An Effective Feature-Preserving Mesh Simplification Scheme Based on Face Constriction

Jian-Hua Wu[a], Shi-Min Hu[a], Chiew-Lan Tai[b] and Jia-Guang Sun[a]

[a]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[b]Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China
[a]wujh@ncc.cs.tsinghua.edu.cn, {shimin,sunjg}@tsinghua.edu.cn, [b]taicl@cs.ust.hk

Abstract

A new mesh simplification scheme that uses the face constriction process is presented. By introducing a statistical measure that can distinguish triangles having vertices of high local roughness from triangles in flat regions into our weight-ordering equation, along with other heuristics, our scheme can better preserve visually important features in the original mesh. To improve the shape quality of triangles, we adopt non-linear face area sensitivity in the weight ordering. Learning and feedback mechanism is also utilized to enhance user controllability. The computations are simple, making our scheme time-effective and easy to implement. In addition to comparing our scheme with other mesh simplification algorithms empirically, we compare their performances by establishing a unifying ground among three basic simplification processes – decimate vertex, collapse edge, and constrict face. This unification allows us to analyze the intrinsic merits and demerits of simplification algorithms to help users make better selections.

Keywords: mesh, mesh simplification, face constriction, feature-preserving

1. Introduction

Much of graphics workstation business requires high detailed models whose complexity is increasing far more rapidly than the performance of the graphics subsystems [28]. Today, by using precise laser range scanners, it is possible to obtain models so complex, like the David sculpture model [23], that no current graphics system can handle them. As models get ever larger, they get more difficult to store, transfer, render, and modify. One of the best solutions is to represent the complex details in multiple levels: *multiresolution* models [15] offer various versions of a model at different resolutions according to the user requirements.

In this paper, we propose an effective geometric mesh simplification scheme for constructing multiresolution meshes. The basic simplification process of our scheme is the *face constriction process* (FCP), whereby a triangle in the mesh is constricted to a new vertex (Figure **2**). The advantages of our scheme include good preservation of visually important features in the mesh, computation efficiency, and simplicity of implementation. By including a statistical measure of local roughness, non-linear sensitivity of triangle area in our weight-ordering computation, and other heuristics, our scheme can better retain distinctive features, such as boundaries, feature-lines, and high frequency details. To improve user controllability, we employ the learning and feedback mechanism to make use of parameter values that are returned from previous simplification runs for normalizing the different factors influencing the ordering weight.

In addition to performing visual comparison of our simplification results with the output of other algorithms, we also analyze the intrinsic characteristics of three basic simplification processes adopted by these algorithms, and obtain a unifying ground for comparing them in terms of speed and error introduced. Such comparison allows users to expediently select an appropriate class of simplification algorithms that suits the needs of their applications.

The remainder of this paper is organized as follows. Section 2 reviews related work and presents the three basic geometric simplification processes adopted by most simplification algorithms. In Section 3, we introduce some notations to formally describe the basic simplification process we employ, the *face constriction process*, and present a systematic procedure for checking the validity of a FCP. Section 4 presents our simplification scheme — the ordering weight computation and the placement of the new vertex. Section 5 shows some experimental results of our algorithm, contrasting them with results of other algorithms. In Section 6, we discuss the intrinsic relationships of the three basic simplification processes and compare the algorithms that adopt them. Finally, summary and future work are presented in Section 7.

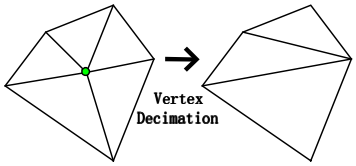
2. Related work

Not losing generality, the target objects of our research are triangular meshes, as they are most common and other polygonal models can be converted to them through local triangulations. Existing mesh simplification algorithms can be broadly classified into geometric-based, and appearance -based. Most of the latter can in fact be extended from the former by incorporating visual or appearance criteria [2,18,21, 3,19,10,6]. Hence, our research focuses only on geometric-based simplification algorithms. These algorithms utilize three different basic simplification processes:

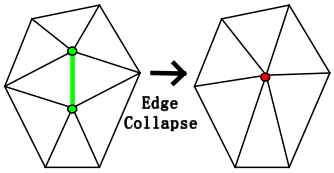
- *Vertex Decimation Process* (VDP) -- a vertex and its surrounding region are deleted, and the resulting hole is re-triangulated (Figure **1**(a)). Most earlier simplification methods are based on this process [30,31,4]. Schroeder et al. [30] use the distance from a vertex to the average plane

of its surrounding vertices to order the VDPs.

- *Edge Collapse Process* (ECP) -- an edge is collapsed into a new vertex, and its two adjacent triangles are deleted (Figure **1**(b)). This process is the most common one and has been extensively researched [20,18, 12,9,25]. Garland and Heckbert's elegant algorithm [9] introduces the quadric error metrics (QEM) to evaluate an ECP. Optimal new vertex placement is achievable under such metrics.
- *Face Constriction Process* (FCP) -- a face is constricted and its adjacent faces are immerged (e.g. Figure **2**). This process is a little more complex than the previous two, and there is little research on it. Hamann [14] and Gieng et al. [11] approximate the underlying surface in the neighborhood of a candidate triangle to obtain curvature estimation for computing the ordering weight.



(a)



(b)

Figure 1: *(a) Vertex Decimation Process: a vertex is deleted and its surrounding hole is re-triangulated; (b) Edge Collapse Process: an edge is collapsed into a new vertex.*

There are other algorithms that cannot be included in this taxonomy, such as the algorithms of Rossignac et al. [29] and Lindstrom [24]. A more general survey of mesh simplification algorithms can be found in [16].

3. FCP:Formal definition and validity checking

The simplification algorithms of Hamann [14] and Gieng et al. [11] both adopt the FCP as their basic simplification process. Gieng et al. [11] use the constriction process shown in Figure **2**, which results in a triangle being degenerated into a vertex, and three adjacent triangles being degenerated into three edges. Hence, it has the advantage of maintaining the topology of the original mesh. In contrast, Hamann's algorithm constricts faces through decimation. It deletes the candidate triangle and all the surrounding triangles, and then re-triangulates the remaining hole, without introducing any new vertex. This decimation procedure thus changes the topology of the mesh more drastically and, in general, attains a lower simplification quality (see the subset vertex placement example in Figure **16**). For this reason, we utilize the former constriction process. For triangular meshes, this FCP deletes four triangles a time, assuming manifold and non-boundary conditions.

