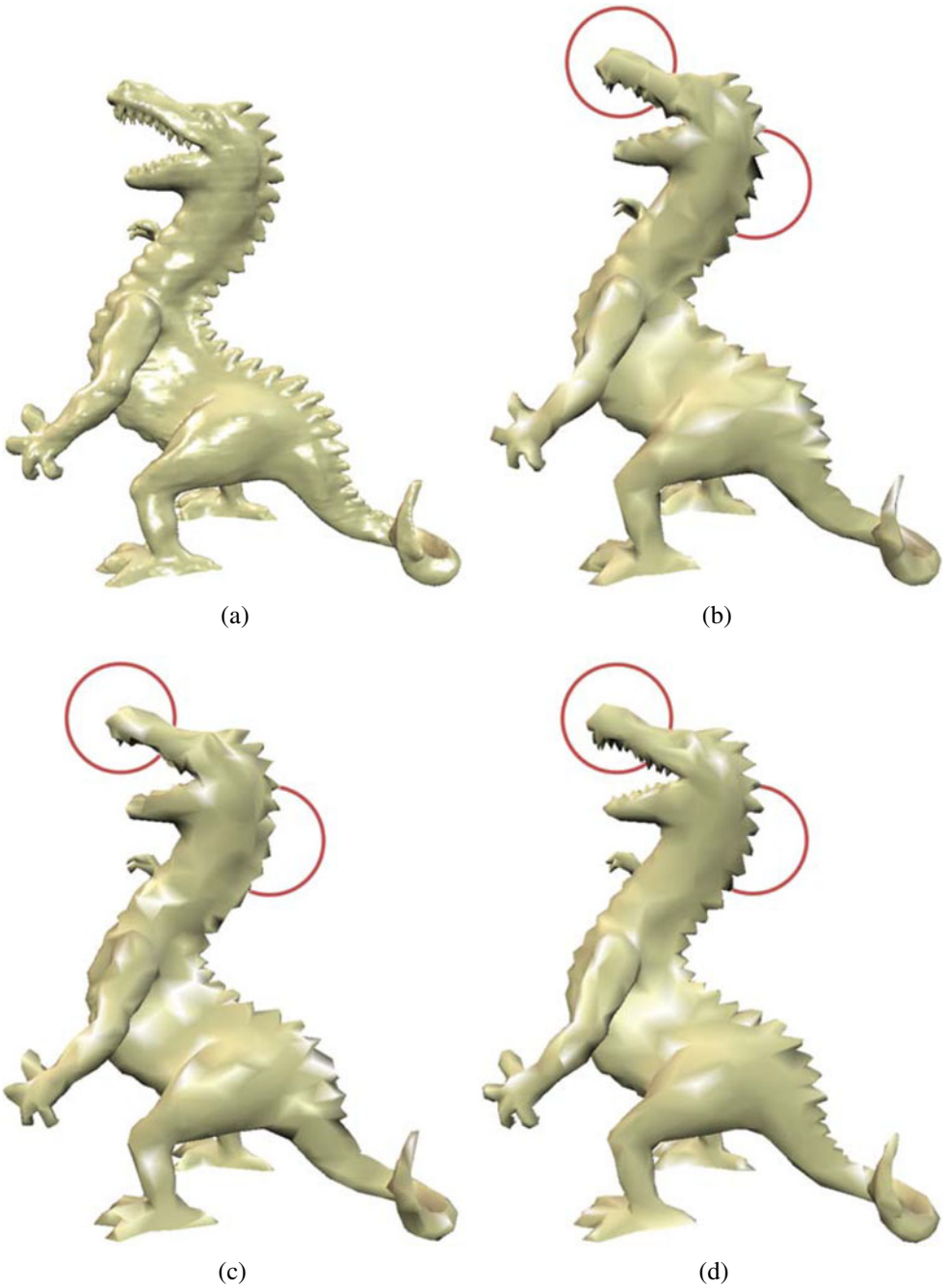

Fig. 16 Visual comparison of simplified dragon models: **a** an original model (number of vertices, 25,000), **b** QEM (number of vertices, 1,500), **c** the roundness method (number of vertices, 1,500), **d** the proposed method (number of vertices, 1,500)

geometric feature is not well preserved (see Fig. 17c). The proposed method can balance the preservation of geometric features and the reduction of color errors (see Fig. 17d).

