Mesh Simplification With Appearance-Driven Optimizations

YONGZHANG XIAN, YUXUE FAN, YAN HUANG, GUOPING WANG, XIANGXU MENG, (Member, IEEE), AND JINGLIANG PENG, (Member, IEEE)

1School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
2Patent Examination Cooperation Center, State Intellectual Property Office, Beijing 100081, China
3School of Software, Shandong University, Jinan 250101, China
4School of Computer Science and Technology, Shandong University, Qingdao 266100, China

Corresponding author: Yan Huang (huang_yan74@163.com)

This work was supported in part by the National Key Technology Research and Development Program of China under Grant 2017YFB1002700, and in part by the National Natural Science Foundation of China under Grant 61632003 and Grant 61872398.

ABSTRACT In this work, we propose a novel solution for simplifying texture-mapped three-dimensional (3D) meshes. It simplifies a 3D triangular mesh to optimally preserve the visual appearance of the original texture-mapped model at reduced vertex budgets. While taking the prevalent strategy of iterative edge contraction, the proposed scheme is novel in that it takes into account the local texture image characteristics when prioritizing local mesh simplification operators, and conducts closed-form optimization when computing the texture coordinates of each replacement vertex. Outstanding performance of the proposed scheme is demonstrated both qualitatively and quantitatively by experimental results. Further, validity of the proposed algorithmic components is well proved through ablation study.

INDEX TERMS Mesh simplification, level of detail (LOD), appearance-driven.

I. INTRODUCTION

While high realism is increasingly demanded in many graphics applications including film production, video gaming, digital city, scientific modeling and visualization and so forth, there still exists a gap between high volume of data and limited resources of storage, bandwidth and/or computation in many contexts. For instance, graphics applications on mobile terminals may only afford to use lightweight three-dimensional (3D) models for efficient storage, transmission, rendering and editing. In order to reduce this gap, we focus on the simplification of 3D triangular meshes with texture attributes.

Three-dimensional model simplification techniques adapt a 3D model created in high resolution to different levels of detail (LODs) for different application scenarios. This field has been intensively investigated in the past, with many good algorithms published. Early 3D model simplification algorithms strove to maintain the geometric fidelity at a reduced vertex count. Later, simplification algorithms were proposed to minimize both the geometric error and the texture deviation in order to preserve the appearance of the texture-mapped model. In general, the texture deviation was controlled less well than the geometric error by those algorithms.

In this work, we propose a novel and complete solution for simplifying texture-mapped 3D meshes. In order to maximally preserve the appearance of the original texture-mapped model, the proposed scheme considers both mesh and texture features to prioritize the local simplification operators and minimizes texture deviations over the simplified surface. Key features of the proposed scheme include:

- **Appearance-driven mesh simplification.** Texture dependent factors are introduced to the simplification cost metric to modify the global vertex distribution such that less texture deviation will be produced in local regions with more texture details. Further, closed-form optimization is proposed for each replacement vertex to derive its optimal texture coordinates which minimize the local texture deviation on the surface.

- **Generality of framework.** Though the proposed scheme uses iterative edge contraction for the mesh simplification, it is generally extensible for other simplification operators (e.g., vertex decimation, vertex clustering) with minor adaptations.
II. RELATED WORK
The field of 3D model simplification has been researched for more than two decades with many algorithms published. Due to the space limit, we briefly review some of those algorithms. For comprehensive surveys of algorithms published in early years, we refer to several good works [1]–[4].

Most early 3D model simplification algorithms focus on preserving geometric fidelity of the simplified model, as reviewed in Section II-A. Attributes other than geometry, especially texture attributes, have also been taken into account by 3D model simplification algorithms for better appearance preservation, as reviewed in Section II-B.

A. GEOMETRY-CENTRIC SIMPLIFICATION
Although some algorithms simplify 3D models through global optimal point sampling and redistribution [5] or surface partitioning and approximation [6], most existing algorithms work by repeatedly applying simplification operators until a distortion threshold or a desired decimation rate is reached. The most commonly used simplification operators include vertex decimation, vertex pair contraction and vertex clustering. Vertex-decimation-based algorithms [7], [8] remove a vertex and re-tessellate the resulting hole at each simplification step. Vertex-pair-contraction-based algorithms [9]–[14] contract an edge or even a pair of unconnected vertices to one vertex and update the local topology correspondingly at each simplification step. Vertex-clustering-based algorithms [15]–[17] merge a cluster of vertices to one and update the local topology correspondingly at each simplification step.