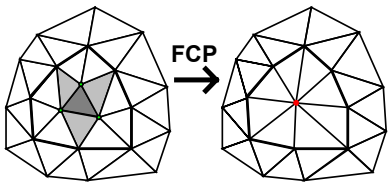


Figure 2: *The Face Constriction Process. The dark-shadowed triangle is constricted into a new vertex. Its three adjacent triangles become degenerated and are deleted.*

3.1 Definitions and notations

To facilitate the presentation of a systematic FCP validity checking procedure and subsequent discussion of our specific simplification scheme, we first introduce some definitions and notations. Let T denote a candidate triangle in a regular triangular mesh M.

Definition 1: The *surrounding triangles* of T are defined as the set $S_T=\{T_i \mid T_i \neq T,\ T_i \in M,\ T_i$ shares at least a vertex of $T\}$.

Analogously, the surrounding triangles of a vertex V are defined as the set S_V of all triangles containing V. Hereafter, the term surrounding triangles refer to the surrounding triangles of a triangle, unless otherwise specified to be those of a vertex.

Definition 2: The triangles in S_T that share an edge of T are called *immerging triangles*, denoted by the set I_T.

Definition 3: The surrounding triangles of T that are not immerging triangles are called *encircling triangles*, i.e., $E_T = S_T - I_T$.

Definition 4: The *encircling vertices* of T are defined as $EV_T = \{v \mid v$ is a vertex of some triangle in S_T and is not a vertex of $T\}$;

As an illustration, the dark-shadowed triangle in Figure **2** is the candidate triangle T of a FCP. Except for T, all the triangles inside the polygon with thicken boundary are in S_T. The three light-shadowed triangles are in I_T. All clear triangles inside the thicken polygon are in E_T, and the vertices on the thicken edges are in EV_T.

With these definitions, we can now describe precisely the effect of a FCP:

- the candidate triangle T is constricted into a new vertex NV, and T is deleted from M;
- the immerging triangles I_T are degenerated into edges, and the triangles are deleted from M;
- NV is connected to the encircling vertices EV_T.

Since the placement of NV is related to the mesh geometry and specific weight-ordering computation, we will discuss it with respect to our simplification scheme in Section 4.

3.2 Validity checking of a FCP

In earlier discussion, we assume that the mesh model is regular; however, many meshes in practice are non-manifold or include boundaries. To ensure that our simplification scheme also works well for such irregular input triangular meshes, we perform a systematic validity checking on can-

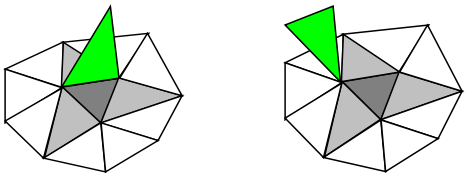
didate triangles of all FCPs.

Given a candidate triangle T of a potential FCP, we identify the FCP as illegal, by checking in the given order, if it satisfies any of the following conditions:

1. being non-manifold;
2. lie on the boundary;
3. contain a cycle or a hidden cycle;
4. cause triangle folding.

If any of these tests is positive, the checking procedure terminates and T is not constricted. The rationale of this test is that T in such regions would contain features that are important, both to the geometric and visual properties of the mesh, and thus should be preserved. Gieng et al. [11] discuss such checking loosely, and ignore the non-manifold condition; here we present a systematic procedure by classifying the surrounding triangles of T based on our terminology.

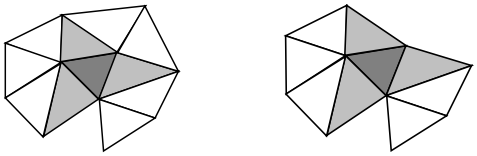
The non-manifold condition is as follows. If any triangle in I_T contains an edge that has more than two adjacent triangles (or more than one in the case of boundary edges) (Figure **3**(a)) or if some vertex in T has more than one ring of surrounding triangles (Figure **3**(b)), then this FCP satisfies the *non-manifold* condition.



(a) *(b)*

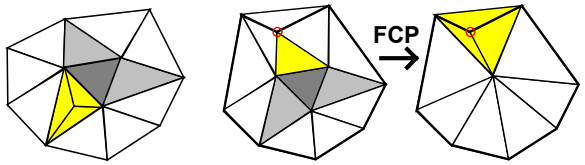
Figure 3: *Non-manifold FCP: (a) one edge has three adjacent triangles; (b) a vertex has two rings of surrounding triangles.*

We test the next two conditions as described in [11]. Using our terminology, if some edges of the triangles in I_T are on the boundary (see Figure **4**(a), (b)), then the FCP satisfies the boundary condition. To check the second condition, if a triangle in I_T is part of a cycle, which is a group of three mutually adjacent triangles (e.g. the yellow region in Figure **5**(a)), then the FCP is said to contain a cycle; applying the FCP would lead to superposition of the remaining two triangles in the cycle. In addition, if a vertex in I_T, but not in T, has valence four (Figure **5**(b)), then the FCP is said to contain a hidden cycle; performing such a FCP would introduce a new cycle, which is undesirable. Note that if T is part of a cycle, the FCP is considered legal.



(a) *(b)*

Figure 4: *Boundary FCP: (a) surrounding triangles are connected; (b) surrounding triangles are disconnected.*



(a) *(b)*

Figure 5: *FCPs containing a cycle or a hidden cycle: (a) surrounding triangles have a cycle; (b) surrounding triangles have a hidden cycle -- the triangle has a vertex of valence 4 which is reduced to valence 3 after the FCP, resulting in a cycle.*

Finally, we test if the FCP would lead to triangle folding (Figure **6**). Gieng et al. [11] detect this situation by checking whether or not the vertices in EV_T form a star polygon. Their method can produce good results, but it also incurs high computation cost. Hence, we utilize the method used by Ronfard et al. [27] and Garland et al. [9]. We compare the differences between the normal vectors of the encircling triangles before and after the FCP. If the differences are larger than a threshold, we consider it as resulting in triangle folding. Although this test is stricter than that of [11], it is much faster. And we have found it to work well in practice.

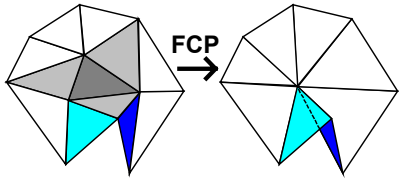


Figure 6: *Triangle folding of a FCP. The light blue and dark blue triangles are folded after the FCP.*

4. Mesh simplification scheme

The general framework of our scheme is as follows:

1) Compute the ordering weights of triangles in the mesh.

2) Insert all triangles into a heap ordered by the computed weights, with the minimum-weight triangle at the top.

