IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2020, accepted March 23, 2020, date of publication April 15, 2020, date of current version September 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2987939

Mesh Simplification With Appearance-Driven Optimizations

YONGZHANG XIAN[1], YUXUE FAN[2], YAN HUANG[3], GUOPING WANG[1], CHANGHE TU[4], XIANGXU MENG[3], (Member, IEEE), AND JINGLIANG PENG[3], (Member, IEEE)

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

The associate editor coordinating the review of this manuscript and approving it for publication was Walter Didimo.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

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**

to 1) incorporate the local textural richness as a factor into the edge contraction cost metric, favoring better preservation of triangles in surface regions with more texture details, and 2) derive the texture coordinates of a replacement vertex using closed-form optimization.

It is worth noting that, while we focus on mesh simplification in this work, the proposed method may directly substitute into edge-contraction based progressive mesh encoders (as surveyed by Peng *et al.* [32] and Maglo *et al.* [33]).

IV. ALGORITHM DETAILS

As stated in Section III, we use the same method for computing each replacement vertex' position as many of the others (*e.g.*, Garland and Heckbert [12]), except that special processing is needed when one or two of the end vertices fall on a boundary between two texture patches (see Section IV-A). Uniquely, we control the texture deviation by an optimized quadric error metric based on both mesh and texture image features and a closed-form local mapping optimization method to determine the texture coordinates for each replacement vertex. Note that, in this work, we represent all texture image colors in the CIELAB (or $L^*a^*b^*$) space, a perceptually uniform color space.

A. OPTIMIZED QUADRIC ERROR METRIC

A standard mesh simplification algorithm based on the quadric error metric works in the following way.

Initially, it associates with each vertex in the original mesh the list of facets incident on that vertex. Later on, whenever it contracts an edge, (v_1,v_2), to a replacement vertex, v, it forms the associated facet list of v by combining those of v_1 and v_2.

If it contracts an edge (v_1,v_2) to v, the approximation error, $\xi^0(v)$, at v is computed by a quadric error metric [12] that sums up the squared point-plane distances from v to the associated facets of v, i.e.,

$$\xi^0(v) = v^T \cdot \left(\sum_{f \in F} P_f P_f{}^T \right) \cdot v = v^T \cdot Q \cdot v \qquad (1)$$

where $v = \left[v_x, v_y, v_z, 1\right]^T$, F is the associated facet list of v and $P_f = \left[a_f, b_f, c_f, d_f\right]^T$ represents the plane of f defined by $a_f x + b_f y + c_f z + d_f = 0$. The replacement vertex, v_o^0, and the contraction cost, $C^0(v_1, v_2)$, are computed as

$$\begin{aligned} v_o^0 &= \underset{v}{argmin}\, \xi^0(v), \\ C^0(v_1, v_2) &= \xi(v_o^0). \end{aligned} \qquad (2)$$

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^*-channel), Y, of the texture image by E, and update the error metric as

$$\begin{aligned} \xi^1(v) &= d_c d_e \xi^0(v), \\ d_c &= \sum_{s \in A} \left\| I(s) - \frac{1}{|A|} \sum_{t \in A} I(t) \right\|_2, \\ d_e &= \sum_{e \in E} Y(e). \end{aligned} \qquad (3)$$

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

$$\begin{aligned} v_o^1 &= \underset{v}{argmin}\, \xi^1(v) \\ &= \underset{v}{argmin}\, \xi^0(v) \\ &= v_o^0, \end{aligned}$$

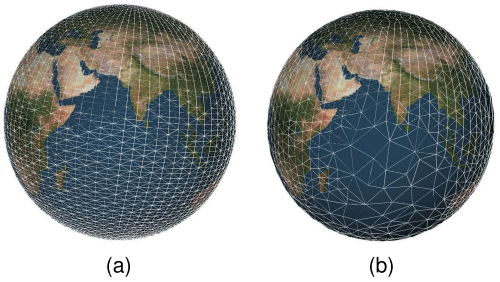


FIGURE 1. The original earth mesh as shown in (a) is simplified by 30% as shown in (b), both rendered with the original texture image.