B. APPEARANCE-CENTRIC SIMPLIFICATION
The majority of these algorithms use iterative edge contraction (or named edge collapse) to simplify a mesh and determine the texture coordinates of each replacement vertex with care. Garland and Heckbert [18] extend their original algorithm [12] by incorporating attributes into the quadric error metric (QEM). Hoppe [19] proposes an improved quadric error metric leading to improved storage use, computational efficiency and resultant model quality. Cohen et al. [20] build successive mappings between LODs and convert the 3D positioning to two dimensional (2D) positioning plus one-dimensional (1D) offsetting of each replacement vertex. Cohen et al. [21] further improve their work by introducing a texture deviation metric and locally finding the new vertex’ texture coordinates based on this metric. Williams et al. [22] prioritize the edge collapses based on a perceptual model taking into account texture deviation, contrast of illumination, dynamic lighting and so forth.

There are other algorithms that refer to the texture images or the rendered images during the mesh simplification. Lindstrom and Turk [23] perform image-driven simplification that utilizes the rendered model images to compute the edge collapse cost. When computing the edge or half edge collapse cost, Shao et al. [24] consider the local texture error and Gonzalez et al. [25] penalize collapsing edges that cross borders in the texture image or have large neighborhood surface areas. Gonzalez et al. [26], [27] define the cost of collapsing an edge as the resultant change in structural appearance of the textured model. The methods of Lindstrom and Turk [23] and Gonzalez et al. [26], [27] require surface rendering at multiple viewpoints for the simplification cost computation, while it may be hard to make a fair, concise and sufficient view sampling that is universally applicable to models of arbitrary geometrical complexity. Moreover, the iteratively conducted 3D model rendering may cause intensive computation. It is also worth noting that, in the methods of Shao et al. [24] and Gonzalez et al. [25]–[27], the texture coordinates of a replacement vertex have not been sufficiently optimized yet.

Other strategies have also been proposed for appearance preservation in texture-mapped model simplification. García and Patow [28] propose a texturing technique named Inverse Geometric Textures (IGT), which defines texture coordinates for all the vertices in a simplified model to preserve texture details from a high resolution reference model. Chen and Chuang [29] and Coll and Paradinas [30] modify the texture image to minimize the texture distortion caused by each edge collapse. It is worth noting that the methods of García and Patow [28], Chen and Chuang [29] and Coll and Paradinas [30] are not well suited for embedded LOD construction, as the texture coordinates of inherited vertices or the texture image content keeps changing during the iterative simplification process.

Cheng and Boulanger [31] simplify a mesh and adaptively compress its texture image based on scale-space filtering (SSF). They parameterize the mesh into an image and, through SSF on this image, identify feature points to be included in the simplified mesh. The texture image is divided into blocks and each block is compressed to a quality level determined by the number of feature points falling inside. The mesh parameterization conducted in this algorithm is quite restricted, limiting its applicability on geometrically complex models.

III. ALGORITHM OVERVIEW
Given a texture-mapped model, we simplify the mesh to maximally preserve the model appearance at reduced vertex budgets. As the distortion in model appearance comes from geometry approximation error and texture deviation on the surface, we strive to minimize both during the simplification process. The pseudo-code is given in Alg. 1 and further explained in the following.

The mesh simplification is based on iterative edge contraction. The contraction cost is computed for every edge and the edge with the minimum cost is contracted to a replacement vertex at each iteration. Following Garland and Heckbert [12], we compute the geometric position of a replacement vertex by minimizing a quadric error metric. Beyond that, we make particular efforts to keep the quality of surface parameterization or, equivalently, minimize the texture deviation on the surface. Specifically, we propose
Algorithm 1 Mesh Simplification With Appearance-Driven Optimizations

**Input:**
- \( M \), original mesh;
- \( I \), texture image;
- \( R_m \), mesh decimation rate;

**Output:**
- \( M' \), simplified mesh;
1: \( M' = M . duplicate() \);
2: \( vBudget = M . vertex . Count() \times (1 - R_m) \);
3: \( Q = construct . Ordered . Edge . Priority . Queue(M', I) \);
4: **while** \( M' . vertex . Count() > vBudget \) **do**
5: \( e = Q . pop() \);
6: \( M' . contract . Edge . And . Update . Queue(e, Q, I) \);
7: **end while**

The edges are always ordered on their contraction costs, and the edge with the least contraction cost is always picked for contraction at each iteration.