3) Remove the top triangle from the heap, test if its FCP is illegal based on the discussions in Section 3.2. If so, assign the maximum cost to this triangle and insert it to the bottom of the heap; otherwise, constrict the triangle, compute the weights of its encircling triangles (i.e. triangles modified by the FCP), and update the heap.

4) Repeat step 3, until it meets the user-specified terminating condition, such as reaching the targeted number of triangles or the minimum weight in the heap.

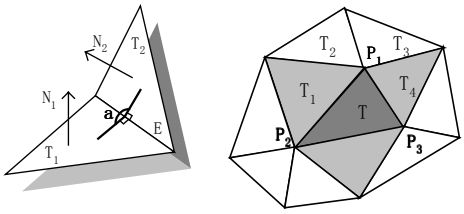
The remaining issues to resolve are how to compute the ordering weight of a candidate triangle and where to place the new vertex. The following two sub-sections address them.

4.1 Ordering weight computation

After applying a FCP, an error is introduced into the resulting simplified model. We must establish a cost measurement for the FCPs to obtain an ordering of the processes that minimizes the simplification error. Given a triangle T, we propose to compute the ordering weight of its FCP as follows:

$$W(T) = BF_n(A(T)) \cdot (wb \cdot G(T) + (1\text{-}wb) \cdot V(T)),$$

where $G(T)$ is the weighted sum of the average dihedral angles of the surrounding triangles of T's vertices, $V(T)$ is the valence of the new vertex, and wb is a parameter within [0,1]; $A(T)$ is the triangle area and the balancing function $BF_n(x) = x^n$ introduces a non-linear area sensitivity to the weight. The equation for computing $V(T)$ is simple: $V(T) = (V_1 + V_2 + V_3) - 9$, where V_1, V_2, V_3 are the valences of T's vertices. The exact equation for computing $G(T)$ will be given shortly. In essence, the $V(T)$ term favors constricting a triangle that produces a low-valence new vertex so that triangles of good shape quality are resulted, and the $G(T)$ term favors triangles in the flat regions that have no vertices contributing to any high-curvature features in the mesh; their relative importance is controlled by the parameter wb. This ordering equation essentially assigns a small weight to a triangle that has a small area and flat neighborhood with no high-curvature features, and that would result in a low-valence new vertex when constricted.



(a) (b)

Figure 7: *(a) Computing the dihedral angle. (b) The neighborhood of triangle T for computing G(T).*

The term $G(T)$ is defined in terms of dihedral angles between triangles. Given two adjacent triangles T_1 and T_2 that share an edge E, as shown in Figure **7**(a), their dihedral angle is the inclination angle a within [0, 180]. In practice, we use $G=1-N_1 \cdot N_2$, where N_1 and N_2 are the normal vectors of the two triangles, such that a large inclination angle (a flat region) corresponds to a low value of G.

To formulate $G(T)$, we illustrate with the candidate triangle T in Figure **7**(b). The vertices of T are P_1, P_2, P_3. For each vertex, we determine the dihedral angles between pairs of surrounding triangles of that vertex, and compute their mean and variance. For example, vertex P_1 has surrounding triangles T, T_1 to T_4; we compute five dihedral angles between T_1T, TT_4, T_4T_3, T_3T_2, and T_2T_1 to obtain the mean dihedral angle G_1 and the variance VAR_1 of these angles. Analogously, we can compute G_2, G_3, VAR_2, VAR_3. We can now define $G(T)$ as follows:

$$G(T) = \frac{G_1 \cdot VAR_1 + G_2 \cdot VAR_2 + G_3 \cdot VAR_3}{VAR_1 + VAR_2 + VAR_3}.$$

To illustrate the significance of the variance terms, we show that the mean terms alone are not adequate. Although low values of G_1, G_2, and G_3 often reflect flatness in T's neighborhood, they alone cannot exclude triangles that have some vertices contributing to high frequency details. Figure **8** illustrates, using a 2D example, a problem that arises when only the mean values are considered. The two meshes have the same mean value and would be concluded as having the same weight by most simplification schemes. But, obviously, the bottom mesh has a high frequency detail and should be constricted later. Since the variance terms can reflect the roughness of a vertex neighborhood, the mean values are weighted by the variances in our formulation. Large variances lead to a large $G(T)$, thus disfavoring constricting T and retaining the rough feature in the model. Most existing simplification algorithms [30,4,9,11] do not deal with such a factor, even though these high frequency details are surely visually important and should be preserved as much as possible.

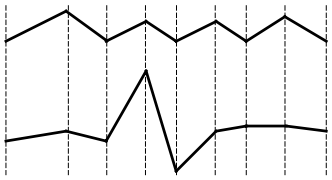


Figure 8:*Two meshes, shown in 2D, have the same mean dihedral angle, but the variance of the bottom mesh is bigger, reflecting the local vertex roughness.*

Lastly, we consider the issue of normalizing $G(T)$ and $V(T)$. Since the ranges of $G(T)$ and $V(T)$ are different ($G(T)$ ranges between 0 and 2, while $V(T)$ is an arbitrary integer), when they form a convex sum in $W(T)$, it is difficult for the user to select the value of wb that corresponds to a desired relative importance. To overcome this problem, we should normalize them to [0, 1] to improve the controllability of our scheme; that is,

$$G(T) = (G(T) - G_{min}) / (G_{max} - G_{min}),$$
$$V(T) = (V(T) - V_{min}) / (V_{max} - V_{min}),$$

where G_{min}, G_{max}, V_{min}, V_{max} are the extrema of $G(T)$ and $V(T)$ in the mesh. However, for complex meshes, it is not easy to obtain these extrema, and most existing algorithms either ignore them or determine them empirically. Here, we introduce the *learning and feedback mechanism* to refine these extrema. The first time we simplify a model, we initialize the maxima and the minima to arbitrary small and large values, respectively, and update them during the simplification. The final-adjusted extrema are used as inputting parameters the next time we simplify the same model.

4.2 New vertex placement

After constricting a candidate triangle T, its three vertices are merged into a new vertex NV. Several methods exist for defining the position of NV, for example, letting it be the position of one of T's vertices (subset), midpoint of one of T's edges, or the center of T (average). These simple methods may be easy to compute, but produce low quality results (see Figure **16**). Inspired by the '*piercing*' step in normal meshes [13], we propose the following placement method. The idea is to select a point in T as the origin O, then shoots a ray from O in the direction of the normal vector of T to intersect with the original mesh, and let the intersection be the position of NV. This approach has the advantage that the new vertices always lie on the original mesh.