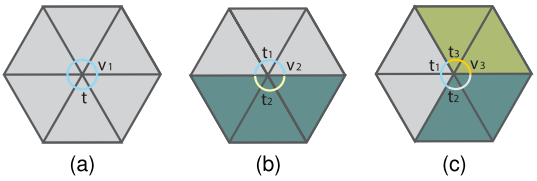


FIGURE 2. Classification of vertices. As shown in (a), (b) and (c), respectively, v_1 is a smooth vertex with incident corners' texture coordinates of t, v_2 a crease vertex with t_1 and t_2, and v_3 a corner vertex with t_1, t_2 and t_3.

$$\begin{aligned} C^1(v_1, v_2) &= \xi^1(v_o^1) \\ &= d_c d_e \xi^0(v_o^0) \\ &= d_c d_e C^0(v_1, v_2). \end{aligned} \tag{4}$$

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_o. If both v_1 and v_2 are smooth vertices, v_o is also smooth. Its position is computed by the method of Garland and Heckbert [12]. Its texture coordinates are initially assigned

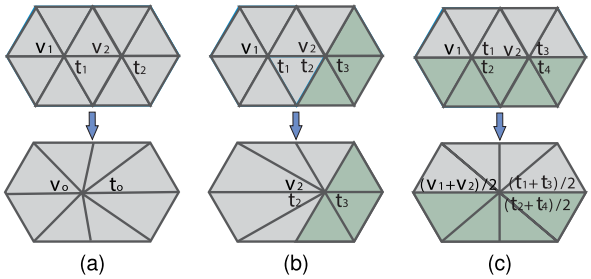


FIGURE 3. Different cases of edge contraction: (a) both v_1 and v_2 are smooth vertices, (b) only v_1 is a smooth vertex, and (c) both v_1 and v_2 are on the same crease.

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_o 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_o 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 deviation, $E_{i,j}(t)$, 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. Specifically, we propose to compute the texture deviation between M_i and M_j as

$$\varepsilon_{i,j}^0(R) = \sum_{t \epsilon R} E_{i,j}(t) \tag{6}$$

Denoting the original model as M_0, the incremental texture deviation 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.

(a) (b) (c)

FIGURE 4. Visual comparison of the Panda mesh simplified by 50% using our scheme (a) without and (b) with the local mapping optimization. The original Panda model is shown in (c) for reference.

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 $\varepsilon^0_{i-1,i}(R_{i-1})$ 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:

$$\varepsilon^1_{i-1,i}(R_{i-1}) = \sum_{t \in R_{i-1}} \left\| f_{i-1}(t) - F_i^{-1}(t) \right\|_2 . \tag{7}$$

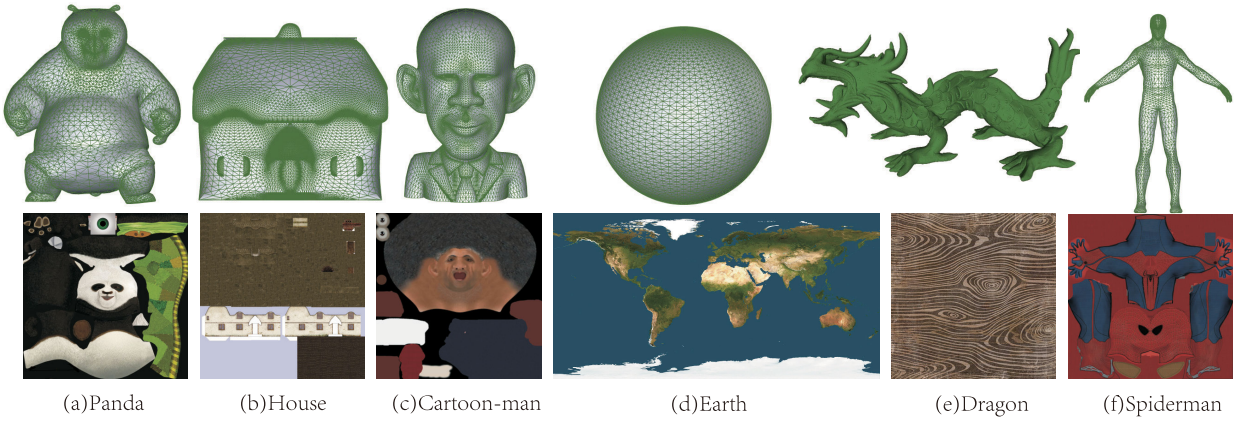
We approximate the computation of $\varepsilon^1_{i-1,i}(R_{i-1})$ by an elaborate sampling on R_{i-1}. Assuming that the replacement vertex is parameterized to $\bar{t}$, we consistently sample each triangle incident on $\bar{t}$ in the texture domain and its corresponding triangle incident on v_o on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