A triangular mesh is usually a piecewise linear approximation to a smooth or piecewise smooth surface, and the approximation accuracy is often positively related to the sampling density. With the QEM described above, denser vertex populations, and correspondingly higher geometric approximation accuracies, will be produced for regions with more shape varieties. This is preferable since human eyes are usually more sensitive to distortions in sharp features than in smooth areas.

Following a similar rationale, texture coordinates at mesh vertices provide a piecewise linear approximation to a smooth or piecewise smooth surface parameterization, and we should retain denser samplings for surface regions with more color varieties (texture details) to achieve better parametric approximation accuracies and therefore less texture deviations for those regions. For this purpose, we propose to incorporate into the QEM (defined in (1)) two factors, \( d_c \) and \( d_e \), which measure the color variety and the edge saliency, respectively, for a local area in the texture image, \( I \). Specifically, we define the local region (or set of pixels), \( A \), in the texture image for an edge, \((v_1, v_2)\), as the parameterized 1-ring neighborhood of this edge (i.e., the union of the parameterized 1-ring neighborhoods of the two end vertices, \( v_1 \) and \( v_2 \)), denote the corresponding region in the edge map (obtained by applying the Canny operator [34] on the \( L^*a^*b^* \)-channel), \( Y \), of the texture image by \( E \), and update the error metric as

\[
\xi^1(v) = d_c . d_e . \xi^0(v),
\]

\[
d_c = \sum_{s \in A} \left\| I(s) - \frac{1}{|A|} \sum_{t \in A} I(t) \right\|_2,
\]

\[
d_e = \sum_{e \in E} Y(e).
\]

As \( d_c \) and \( d_e \) are constants for this edge contraction, the replacement vertex, \( v_o^1 \), and the contraction cost, \( C^1(v_1, v_2) \), are computed as

\[
v_o^1 = \arg\min_v \xi^1(v)
\]

\[
= \arg\min_v \xi^0(v)
\]

\[
= v_o^0
\]
Heckbert [12]. Its texture coordinates are initially assigned
its position is computed by the method of Garland and
three or more texture patches.

Classification of vertices. As shown in (a), (b) and (c),
smooth vertex, crease vertex and
classification of vertices is illustrated by
incident corners have more than two different 2D texture
a crease vertex is on the common boundary of two adjacent
smooth vertex is inside a texture patch,
discrete set of patches in the 2D texture image. On the texture
It is often the case that a 3D surface is parameterized onto a
attribute. A vertex is a
vertex are shown in Fig. 2 (a), (b) and (c), respectively.
FIGURE 2. Classification of vertices. As shown in (a), (b) and (c),
coordinates. This classification of vertices is illustrated by
a corner vertex with
It is often the case that a 3D surface is parameterized onto a

Since the new cost metric takes into account the texture
image characteristics as well, the simplification process leads
to a texture-adaptive primitive distribution on the mesh
surface. As can be seen from Fig. 1, more vertices tend to
distribute in surface regions with more texture details.

B. LOCAL MAPPING OPTIMIZATION
Following Isenburg and Snoeyink [35], we classify vertices
based on the configurations of their incident corners’ texture
attributes. A vertex is a smooth vertex if all its incident
corners have the same 2D texture coordinates; a vertex is a crease vertex if its incident corners are divided to two
sets each containing contiguous corners sharing the same
2D texture coordinates; a vertex is a corner vertex if its
incident corners have more than two different 2D texture
coordinates. This classification of vertices is illustrated by
Fig. 2 where examples of smooth vertex, crease vertex and
corner vertex are shown in Fig. 2 (a), (b) and (c), respectively.
It is often the case that a 3D surface is parameterized onto a
discrete set of patches in the 2D texture image. On the texture
mapped 3D surface, a smooth vertex is inside a texture patch,
a crease vertex is on the common boundary of two adjacent
texture patches, and a corner vertex is at the intersection of
three or more texture patches.