The selection of the origin O should be dependent on the neighborhood of the triangle. As mentioned earlier, the three

variances of the dihedral angles, VAR_1, VAR_2, VAR_3, reflect the roughness around the three vertices P_1, P_2, P_3 of T. Therefore, we let the origin be at the position P_O given by:

$$P_0 = \frac{P_1 \cdot VAR_1 + P_2 \cdot VAR_2 + P_3 \cdot VAR_3}{VAR_1 + VAR_2 + VAR_3}$$

Computing the ray's intersections with all the triangles in the original mesh would be extremely time-consuming. Hence, for each triangle, we maintain a list of indices of related triangles in the original mesh. Initially, the list of each triangle contains only index to that triangle. After performing an FCP involving a triangle T, we append T and I_T's lists to E_T's list. In this way, for each triangle in a simplified mesh, we can record those original triangles that are related to it, thus keeping the history information. For each constricted triangle T, we only need to check the intersections of a ray with triangles in the history list of T and S_T. If no intersection exists, we use O as the new vertex.

5. Experimental results and comparisons

To evaluate the quality of the output of our simplification scheme, we use the error measurement introduced by Garland and Heckbert [9]; that is, given an original mesh model M and a simplified one M_s, the error between them is computed as:

$$error = \frac{1}{n + n_s}\left(\sum_{v \in MV} d^2(v, M_s) + \sum_{v \in MV_s} d^2(v, M)\right)$$

where MV and MV_s are the vertex sets of M and M_s respectively, n and n_s are the sizes of the respective sets, and $d(v, M)$ represents the minimal distance from v to the closest triangle in M. For large models, this error computation can be very time-consuming. However, since our simplification scheme records the history information, we can use this information to speed up the error computation as in [7].

No. Faces	Simplified %	Running Time (s)	Error
2900	50%	3.745	1.8297e-6
1160	80%	5.218	2.0293e-5
580	90%	5.907	5.0502e-5
290	95%	6.054	2.0505e-4

Table 1: *Running time and error of simplifying the cow model shown in Figure* ***18*** *using our scheme.*

We have applied our scheme to some mesh models that are classic in this field. Table **1** shows the running time and error of simplifying the cow model shown in Figure **18** into various levels of detail using our scheme. Table **2** summarizes some typical performance of our scheme in simplifying other mesh models. All the experiments were carried out on a PC with a PIII 600MHz processor and 128MB RAM. The n in the function $BF_n(x)$ is set to 0.5 and wb is 0.7. The learning and feedback is also adopted.

We now discuss some visual characteristics of our simplification outputs and compare them with outputs of some existing algorithms. In the next two subsections, we first discuss feature-preserving characteristics, followed by other geometric properties.

Models	Org. No of Faces	Sim. No. of Faces	Sim. %	Running Time (s)
Hypersheet	3,832	632	84%	3.12
Cow	5,804	580	90%	5.91
Fandisk	12,946	1,632	87%	16.82
Bunny	69,451	9,999	86%	59.37
Venus	100,000	5,000	95%	99.40

Table 2: *Typical run time performance of simplifying various mesh models.*

5.1 Feature-preserving characteristics

Since our scheme is completely geometric-based, it is essential to employ some heuristics to preserve features during simplification. We consider the *features* of a model as those that are both geometrically and visually important to human perception. Specifically, our scheme can preserve the following features.

5.1.1 Preserving boundaries

We consider those FCPs that satisfy the boundary and non-manifold conditions to be illegal; hence our scheme naturally preserves boundary features. Figure **9** shows an example of simplifying a hypersheet model. We can see that the boundaries are preserved precisely, albeit at the expense of requiring more faces. Some recent works [1,3,9,25] can simplify boundaries while still maintaining reasonably good boundary quality. We hope to improve our boundary preserving heuristics in our ongoing work.

5.1.2 Preserving feature-lines

Feature-lines [22] are sharp edges whose two adjacent faces have a dihedral angle of less than some threshold [17, 18]. These lines reflect the overall geometry appearance of a model and are visually important. Because we use dihedral angles as the main part of the weight-ordering computation, our scheme can automatically preserve these feature-lines without any additional aids. Figure **19** shows an example where most feature-lines are exactly preserved.

Some other algorithms [17,18,22] can produce similar results, but they require the feature lines to be tagged and handled by special codes, thus increasing the memory requirement, running time, and implementation complexity.

5.1.3 Preserving high frequency details

The measurement of vertex local roughness in the ordering weight equation, along with non-linear face area sensitivity and triangle folding testing, enable better preservation of high frequency details in the mesh. In Figure **10**, we compare our result with that of Garland and Heckbert's QEM [9]. Notice that the eye and nose regions in our simplified mesh are better preserved.

Another example of preserving such details is shown in Figure **20**, which compares the results produced with and without our proposed heuristics. The examples show that preserving high frequency details is necessary in simplification and can be achieved by our scheme. To the best of our knowledge, no other current algorithms have such capability.

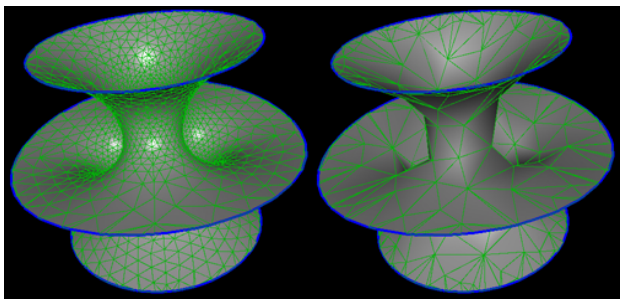


Figure 9: *Preserving boundaries. The original hypersheet mesh of 3832 faces (left), and a simplified mesh of 632 faces (right).*

Figure 10: *Comparison with QEM. The original model of 5,804 faces (left), and simplified models of 2,000 faces using our scheme (middle) and QEM (right).*

5.2 Comparisons and additional features

We now compare our output with that of Gieng et al [11] since their method uses the same FCP as its basic simplification process.