$$\varepsilon^2_{i-1,i}(R_{i-1}, \bar{t}) = \sum_{h=1}^{k} \sum_{j=1}^{m} a_h \left\| f_{i-1}(l_j \cdot \Delta_{h,\bar{t}}) - l_j \cdot \Delta_{h,v_o} \right\|_2 \tag{8}$$

TABLE 1. Statistics of models used in our experiments.

Model	#Vertices	#Faces	#Texture Patches	Texture Size
Panda	9,884	19,728	10	1,024× 1,024
House	19,438	40,832	23	1,024× 1,024
Cartoon-man	17,066	33,288	11	1,024× 1,024
Earth	6,242	12,480	1	512× 1,024
Spiderman	5,510	10,696	16	512× 512
Dragon	100,002	200,000	1	2,048× 2,048



(a)Panda (b)House (c)Cartoon-man (d)Earth (e)Dragon (f)Spiderman

FIGURE 5. Models used in the experiments. The geometric meshes are shown in the top row, and the corresponding texture images in the bottom.

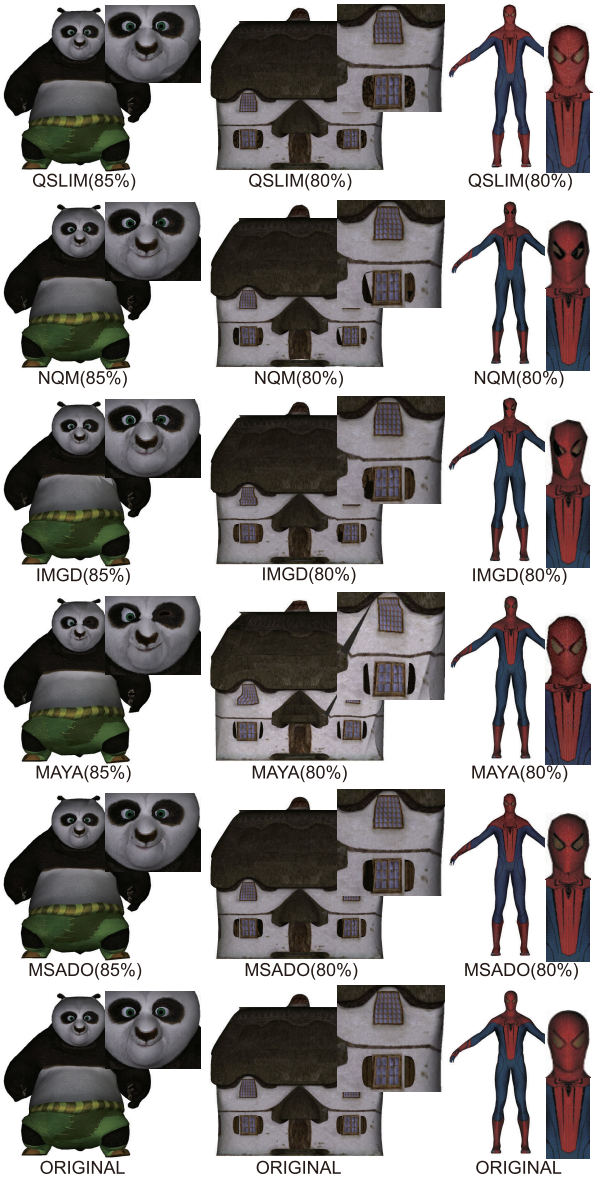


FIGURE 6. Visual results of the mesh simplification for Panda, House and Spiderman using QSLIM, NQM, IMGD, MAYA and MSADO. The algorithm name and decimation rate are marked below each model. The original models are shown for reference.