The geometric error is also compared by calculating the deviations of the normal distance between the original and the approximated models using the same orientation and scale. The difference in geometric features and volume shrinkage can be estimated by calculating the normal distance between the faces of the original model and the corresponding faces of the approximated models. The calculation of normal distance is conducted by using commercial software Rapidform® (INUS Technology Inc.). It is widely used to calculate the normal distance error between two mesh models [37].

Table 2 shows the average errors of normal distances comparing with the original model for different simplification methods. Four models that are used for visual comparison are used again. The diagonal lengths of all models are normalized to 100 mm to check the error of the models with consistency.

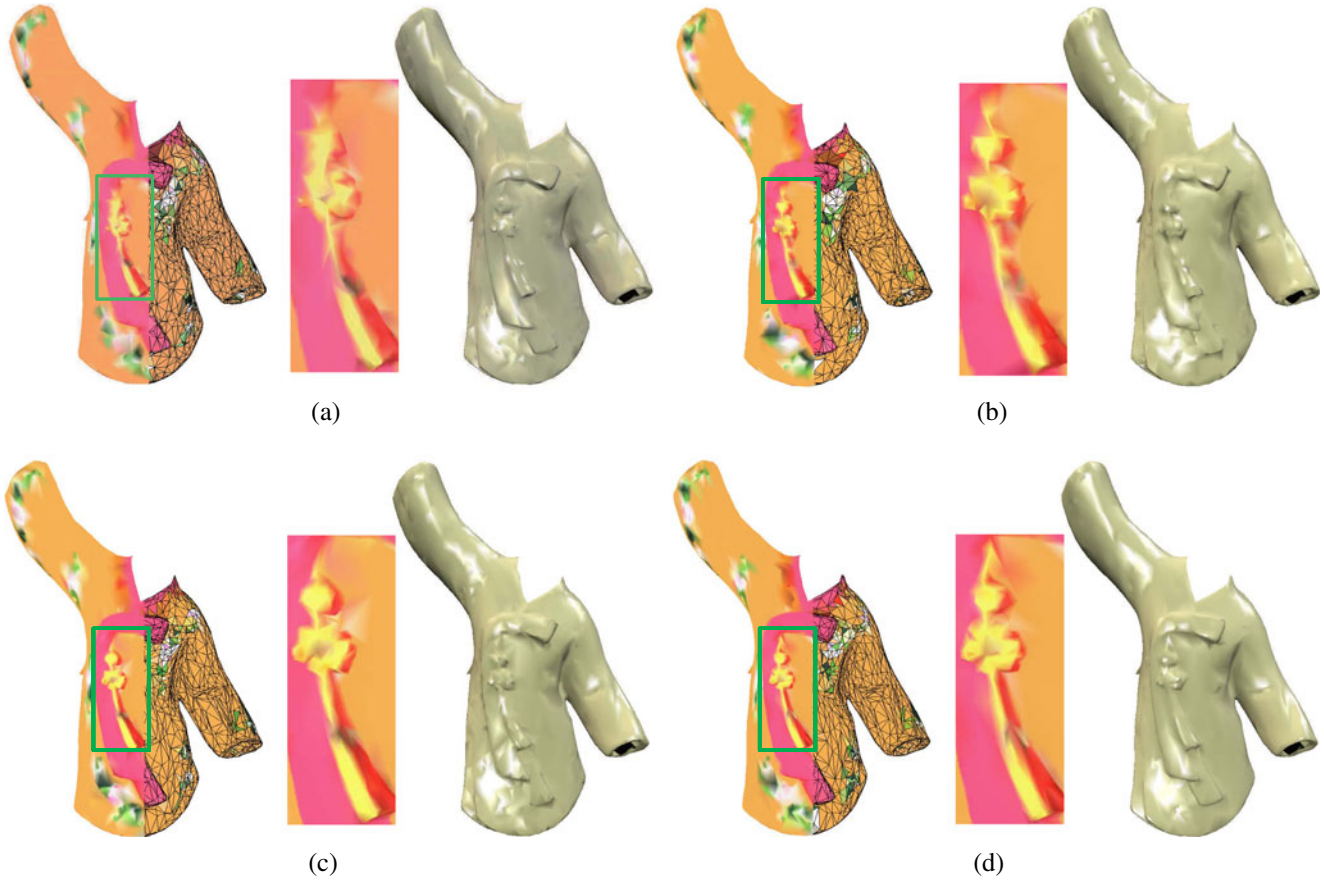

Fig. 17 Visual comparison of Hanbok model with vertex color: **a** QEM (number of vertices, 2,000), **b** roundness (number of vertices, 2,000), **c** extended QEM (number of vertices, 2,000), **d** the proposed method (number of vertices, 2,000)

As it is shown in the table, the proposed method gives better results. Note also that the error not only depends on the simplification ratio but also depends on the geometry of the model. For example, the simplification ratio of the dragon model is smaller than that of the IGEA and Nefertiti model, but the geometry error of the dragon model is bigger than others. It mainly is caused by the collapse of the sharp features.