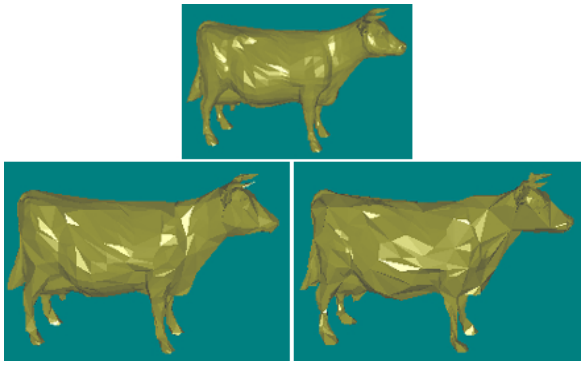


Figure 11: *Comparison with Gieng et al's scheme. The original model of 5,804 faces (top), simplified models of 1,500 faces using our method (bottom-left) and Gieng et al's (bottom-right).*

As their paper does not include results on running time or error, we implemented their algorithm to facilitate comparison. Figure **11** shows two simplified models of 1,500 faces, produced by our scheme and by Gieng el al's. It can be observed that the new ideas introduced in our scheme, in particular the flatness and the vertex local roughness considerations, non-linear area sensitivity, and the piercing vertex placement, are able to produce better quality output that preserves the distinctive features in the mesh, e.g. the hind leg and foreleg regions. Preservation of these important visual features also causes the silhouette of our simplified models to remain relatively little changed, e.g. at the neck region. Our scheme also performs better in terms of running time and error. This is due to the use of simple dihedral angle computation, which only needs a dot product, compared with the complex curvature estimation and least square solutions required by the algorithm in [11]. The running time and error of our scheme and those of Gieng et al's for producing the two models in Figure **11** are respectively 4.873s and 17.405s, 1.0354e-005 and 2.3531e-005.

In addition to the above general comments, we now summarize some additional properties of our scheme:

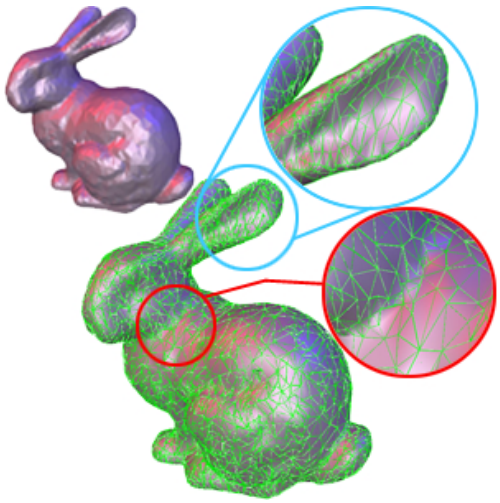


Figure 12: *Example of automatic adaptive simplification. Two identical bunny models simplified to 9,999 faces using our algorithm. The top model is flat shaded, and the bottom is smooth shaded.*

- Since our scheme uses the dihedral angles, which can reflect flatness, as the main component for ordering the weight, it automatically achieves adaptive simplification. This effect can be observed in Figure **12**; the high curvature regions at the neck of the bunny and the brim of its ear has denser and smaller triangles, while the flatter regions at its back have relatively large triangles.

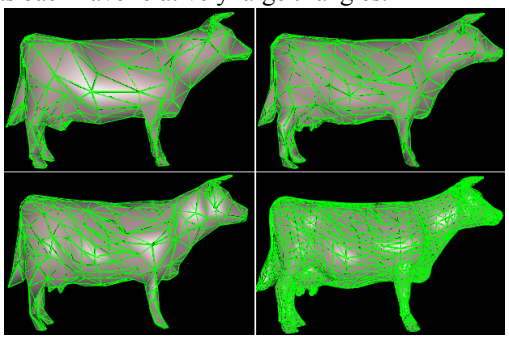


Figure 13: *Visual effects of different values of n. The original mesh (bottom-right) and simplified meshes of 800 faces obtained with n equals 0.5 (top-left), 1.0 (top-right), and 5.0 (bottom-left).*

- The non-linear area sensitivity conforms to the characteristic of human vision, and thus further ensures good shape

quality triangles. Figure **14** plots the errors of simplifying the cow model with different values of *n* and *wb*. For *n=0.5*, although the error increases rapidly for very low and very large values of *wb*, it is the smallest for most values of *wb*∈ [0,1]. The linear sensitivity of *n=1.0*, which is used by most algorithms, produces larger errors and fluctuates with different *wb* values. Similar problems occur with *n=5.0*. Figure **13** shows the simplified meshes produced using different values of *n* and the same *wb*=0.7. Observe that for *n=0.5*, the simplified mesh has triangles of good shapes, compared with *n=1.0* and *5.0* where many triangles are long and narrow. Hence, in our implementation, *n* is set to *0.5*.

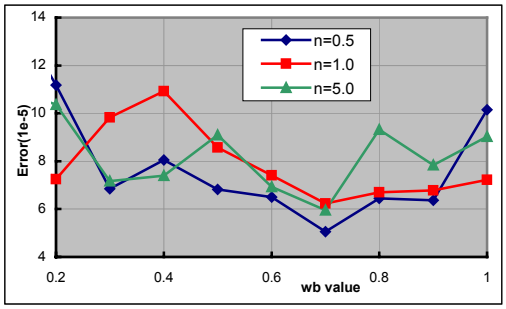


Figure 14: *Effects of different values of n in $BF_n(x)$. The graph plots the errors of simplifying the cow model to 580 faces using n=0.5, 1.0, 5.0 and wb varied from 0.2 to 1.0.*

- The learning and feedback mechanism adopted for normalizing the two factors *G(T)* and *V(T)* improves the controllability of our scheme. Figure **15** shows the error curves of simplifying the cow model with and without using the mechanism and with various *wb* values. It is observed that, although they have similar minimum error, the one that adopts the mechanism is clearly flatter, facilitating the user to select a proper *wb* in applications.

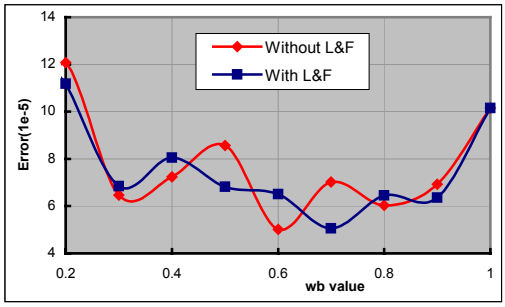


Figure 15: *Effects of the learning and feedback (L&F) mechanism. The curve plots the errors of simplifying the cow model to 580 faces with and without L&F mechanism, and wb varied from 0.2 to 1.0.*

- Our vertex placement method of finding the intersection of a ray with the original mesh further improves the quality of the simplification output. Compared with the subset (chooses the position of one of T's vertices) and average vertex placement methods, the piercing placement method produces models of better visual appearance and low error. Figure **16** shows the results of our simplification schemes using the three vertex placement methods. Comparing the highlight areas in the original and simplified models, our vertex placement appears to be the most effective.