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

Model	QSLIM	NQM	IMGD	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	37.716

where $\Delta_{h,\bar{t}} = (r_h, r_{(h+1)\%k}, \bar{t})$ and $\Delta_{h,v_o} = (b_h, b_{(h+1)\%k}, v_o)$ with % standing for the modulo operator, operator $\cdot$ means dot product between two 3-tuples and a_h is the area of Δ_{h,v_o}.

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%	8.95	1.43
		30%	9.00	1.45
	NQM	85%	8.91	1.73
		30%	8.97	**1.36**
	IMGD	85%	6.73	2.28
		30%	8.83	1.65
	MAYA	85%	7.73	2.28
		30%	8.81	1.45
	MSADO	85%	**9.03**	**1.11**
		30%	**9.45**	1.59
House	QSLIM	80%	8.44	1.50
		20%	9.18	1.71
	NQM	80%	8.28	1.97
		20%	8.67	1.41
	IMGD	80%	7.25	1.81
		20%	8.81	1.69
	MAYA	80%	5.42	2.20
		20%	8.56	1.61
	MSADO	80%	**9.06**	**0.90**
		20%	**9.54**	**1.22**
Cartoon-man	QSLIM	80%	8.25	1.27
		10%	9.08	**1.25**
	NQM	80%	6.72	2.19
		10%	8.92	1.93
	IMGD	80%	6.58	2.18
		10%	8.92	1.52
	MAYA	80%	8.42	1.75
		10%	9.08	1.32
	MSADO	80%	**8.94**	**1.07**
		10%	**9.20**	1.44
Earth	QSLIM	90%	8.56	**1.30**
		30%	9.14	1.22
	NQM	90%	8.17	1.59
		30%	9.05	1.29
	IMGD	90%	6.00	2.24
		30%	9.08	1.32
	MAYA	90%	7.89	1.83
		30%	9.03	1.46
	MSADO	90%	**8.75**	2.17
		30%	**9.19**	**1.21**
Spiderman	QSLIM	80%	8.00	1.60
		20%	9.00	1.35
	NQM	80%	7.33	1.71
		20%	9.03	1.36
	IMGD	80%	5.81	2.32
		20%	9.08	1.30
	MAYA	80%	7.50	**1.59**
		20%	8.97	1.56
	MSADO	80%	**8.03**	1.86
		20%	**9.19**	**1.09**
Dragon	QSLIM	99%	7.50	1.98
		20%	8.80	1.75
	NQM	99%	7.39	1.79
		20%	8.86	1.38
	IMGD	99%	6.69	1.72
		20%	8.69	1.41
	MAYA	99%	7.08	1.96
		20%	8.78	1.48
	MSADO	99%	**8.53**	**1.63**
		20%	**9.03**	**1.32**

Finally, we obtain t_o as

$$t_o = \underset{\bar{t}}{argmin}\ \varepsilon^2_{i-1,i}(R_{i-1}, \bar{t}), \quad s.t.\ \bar{t} \epsilon R_{i-1}. \tag{9}$$