In case of the foot model, the overall error is relatively bigger than others. This experiment shows the comparison of results when the model is highly simplified using previous methods and the proposed one.

The normal distance error is visualized using a color map as shown in Fig. 18. For color visualization, we normalize the color error range from 0 to 0.18762. As shown in the figure, the maximum error generally occurs in the highly curved areas.

Additionally, the comparison of the compactness is conducted. Thin triangles are inevitably generated

Table 2 Normal distance error

Model name (number of vertices of an original model)	Number of vertices of simplified models	Average normal distances error (mm)		
		QEM	Roundness	The proposed method
Foot	400	0.15806	0.18037	0.11724
(5,200)	200	0.21849	0.31440	0.15809
IGEA	2,000	0.07299	0.07661	0.05611
(50,000)	1,500	0.09153	0.09761	0.07145
Nefertiti	2,000	0.04634	0.05032	0.03001
(90,000)	1,500	0.05749	0.06310	0.03960
Dragon	2,000	0.11564	0.12592	0.08636
(25,000)	1,500	0.15106	0.14700	0.11161

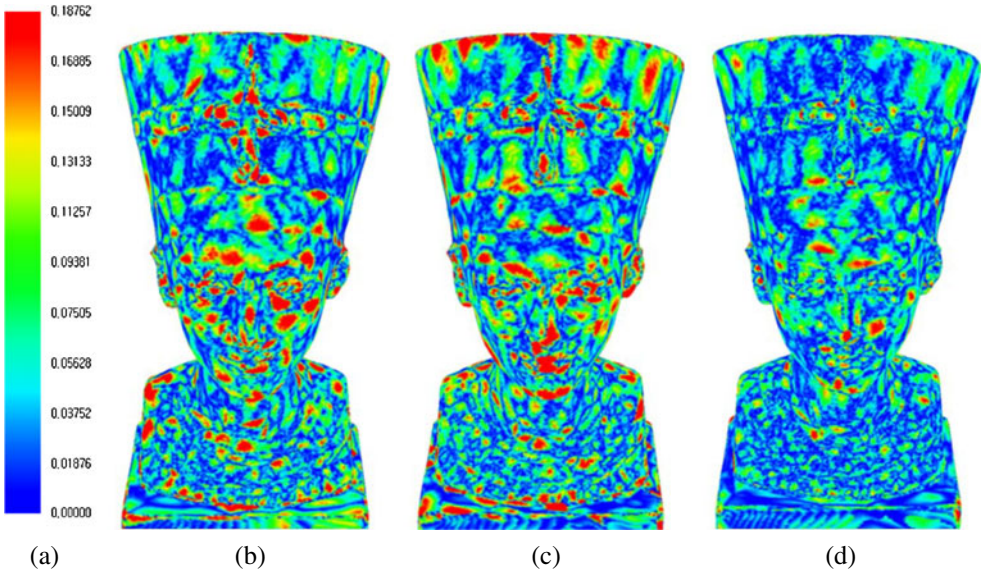

Fig. 18 The visualization of the normal distance error: **a** error range, **b** QEM, **c** roundness, **d** the proposed method

Table 3 The comparison of the compactness

Model name (number of vertices of an original model)	Number of vertices of simplified models	Compactness (standard deviation of compactness)		
		QEM	Roundness	The proposed method
Foot	400	0.6845 (0.2241)	0.6950 (0.2114)	0.6751 (0.2145)
(5,200)	200	0.6840 (0.2123)	0.6827 (0.2073)	0.6320 (0.2329)
IGEA	2,000	0.7029 (0.2186)	0.7128 (0.2089)	0.7156 (0.2111)
(50,000)	1,500	0.7062 (0.2167)	0.7136 (0.2062)	0.7139 (0.2144)
Nefertiti	2,000	0.6654 (0.2318)	0.6685 (0.2222)	0.6479 (0.2349)
(90,000)	1,500	0.6552 (0.2344)	0.6696 (0.2195)	0.6512 (0.2359)
Dragon	2,000	0.6920 (0.2175)	0.7045 (0.2074)	0.7044 (0.2090)
(25,000)	1,500	0.6904 (0.2172)	0.7021 (0.2012)	0.7069 (0.2108)