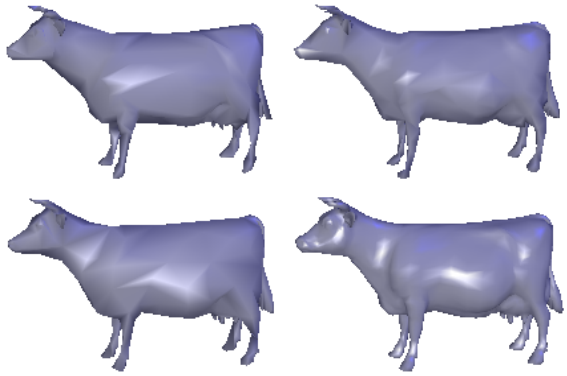


Figure 16: *Effects of vertex placement methods. The original cow model of 5804 faces (bottom-right), and simplified models of 1,000 faces obtained using our simplification scheme and employing the subset (top-left), average (top-right), and piercing (bottom-left) vertex placements, with errors 3.2302e-005 (regarded as 100%), 2.8326e-005 (87.7%) and 2.4127e-005 (74.7%) respectively.*

6. Analytical comparisons of mesh simplification algorithms

After evaluating our simplification scheme empirically in the last section, we now compare it with other algorithms analytically. As mentioned earlier, most mesh simplification algorithms can be classified into three classes according to the basic simplification process they adopt. They are VDP-based, ECP-based and FCP-based simplification algorithms. Since their target elements are vertices, edges and faces, which are inter-connected in a mesh, we attempt to find a common ground for comparison.

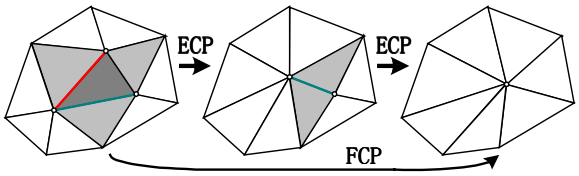


Figure 17: *Relationship between FCP and ECP. The leftmost mesh is simplified to the rightmost mesh by collapsing two edges successively. The result can also be achieved directly by one FCP.*

We first compare VDP and ECP. Although a VDP deletes a vertex and re-triangulates the resulting hole, it can be viewed as collapsing an edge that is connected to the vertex into its other endpoint in a given triangulation. Hence, we can consider a *VDP* as an *endpoint ECP*. Analogously, we can find the intrinsic relationship between FCP and ECP. As shown in Figure **17**, if two successive ECPs involve two edges from the same triangle, the result is the same as a FCP. That is, we can consider a *FCP* as two *successive ECPs*.

Based on their intrinsic relationships, we conclude that all the three processes can be unified using ECP. We therefore consider ECP to be the fundamental process. A significant

practical benefit of this unification is that different error measurements and implementations of a simplification scheme that adopts one basic process can be employed by another scheme that adopts another basic process. For example, although the basic process of QEM [9] and its experimental implementation QSlim [8] is ECP, we can easily apply their quadric error metrics to a VDP-based algorithm by assigning the Q matrix to every vertex. Similarly, we can introduce the Q matrices to all the vertices of a triangle to evaluate the error of a FCP. As another example, Garland [7] used the QEM in an FCP-based simplification scheme, and QSlim [8] offers two almost identical implementations of ECP and FCP using QEM.

Although the three basic processes have a unifying ground, they are distinctive in their advantages and suitability for different applications. We now analyze the three classes of simplification algorithms in terms of speed and error based on the established unifying ground. The comparison can aid users in making a better selection of simplification algorithms. We note that Garland [7] has compared empirically the running time and error of his QEM-based ECP and FCP algorithms, but did not offer detailed explanations for their differences in performance.

- Speed

It is often complex to analyze the speed of different mesh simplification algorithms because they may adopt different basic processes. Even if they use the same process, their process-ordering methods and implementations (such as data structures, algorithm frameworks, query methods, sorting methods) may differ, thus making analytical comparisons non-trivial. Almost all researchers present only the absolute speed of their algorithms, such as the running time, as we have done in Section 5. Lindstrom et al. [26] evaluated the absolute speed of many simplification algorithms. However, such absolute speed evaluation only partially reflects the advantages of an algorithm and not its intrinsic properties. Due to the unifying nature of the basic processes, which implies that the same implementations and ordering methods can be used by different schemes adopting different basic processes, we can omit these factors and only evaluate the relative speeds of the basic processes per se.

The VDP and the ECP delete two triangles from the mesh, while the FCP deletes four. Hence, the unification allows us to conclude that FCP is the fastest. Next, comparing the VDP and ECP processes, we conclude that the former is faster because it does not have to compute new vertex placement while the latter does. Thus we conclude that $SPEED_{FCP} > SPEED_{VDP} > SPEED_{ECP}$. This conclusion is consistent with the results in Section 6.1 of [7].

- Error

To compare the errors of the different classes of algorithms, we presume that they adopt the same ordering formulae and implementations. As the VDP does not introduce new vertices, its error is larger than ECP and FCP. On the other hand, a FCP is equivalent to two successive ECP applied to two edges from the same triangle. Since the chances that two successive ECPs being applied to two edges of the same triangle are low in an ECP-based algorithm, and the FCP is likely to delete some edges that have larger weights than the collapsed edges of the two ECPs, we conclude that the error of the FCP is larger than that of ECP. Hence, our conclusion is $ERROR_{VDP} > ERROR_{FCP} > ERROR_{ECP}$.

With the above discussion, we summarize our comparisons of the three classes of simplification algorithms in Table **3**.

Simplification Algorithms	**Speed**	**Error**
The VDP-based simplification	Fast	Largest
The ECP-based simplification	Slowest	Smallest
The FCP-based simplification	Fastest	Small

Table 3: *Comparisons of three classes of simplification algorithms.*

Our comparisons enable users to better select a simplification scheme to suit the needs of their applications. For example, the ECP-based simplification has the smallest error and produces the best quality results, so it can be used in applications where the accuracy of the mesh is of paramount importance. The FCP-based simplification is the fastest, hence it can be applied in speed critical conditions. With the unifying ground, we can distinguish the intrinsic properties of individual basic processes from the absolute speed results given in [26]. By removing these intrinsic factors, the results in [26] can also reflect the advantages of different ordering methods and implementations. Hence, after selecting the basic process based on our comparisons, the user can choose a reasonable ordering method and implementation according to the evaluation of [26] to optimize their final application.