Assume that we contract edge \((v_1, v_2)\) to replacement
vertex \(v_0\). If both \(v_1\) and \(v_2\) are smooth vertices, \(v_0\) is also smooth.
Its position is computed by the method of Garland and
Heckbert [12]. Its texture coordinates are initially assigned
the average of \(v_1\)’s and \(v_2\)’s texture coordinates and further
adjusted by our local mapping optimization method, as shown in
Fig. 3(a). If one of \(v_1\) and \(v_2\) is a smooth vertex and the
other is not, \(v_0\) inherits the type, the geometric position and
the incident corners’ texture attributes from the non-smooth
one, as shown in Fig. 3(b). If both \(v_1\) and \(v_2\) are crease vertices
on the same crease, as shown in Fig. 3(c), the geometric
position and the corners’ texture attributes of \(v_0\) are obtained
by averaging those of \(v_1\) and \(v_2\). For other cases, contracting
edge \((v_1, v_2)\) may introduce serious texture deviation and is
therefore prohibited in our scheme. Details of our proposed
local mapping optimization method are given in the following
subsections.

1) TEXTURE DEVIATION
We adapt the texture deviation metric proposed by
Cohen et al. [21] to measure the mapping distortion caused by
an edge contraction. Given a 2D point, \(t\), in the texture
space, the Euclidean distance between its 3D positions on two
simplified versions of the mesh, \(M_i\) and \(M_j\), gives the texture
development, \(E_{i,j}(t)\), formally defined as:

\[
E_{i,j}(t) = \left\| F_i^{-1}(t) - F_j^{-1}(t) \right\|_2 \tag{5}
\]

where \(F_i(F_j)\) is the function mapping the 3D surface of
\(M_i(M_j)\) to the 2D texture domain and \(F_i^{-1}(F_j^{-1})\) does the reverse. Cohen et al. [21] measure the texture deviation at
the vertices of the overlayed parameterizations of \(M_i\) and
\(M_j\) and take the maximum as the texture deviation between
\(M_i\) and \(M_j\). Although this provides an upper bound, it may
not completely describe the texture deviation for the whole
affected region in the texture domain. Instead, we integrate
the texture deviation at all the locations in texture region, \(R\),
that is affected by the simplification from \(M_i\) to \(M_j\). Specifi-
cally, we propose to compute the texture deviation between
\(M_i\) and \(M_j\) as

\[
e_{i,j}^0(R) = \sum_{t \in R} E_{i,j}(t) \tag{6}
\]

Denoting the original model as \(M_0\), the incremental texture
development between any two adjacent LODs \(M_{i-1}\) and \(M_i\),
\(i > 0\), may be non-zero only in the parameterized 1-ring
neighborhood of the contracted edge.
2) TEXTURE COORDINATE OPTIMIZATION

Suppose that we are to contract edge \( e = (v_1, v_2) \) in \( M_{i-1} \) to produce \( M_i \). We denote the \( k \) 1-ring neighbor vertices of \( e \) (i.e., vertices connected to \( v_1 \) and/or \( v_2 \) by an edge) as \( b_1, b_2, \ldots, b_k \), and their parameterized positions as \( r_1, r_2, \ldots, r_k \), respectively. The texture domain region, \( R_{i-1} \), affected by this edge contraction is the polygonal region bounded by \( r_1, r_2, \ldots, r_k \). Further, we assume that \( v_1 \) and \( v_2 \) are parameterized to \( t_1 \) and \( t_2 \), respectively.

Cohen et al. [21] confine the texture coordinates of the replacement vertex to the kernel of the polygon \( R_{i-1} \), which is formed by the intersection of a set of half-planes, each inside a distinct edge of \( R_{i-1} \). They consider a discrete set of texture domain locations which includes the parameterized positions of the two end vertices, their middle point and the center of three selected corners of the kernel polygon. In contrast, our objective is to find the optimal texture coordinates, \( t_o \), for the replacement vertex, \( v_o \), from the full region of \( R_{i-1} \) such that \( \epsilon_{i-1,i}(R_{i-1}) \) is minimized.