Table 4 Computation time

Model name (number of vertices of an original model)	Number of vertices of simplified models	Elapsed time (s)		
		QEM	Roundness	The proposed method (Optimal positioning, curvature color + filtering)
Foot	400	0.230	0.341	1.340 (1.014, 0.062)
(5,200)	200	0.231	0.356	1.357 (1.028, 0.062)
IGEA	2,000	2.599	4.026	13.351 (10.169, 0.406)
(50,000)	1,500	2.682	4.095	13.743 (10.447, 0.406)
Nefertiti	2,000	4.541	4.670	11.011 (5.689, 0.539)
(90,000)	1,500	4.532	4.721	11.361 (6.006, 0.539)
Dragon	2,000	1.171	1.815	6.421 (4.865 ,0.219)
(25,000)	1,500	1.195	1.893	6.513 (4.931, 0.219)

around a geometric feature to preserve the feature within the user specified number of vertices. Consequently, the proposed method is expected to give the poor compactness over the entire triangles because of its feature sensitiveness. However, the proposed method gives the similar compactness compared to the other methods as described in Table 3 since the compactness (see Section 3.3.4) is included as a parameter of the optimal positioning function. However, the proposed method requires additional computation not only for the estimation of curvatures and optimal positions but also for the computation of extended QEM as shown in Table 4. As we observe from the table, the computation time mainly depends on the number of vertices of the original model and the desired number of vertices to be simplified except the Nefertiti model. In the case of the Nefertiti model, the computation time is relatively small compared to that of the IGEA model in spite of the larger number of vertices. Since the model has flat regions and the optimal positioning for these regions is not important, it ends up with relatively less computation time. All experiments were done on a PC with 2.60 GHz Intel Core 2 CPU and 2 GB RAM.

5 Conclusion

We have developed an improved QEM-based mesh simplification method that considers the curvatures and optimal positioning of collapsed vertices simultaneously. In this paper, the bilateral filtering method is proposed using attribute values and the edge length. It preserves the overall feature of the attribute value while noisy regions are smoothed out. It preserves the feature profile with a higher priority than the profile caused by the noise. The proposed method also presents an overall procedure for preserving fidelity of the geometry and the attributes. In order to achieve this, filtering of attribute values and optimal positioning of geometry are included in the procedure. In addition, enhanced preservation of geometric feature is achieved in the proposed method by transforming the curvature to colors and using them as attribute values. To demonstrate the performance of our method, visual comparison is made and the normal distance error between the original and simplified models is calculated. The proposed method shows better results in terms of the visual quality of the mesh compared to those obtained by QEM and the roundness method. The proposed method also yields better results in preserving the geometric features as shown by the normal distance error. However, this method requires additional computation time to calculate the curvatures and a PN triangle.

Other attributes such as texture coordinate, normal value, and material property can be applied to the proposed method. For example, the texture coordinates $\mathbf{t}[u, v]$ are easily used for bilateral filtering and also the QEM can be extended to five dimensions to accommodate vertex $\mathbf{v}[x, y, z, u, v]$.

Acknowledgements This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 20090083091) and the CTI development project supported by the MCST and KOCCA in S. Korea.

References

1. Schroeder WJ, Zarge JA, Lorensen WE (1992) Decimation of triangle mesh. In: Proceedings of SIGGRAPH 1992, pp 65–70
2. Ciampalini A, Cigonini P, Montani C, Scopigno R (1997) Multiresolution decimation based on global error. Vis Comput 13(5):228–246
3. Kobbelt L, Campagna S, Seidel HP (1998) A general framework for mesh decimation. In: Proceedings of graphics interface 1998, pp 43–50
4. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1993) Mesh optimization. In: Proceedings of SIGGRAPH 1993, pp 19–26
5. Lee SH, Lee KS (2002) Local mesh decimation for view-independent three-axis NC milling simulation. Int J Adv Manuf Technol 19:579–586
6. Garland M, Heckbert PS (1997) Surface simplification using quadric error metrics. In: Proceedings of SIGGRAPH 1997, pp 209–216
7. Chang M, Park SC (2009) Range data simplification for reverse engineering. Int J Adv Manuf Technol 41:86–96
8. Choi HK, Kim HS, Lee KH (2008) A mesh simplification method using noble optimal positioning. Lect Notes Comput Sci 4975:512–518
9. Garland M, Heckbert PS (1998) Simplifying surfaces with color and texture using quadric error metrics. In: Proceedings of IEEE visualization 1998, pp 263–269
10. Hoppe H (1999) New quadric error metric for simplifying meshes with appearance attributes. In: Proceedings of IEEE visualization 1999, pp 59–66
11. Kim HS, Choi HK, Lee KH (2008) Mesh simplification with vertex color. Lect Notes Comput Sci 4975:258–271
12. Garland M, Zhou Y (2005) Quadric-based simplification in any dimension. ACM Trans Graph 24(2):209–239
13. Cohen J, Olano M, Manocha D (1998) Appearance-preserving simplification. In: Proceedings of the 25th annual conference on computer graphics and interactive techniques, pp 115–122
14. Lindstrom P, Turk G (2000) Image-driven simplification. ACM Trans Graph 19(3):204–241
15. David CS, Alliez P, Desbrun M (2004) Variational shape approximation. In: Proceedings of SIGGRAPH 2004, pp 905–914
16. Wu J, Kobbelt L (2005) Structure recovery via hybrid variational surface approximation. Comput Graph Forum 24(3):277–284