7. Summary and Future Work

In this paper, we have introduced an effective feature-preserving mesh simplification scheme that adopts the basic face constriction process proposed by Gieng et al. [11]. In formulating the weight-ordering equation, we borrow a statistical idea to introduce terms involving the mean and variance of dihedral angles, which can detect flat neighborhoods that are free from rough features. Along with other heuristics, like FCP validity checking, the weight ordering equation enables our scheme to be effective in preserving features of high geometric and visual importance, such as boundaries, feature-lines and high frequency details. The computations are simple, making our scheme time-effective and easy to implement. Together with favoring constricting triangles that give rise to low-valence new vertices, the weight ordering incorporates non-linear area sensitivity to further ensure good triangle shapes in the output. We also introduce the learning and feedback mechanism to normalize the ordering factors, which enhances user controllability. By making use of the history information when computing the new vertex placement, our simplified models is able to retain as much original details as possible.

For analytic comparison of various algorithms that adopt different basic processes, we have unified three basic simplification processes, the Vertex Decimation Process (VDP), the Edge Collapse Process (ECP) and the Face Constriction Process (FCP), by showing their intrinsic relationships. Based on this unification, we discuss the differences among the three basic processes and describe their merits and demerits.

There are several directions for future work:

- Since a FCP is fundamentally ECP, we can extend our scheme to construct progressive mesh representations [18] for mesh models. This will allow our scheme to perform multiresolution modeling.
- Permit more types of triangles, such as those having their surrounding triangles on the boundary, to be constricted to achieve higher simplification rate.
- Include some appearance criteria involving visual attributes like color, texture and viewing parameters into our scheme to perform appearance preserving and view-dependent simplification.
- Apply our scheme to simplify large-scale and complex environments such as those in virtual reality applications to speed up rendering while ensuring high fidelity.

Acknowledgements

This research was supported by the Natural Science Foundation of China (Project Number 69902004) and the Hong Kong Research Grant Council (AOE 98/99.EG01). The mesh models in our paper are courtesy of the Stanford University Computer Graphics Laboratory, the Computer Graphics Group in University of Washington, Michael Garland and Hugues Hoppe. We are particularly grateful to Dr Hujun Bao for his valuable comments and suggestions.

References

1. C. Bajaj, D. Schikore, Error-bounded Reduction of Triangle Meshes with Multivariate Data, *SPI*E, 2656, 34–45, 1996
2. A. Certain, J. Popovic, T. DeRose, etc, Interactive Multiresolution Surface Viewing, *Computer Graphics (SIGGRAPH'96 Proceedings*), 91–98, 1996
3. J. Cohen, D. Manocha, M. Olano, Simplifying Polygonal Models Using Successive Mappings, *Proceedings of IEEE Visualization'9*7, 395–402, 1997
4. J. Cohen, A. Varshney, D. Manocha, etc, Simplification Envelopes, *Computer Graphics (SIGGRAPH'96 Proceedings*), 119-128, 1996
5. M. Eck, T. DeRose, T. Duchamp, etc, Multiresolution Analysis of Arbitrary Meshes, *Computer Graphics (SIGGRAPH'95 Proceedings*), 173–182, 1995
6. J. El-Sana, Y. J. Chiang, External Memory View-Dependent Simplification, *Computer Graphics Forum*, 19(3), 2000, EG2000 Proceedings
7. M. Garland, Quadric-Based Polygonal Surface Simplification, PhD thesis, School of CS, CMU, 1999
8. M. Garland, QSlim v2.0, http://www.uiuc.edu/~garland/CMU/quadrics/qslim.html, 1999
9. M. Garland, P. Heckbert, Surface Simplification Using Quadric Error Metrics, *Computer Graphics (SIGGRAPH'97 Proceedings*), 209-216, 1997
10. M. Garland, P. Heckbert, Simplifying Surfaces with Color and Texture using Quadric Error Metrics, *Proceedings of IEEE Visualization'98*, 1998,
11. T. S. Gieng, B. Hamann, K. I. Joy, etc, Smooth Hierarchical Surface Triangulations, *Proceedings of IEEE Visualization'97*, 379-386, 1997
12. A. Gueziec, Surface Simplification Inside a Tolerance Volume, *Second Annual International Symposium on Medical Robotics and Computer Aided Surgery*, 132-139, 1995
13 I. Guskov, K. Vidimce, W. Sweldens, etc, Normal meshes, in *Computer Graphics (SIGGRAPH'2000 Proceedings*), 95-102, 2000
14. B. Hamann, A Data Reduction Scheme for Triangulated Surfaces, *Computer Aided Geometric Design*, 11, 197-214, 1994
15. P. Heckbert, Multiresolution Surface Modeling, Siggraph97 course notes 25, 1997
16. P. Heckbert, M. Garland, Survey of Polygonal Surface Simplification Algorithms, T.R of Dept. CS, Carnegie Mellon Univ., May, 1997
17. H. Hoppe, T. DeRose, T. Duchamp, etc, Piecewise Smooth Surface Reconstruction, *Proceedings of SIGGRAPH'9*4, Computer Graphics Proceedings, Annual Conference Series, 295-302, 1994
18. H. Hoppe, Progressive Meshes， *Computer Graphics (SIGGRAPH'96 Proceedings*), 99-108, 1996
19. H. Hoppe, View-dependent Refinement of Progressive Meshes, *Computer Graphics (SIGGRAPH'97 Proceedings*), 189–198, 1997
20. H. Hoppe, T. DeRose, T. Duchamp, etc, Mesh Optimization， *Computer Graphics (SIGGRAPH'93 Proceedings*), 19-26, 1993
21. M. Hughes, A. A. Lastra, E. Saxe, Simplification of Global-illumination Meshes, *Computer Graphics Foru*m, 15(3), 339–345, 1996, Proceedings of Eurographics'96
22. A. Lee, W. Sweldens, P. Schreoder, etc, MAPS: Multiresolution Adaptive Parameterization of Surfaces, *Proceedings of SIGGRAPH'9*8, Computer Graphics Proceedings, Annual Conference Series, 95-104, 1998
23. M. Levoy, K. Pulli, B. Curless, etc, The Digital Michelangelo Project: 3D Scanning of Large Statues, *Computer Graphics (SIGGRAPH' 2000 Proceedings*), 131-144, 2000
24. P. Lindstrom, Out-of-Core Simplification of Large Polygonal Models, *Computer Graphics (SIGGRAPH' 2000 Proceedings*), 2000
25. P. Lindstrom, G. Turk, Fast and Memory Efficient Polygonal Simplification, *Proceedings of IEEE Visualization'98*, 279-286, 1998
26. P. Lindstrom, G. Turk, Evaluation of Memoryless Simplification, *IEEE Transactions on Visualization and Computer Graphics*, 5(2), 98-115, 1999
27. R. Ronfard, J. Rossignac, Full-range Approximation of Triangulated Polyhedra, *Computer Graphics Foru*m, 15(3), Aug. 1996, Proc. of Eurographics '96
28. J. Rossignac, Geometric Simplification and Compression, in Siggraph97 course notes, 1997
29. J. Rossignac, P. Borrel, Multiresolution 3D Approxi-mations for Rendering Complex Scenes， *Modeling in Computer Graphics*, edited by B.Falcidieno and T.L.Kunii, Springer-Verlag, 455-465, 1993
30. W. J. Schroeder, J. A. Zarge, W. E. Lorensen, Decimation of Triangle Meshes, *Computer Graphics (SIGGRAPH'92 Proceedings)*, 26(2), 65-70, 1992
31. M. Soucy, D. Laurendeau, Multiresolution Surface Modeling Based on Hierarchical Triangulation, *Computer Vision and Image Understandin*g, 63(1), 1–14,1996