In order to make a closed-form formulation, we first construct a fitting parametric surface \( f_{i-1}(t) = (x(t), y(t), z(t)) : R^2 \rightarrow R^3 \) over the region \( R_{i-1} \). Each of \( x(t), y(t) \) and \( z(t) \) is modeled as a second-order polynomial over the parameter domain, which is obtained by the least square fitting to the corresponding coordinate components of \( v_1, v_2, b_1, b_2, \ldots, b_k \) at the parameter locations, \( t_1, t_2, r_1, r_2, \ldots, r_k \), respectively. We are motivated to use second-order polynomials by the fact that polynomial surfaces (e.g., Bézier surfaces and many subdivision surfaces) are prevalently used for global or local mesh parameterization and, in particular, moving least squares (MLS) is frequently used to approximate local surfaces with second-order polynomials over local parameter domains. Correspondingly, we modify the texture deviation metric on \( R_{i-1} \) to:

\[
\epsilon_{i-1,i}(R_{i-1}) = \sum_{t \in R_{i-1}} \left\| f_{i-1}(t) - F_{i-1}^{-1}(t) \right\|_2^2. \tag{7}
\]

We approximate the computation of \( \epsilon_{i-1,i}(R_{i-1}) \) by an elaborate sampling on \( R_{i-1} \). Assuming that the replacement vertex is parameterized to \( \bar{t} \), we consistently sample each triangle incident on \( \bar{t} \) in the texture domain and its corresponding triangle incident on \( v_o \) on the surface of \( M_i \).

Specifically, we consistently sample every triangle using the same set, \( L = \{l_1, l_2, \ldots, l_m\} \), of barycentric coordinates where \( l_j = (\lambda_{1,j}, \lambda_{2,j}, \lambda_{3,j}) \) gives the barycentric coordinates of the \( j \)-th \( (j = 1, 2, \ldots, m) \) sample. Correspondingly, we modify our texture deviation metric as

\[
\epsilon_{i-1,i}(R_{i-1}, \bar{t}) = \sum_{l = 1}^{k} \sum_{h = 1}^{m} a_h \left\| f_{i-1}(l \cdot \Delta h, \bar{t}) - l_j \cdot \Delta h, v_o \right\|_2^2. \tag{8}
\]
where $\Delta_{h,\text{f}} = (r_h, r_{(h+1)\%k}, \overline{t})$ and $\Delta_{h,\text{v}_o} = (b_h, b_{(h+1)\%k}, \text{v}_o)$ with % standing for the modulo operator, operator $\cdot$ means dot product between two 3-tuples and $a_h$ is the area of $\Delta_{h,\text{v}_o}$.

Finally, we obtain $t_o$ as

$$t_o = \arg\min_{\overline{t}} \frac{s^2}{\overline{t}^2} (R_{i-1}, \overline{t}), \quad s.t. \overline{t} e R_{i-1}. \quad (9)$$

### TABLE 3. Mean and variance statistics of the users’ scores on the simplified models’ visual similarity to the original.

| Model  | Method | Decimation Rate | Mean Score | Variance |
|--------|--------|-----------------|------------|----------|
| Panda  | QSLIM  | 85%             | 8.85       | 1.43     |
|        | NQM    | 85%             | 8.91       | 1.73     |
|        | IMGD   | 85%             | 6.73       | 2.28     |
|        | MAYA   | 85%             | 8.83       | 1.65     |
|        | MSADO  | 85%             | 8.45       | 1.95     |
| House  | QSLIM  | 20%             | 9.18       | 1.71     |
|        | NQM    | 80%             | 8.28       | 1.97     |
|        | IMGD   | 20%             | 8.67       | 1.41     |
|        | MAYA   | 20%             | 8.31       | 1.68     |
|        | MSADO  | 20%             | 8.56       | 1.61     |
| Cartoon-man | QSLIM  | 80%             | 8.25       | 1.27     |
|          | NQM    | 80%             | 8.72       | 2.19     |
|          | IMGD   | 10%             | 8.92       | 1.93     |
|          | MAYA   | 10%             | 8.42       | 1.75     |
|          | MSADO  | 10%             | 8.94       | 1.07     |
| Earth  | QSLIM  | 90%             | 8.56       | 1.30     |
|        | NQM    | 90%             | 9.14       | 1.22     |
|        | IMGD   | 90%             | 8.92       | 1.52     |
|        | MAYA   | 90%             | 8.42       | 1.75     |
|        | MSADO  | 90%             | 8.75       | 2.17     |
| Spiderman | QSLIM  | 80%             | 8.00       | 1.60     |
|          | NQM    | 90%             | 9.00       | 1.35     |
|          | IMGD   | 80%             | 7.33       | 1.71     |
|          | MAYA   | 20%             | 9.04       | 1.36     |
|          | MSADO  | 80%             | 8.03       | 1.86     |
| Dragon | QSLIM  | 99%             | 7.50       | 1.98     |
|        | NQM    | 99%             | 7.80       | 1.75     |
|        | IMGD   | 99%             | 8.80       | 1.38     |
|        | MAYA   | 99%             | 8.89       | 1.41     |
|        | MSADO  | 99%             | 8.89       | 1.41     |