17. Yan DM, Liu Y, Wang W (2006) Quadric surface extraction by variational shape approximation. Lect Notes Comput Sci 4077:73–86
18. Volpin O, Sheffer A, Bercovier M, Joskowicz L (1998) Mesh simplification with smooth surface reconstruction. Comput-Aided Des 30(11):875–882
19. Sheffer A (2001) Model simplification for meshing using face clustering. Comput-Aided Des 33(13):925–934
20. Lee AWF, Sweldens W, Schröder P, Cowsar L, Dobkin D (1998) MAPS: multiresolution adaptive parameterization of surfaces. In: Proceedings of SIGGRAPH 1998, pp 95–104
21. Sheffer A, Lévy B, Mogilnitsky M, Bogomyakov A (2005) ABF++: fast and robust angle based flattening. ACM Trans Graph 24(2):311–330
22. Ray N, Lévy B (2003) Hierarchical least squares conformal map. In: Proceedings of the 11th pacific conference on computer graphics and applications, pp 263–270
23. Lee AWF, Dobkin D, Sweldens W, Schröder P (1999) Multiresolution mesh morphing. In: Proceedings of SIG GRAPH 1999, pp 343–350
24. Kilian M, Mitra NJ, Pottmann H (2007) Geometric modeling in shape space. ACM Trans Graph 26(3):64–71
25. Guskov I, Sweldens W, Schröder P (1999) Multiresolution signal processing for meshes. In: Proceedings of SIGGRAPH 1999, pp 325–334
26. Meyer M, Desbrun M, Schröder P, Barr AH (2002) Discrete differential geometry operators for triangulated 2-manifolds. In: Proceedings of VisMath 2002, pp 35–57
27. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. In: Proceedings of ICCV 1998, pp 839–846
28. Fleishman S, Drori I, Cohen-Or D (2003) Bilateral mesh denoising. In: Proceedings of SIGGRAPH 2003, pp 950–953
29. Jones TR, Durand F, Desbrun M (2003) Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 22(3):943–949
30. Vlachos A, Peters J, Boyd C, Mitchell JL (2001) Curved PN triangles. In: Proceedings of the 2001 symposium on interactive 3D graphics, pp 159–166
31. ATI Technologies Inc. (2002) ATI OpenGL extension support. In: OpenGL extensions. AMD Inc http://ati.amd.com/developer/atiopengl.pdf. Accessed 16 Nov 2009
32. Rigiroli P, Campadelli P, Pedotti A, Borghese NA (2001) Mesh refinement with color attributes. Comput Graph 25(3):449–461
33. Guéziec A (1995) Surface simplification with variable tolerance. In: Proceedings of second int. symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95), pp 132–139
34. Frey JP, Borouchaki H (1998) Geometric evaluation of finite element surface meshes. Finite Elem Anal Des 31(1): 33–53
35. Botsch M, Steinberg S, Bischoff S, Kobbelt L (2002) Openmesh—a generic and efficient polygon mesh data structure. In: Proceedings of OpenSG symposium 2002
36. Edelsbrunner H (2001) Geometry and topology of mesh generation. Cambridge Univ Press, England
37. Kau CH, Richmond S, Zhurov AI, Knox J, Chestnutt I, Hartles F, Playle R (2005) Reliability of measuring facial morphology with a 3-dimensional laser scanning system. Am J Orthod Dentofac Orthop 128(4):424–430