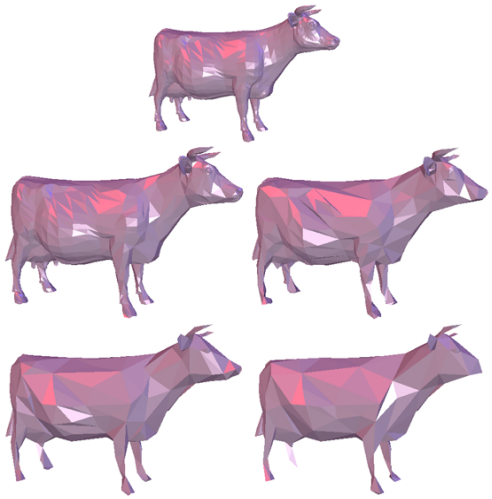


Figure 18: *Different levels of detail of a cow model. The topmost original model has 5804 faces. From left to right, top to bottom, the other models have 2900, 1160, 580, 290 faces respectively.*

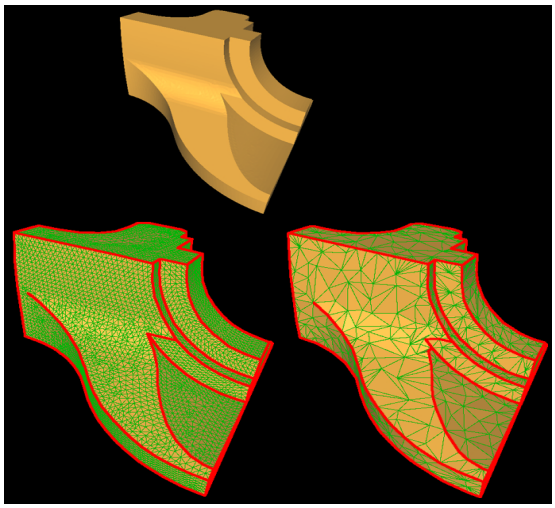


Figure 19: *Preserving feature-lines. The original fandisk model (top and bottom-left) of 12,946 faces is simplified to 1,632 faces (bottom-right). The feature-lines are red.*

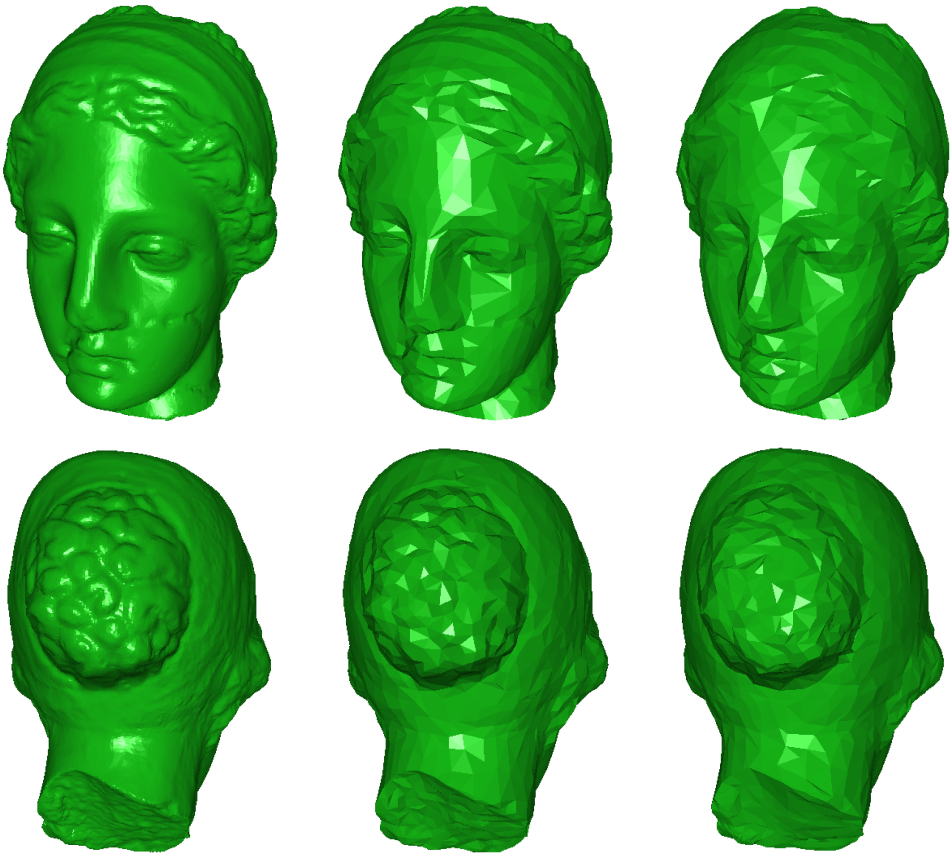


Figure 20: *Preserving high frequency details. The left column shows the front and back views of the original venus model of 100,000 faces; the middle and the right columns show simplified models of 5,000 faces, respectively obtained with and without employing the heuristics for preserving details. That is, the middle model is obtained using our scheme, and the right model is obtained with G(T) and the origin P_o computed using only the mean values, and n in $BF_n(x)$ set to be 1.0. Although these two conditions give relatively similar results, we can see that our heuristics can better preserve high frequency details, for example, the hair regions just above the forehead, the hair waves in the front view, and the ringlet mass in the back view.*