**TABLE 2.** Statistics of running time (s) for mesh simplification.

| Model    | QSLIM | NQM | IMGD | MAYA | MSADO |
|----------|-------|-----|------|------|-------|
| Panda    | 0.417 | 1.754 | 8.101 | 110  | 5.105 |
| House    | 0.681 | 5.436 | 2.084 | 215  | 7.201 |
| Cartoon-man | 0.530 | 3.022 | 1.683 | 521  | 6.388 |
| Earth    | 0.226 | 0.837 | 453.486 | 3.639 |
| Spiderman | 0.218 | 0.720 | 324.063 | 2.807 |
| Dragon   | 3.097 | 19.274 | 14.317 | 101.10 | 37.716 |
We solve this constrained nonlinear optimization problem using a quasi-Newton method. We set the initial value of $t_0$ to the average of $t_1$ and $t_2$. If no feasible solution was found, i.e., the optimal $\tilde{t}$ found by $\arg\min_{\tilde{t}} \epsilon_{i-1,i}^2 (R_{i-1}, \tilde{t})$ is out of $R_{i-1}$, we simply use the initial value.

In order to demonstrate the effectiveness of the proposed local mapping optimization technique, we show partially in Fig. 4 the Panda mesh (see Section V-A) simplified by 50% using our scheme (a) without and (b) with local mapping optimization. For reference, the original is shown in Fig. 4(c). It is evident from Fig. 4 that the proposed local mapping optimization technique significantly reduces the texture deviation, especially around the outer boundaries of the eye regions and the line across the chest.

V. EXPERIMENTS

A. TEST MODELS AND DISTORTION METRIC

Six texture-mapped models are used in our experiments. They are Panda, House, Cartoon-man, Earth, Spiderman and Dragon, whose geometric meshes and texture images are shown in Fig. 5. The original Cartoon-man model is from Intel. The original House model is from http://www.turbosquid.com, the original Dragon model is from https://graphics.stanford.edu/data/3Dscanrep/ and the other original models are from http://www.cgjoy.com. Note that, for our easy use, we have pre-processed the original geometric models and the original texture images and designed a texture for Dragon using commodity software tools. More information about these processed models is provided in Table 1. All the textured models are rendered with bilinear filtering in our experiments.

For the distortion measurement, we render both the original and the simplified 3D texture-mapped models using the same lighting from a randomly selected set of view points. At each view point, we compare the rendered images of the original and the simplified models using the well known SSIM (structural similarity) index [36]. We use the average SSIM index over all the view point samples to measure the quality, which is inversely related to the distortion of the simplified model. For convenience, we call it distortion metric as well.
B. RESULTS OF MESH SIMPLIFICATION

We experiment on simplifying 3D meshes with texture attributes, and compare with four representative benchmark algorithms: QSLIM (extended QEM with attributes incorporated) [18], NQM (new quadric metric) [19], IMGD (image-driven simplification) [23] and MAYA (Autodesk Maya 2017). For QSLIM, we use the software provided by the authors on http://www.cs.cmu.edu/afs/cs/user/garland/www/quadrics/qslim20.html. For the other two, we use our own implementations.

We run QSLIM, NQM, IMGD, MAYA and our proposed mesh simplification with appearance-driven optimizations (abbreviated as MSADO) method on the test models to simplify the meshes with texture attributes. Selected visual results are shown in Fig. 6, where the corresponding algorithm and decimation rate are marked under each model. For better illustration, both whole model images and enlarged local regions are shown. From Fig. 6, we observe that MSADO leads to better preserved model appearance for all the three test models.

Further, we measure the SSIM indices at various vertex decimation rates for all the test models, which are plotted in Fig. 7. From the curves in Fig. 7, we observe that MSADO leads to much higher SSIM indices than IMGD. Compared with NQM, QSLIM and MAYA, MSADO also produces higher SSIM indices, especially at higher decimation rates. These curves again demonstrate the advantage of MSADO over IMGD, NQM, QSLIM and MAYA in simplifying 3D meshes with texture attributes. It is worth mentioning that the proposed algorithm preserves the original model’s surface area very well. For instance, even at a high decimation rate of 80%, the average ratio of area between the simplified and the original models is around 99% for our test models.

The running time for Panda, Spiderman, House, Cartoonman, Earth and Dragon at the rate of 85%, 80%, 80%, 80%, 80%, 80%, respectively, is shown in Table 2. The experiments are conducted on a desktop computer with 12GB memory, 3.40GHz Intel(R) Xeon(R) CPU. Table 2 shows that QSLIM achieves the best efficiency, MSADO runs slower than NQM, while IMGD consumes the most time. The computation cost of MSADO comes mainly from the texture coordinate
optimization which involves parametric surface fitting and constrained nonlinear optimization. Note that our current research code for MSADO is not optimized yet.

Finally, we emphasize the effectiveness of our algorithm on the large Dragon model by the visual results in Fig. 8, where the corresponding decimation rate is marked under each model. From Fig. 8, we observe that a fair-quality approximation already shows up at an extremely high decimation rate of 99%, and the simplified model is almost indistinguishable from the original at the high decimation rate of 80%.

C. USER STUDY

We asked 36 subjects to evaluate the visual quality of simplified models. Specifically, we simplified each test model by a couple rates using different methods, and asked the subjects to grade the visual similarity of each resultant model to the original. In our design, the score ranges from 1 to 10 with 1 representing the lowest similarity and 10 the highest. Statistics of the responses to the questionnaire is shown in Table 3. We observe from Table 3 that, for almost all the models and decimation rates, MSADO achieves the highest mean and the lowest variance of the similarity scores. This user study again corroborates the outstanding performance of MSADO.

D. ABLATION STUDY

In this section, we demonstrate the effects of the major novel components in our proposed scheme. As the baseline, we use the basic QEM-based method [12] for geometry simplification and obtain a replacement vertex’ texture coordinates by averaging those of the two end vertices. Starting from the baseline, we incrementally add our proposed components to obtain Baseline+W (with $d_e$ and $d_t$ introduced as weights to the edge contraction cost metric), Baseline+W+TC (with texture coordinate optimization further introduced).

Selected results of Baseline, Baseline+W and Baseline+W+TC on Panda, House and Spiderman are shown in Fig. 9. Comparing the results of Baseline+W and Baseline, we observe reduced texture deviation around regions with sharp textural features; comparing the results of Baseline+W+TC and Baseline+W, we observe further reduced texture deviation.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an original scheme to simplify texture-mapped 3D triangular meshes. It strives to maximize the preservation of the original model appearance at any mesh decimation rate.

There are a few most important novelties of the proposed scheme. Firstly, we optimize the edge contraction cost metric to guide the adaptive distribution of 3D primitives on the simplified surface according to both geometric and textural characteristics. Secondly, we propose to optimize the texture coordinates of each replacement vertex using a closed-form formulation of texture deviation on a more complete sampling of a local texture domain. As a result, the proposed scheme achieves outstanding appearance-preservation performance, as experimentally demonstrated. Further, it should be straightforward to extend the proposed scheme for other types of 3D model simplification operators as well.

In the future, we may extend our current scheme for optimal progressive compression of texture-mapped 3D models. It is also interesting to investigate adaptive simplification of texture images in association with 3D meshes.

REFERENCES
Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” Signal, Image Video Process., vol. 7, no. 3, pp. 479–492, May 2013.

I. Garcia and G. Patow, “IGT: Inverse geometric textures,” ACM Trans. Graph., vol. 27, no. 5, pp. 1371–1379, 2008.

C.-C. Chen and J.-H. Chuang, “Texture adaptation for progressive meshes,” Comput. Graph. Forum, vol. 25, no. 3, pp. 343–350, Sep. 2006.

N. Coll and T. Paradinas, “Accurate simplification of multi-chart textured models,” Comput. Graph. Forum, vol. 29, no. 6, pp. 1842–1853, Sep. 2010.

I. Cheng and P. Boulanger, “Scale-space 3D TexMesh simplification,” in Proc. IEEE Int. Conf. Multimedia Expo (ICME), vol. 1, Jun. 2004, pp. 141–144.

J. Peng, C.-S. Kim, and C.-C. J. Kuo, “Technologies for 3D mesh compression: A survey,” J. Vis. Commun. Image Represent., vol. 16, no. 6, pp. 688–733, Dec. 2005.

A. Maglo, G. Lavoue, F. Dupont, and C. Hudelot, “3D mesh compression: Survey, comparisons, and emerging trends,” ACM Comput. Surveys, vol. 47, pp. 44–54, Feb. 2015.

J. Canny, “A computational approach to edge detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

M. Isenburg and J. Snoeyink, “Compressing the property mapping of polygon meshes,” Graph. Models, vol. 64, no. 2, pp. 114–127, Mar. 2002.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, Apr. 2004.

C. González, P. Castelló, M. Chover, M. Sbert, M. Feixas, and J. Gumbau, “Simplification method for textured polygonal meshes based on structural appearance,” Signal, Image Video Process., vol. 7, no. 3, pp. 479–492, May 2013.

I. Cheng and P. Boulanger, “Scale-space 3D TexMesh simplification,” in Proc. IEEE Int. Conf. Multimedia Expo (ICME), vol. 1, Jun. 2004, pp. 141–144.

YONGZHANDG XIAN received the B.S. degree from Chongqing University, in 2014. He is currently pursing the Ph.D. degree in electronics engineering and computer science with Peking University. His research interest includes geometry processing.

J. Peng, C.-S. Kim, and C.-C. J. Kuo, “Technologies for 3D mesh compression: A survey,” J. Vis. Commun. Image Represent., vol. 16, no. 6, pp. 688–733, Dec. 2005.

XINGXU MENG (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Shandong University, China, and the Ph.D. degree in computer science and technology from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a Professor of computer science with Shandong University. He has published more than 100 articles and four books, covering topics such as photorealistic rendering, geometric modeling, intelligent natural interaction, and service computing. His research interests include computer graphics, computer-aided design, human–computer interaction, and grid computing. He serves on the Editorial Boards of the Journal of Computer-Aided Design and Computer Graphics, the Chinese Journal of Computers, and Computer Integrated Manufacturing Systems.

YAN HUANG received the B.S. degree in computer science from Peking University, in 1997, and the M.S. and Ph.D. degrees in computer science from the University of California Irvine, in 2003 and 2009, respectively. She is currently an Associate Professor with the School of Software, Shandong University, China. Her research interests mainly reside in large scale 3D data visualization and intelligent multimedia data analysis.

YAN HUANG received the B.S. degree from Peking University, in 1997, and the M.S. and Ph.D. degrees in computer science from the University of California Irvine, in 2003 and 2009, respectively. She is currently an Associate Professor with the School of Software, Shandong University, China. Her research interests mainly reside in large scale 3D data visualization and intelligent multimedia data analysis.

YAN HUANG received the B.S. degree from Peking University, in 1997, and the M.S. and Ph.D. degrees in computer science from the University of California Irvine, in 2003 and 2009, respectively. She is currently an Associate Professor with the School of Software, Shandong University, China. Her research interests mainly reside in large scale 3D data visualization and intelligent multimedia data analysis.

JINGLIANG PENG (Member, IEEE) received the B.S. and M.S. degrees in computer science from Peking University, in 1997 and 2000, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, in 2006. He is currently a Professor with the School of Software, Shandong University, China. His research interests mainly reside in digital geometry processing, virtual/augmented reality, and image/video analysis.

**GUOPI WANG** received the B.S. and M.S. degrees from the Harbin Institute of Technology, Harbin, China, in 1987 and 1990, respectively, and the Ph.D. degree from Fudan University, Shanghai, in 1997, all in mathematics. He was an Associate Professor with the Department of Computer Science and Technology, Peking University, Beijing, from 1999 to 2002. He has been serving as the Director of the HCI and Multimedia Laboratory and a Professor with the School of Electronics Engineering and Computer Science, Peking University, since 2002, where he currently serves as the Deputy Director of the Institute of Software. He is also the Director of the Beijing Engineering Research Center of Virtual Simulation and Visualization (BERCVSV). His research interests include computer graphics, virtual reality, geometrical modeling, CAD, and human–computer interaction.