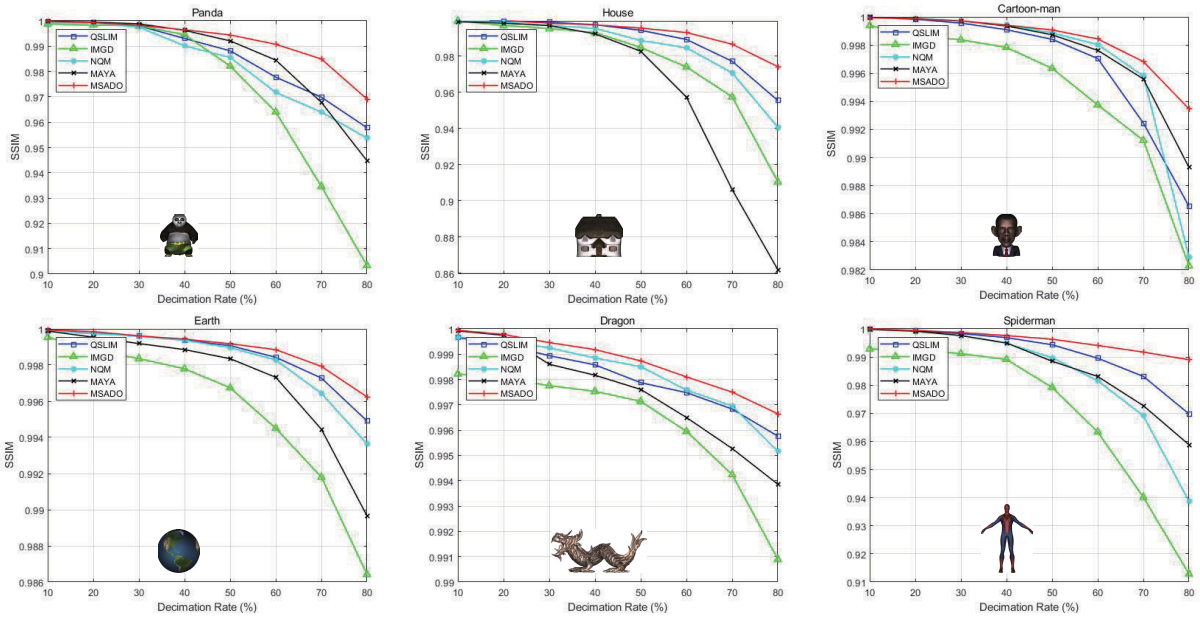


FIGURE 7. Rate-distortion curves of the mesh simplification for Panda, House, Cartoon-man, Earth, Dragon and Spiderman using QSLIM, NQM, IMGD, MAYA and MSADO.

FIGURE 8. Visual results of the mesh simplification for Dragon using MSADO. The corresponding decimation rate is marked below each model. The original model is shown for reference.

We solve this constrained nonlinear optimization problem using a quasi-Newton method. We set the initial value of t_o to the average of t_1 and t_2. If no feasible solution was found, *i.e.*, the optimal $\bar{t}$ found by $\underset{\bar{t}}{argmin}\ \varepsilon^2_{i-1,i}(R_{i-1}, \bar{t})$ 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.

FIGURE 9. Visual results of mesh simplification for Panda, House and Spiderman using Baseline, Baseline+W and Baseline+W+TC. The corresponding algorithm name and vertex decimation rate are marked below each model. The original models are shown for reference.

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_c and d_e 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

[1] P. Heckbert and M. Garland, "Multiresolution surface modelling," in *Proc. ACM SIGGRAPH Course Notes 25*, Los Angeles, CA, USA, Aug. 1997, p. 336.

[2] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithm," *Comput. Graph.*, vol. 22, no. 1, pp. 37–54, 1998.

[3] D. P. Luebke, "A developer's survey of polygonal simplification algorithms," *IEEE Comput. Graph. Appl.*, vol. 21, no. 1, pp. 24–35, May 2001.

[4] J. Talton, "A short survey of mesh simplification algorithms," Dept. Comput. Sci., Univ. Illinois at Urbana, Urbana, IL, USA, Tech. Rep. Course CS 598 MJG, 2004.

[5] G. Turk, "Re-tiling polygonal surfaces," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 55–64, Jul. 1992.

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 905–914, Aug. 2004.

[7] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 65–70, Jul. 1992.

[8] W. J. Schroeder, "A topology modifying progressive decimation algorithm," in *Proc. Vis.*, 1997, pp. 205–212.

[9] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle, "Mesh optimization," in *Proc. 20th Annu. Conf. Comput. Graph. Interact. Techn.*, 1993, pp. 19–26.

[10] H. Hoppe, "Progressive meshes," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 99–108.

[11] R. Ronfard and J. Rossignac, "Full-range approximation of triangulated polyhedra.," *Comput. Graph. Forum*, vol. 15, no. 3, pp. 67–76, Aug. 1996.

[12] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. 24th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1997, pp. 209–216.

[13] P. S. Heckbert and M. Garland, "Optimal triangulation and quadric-based surface simplification," *Comput. Geometry*, vol. 14, nos. 1–3, pp. 49–65, Nov. 1999.

[14] M. Garland and Y. Zhou, "Quadric-based simplification in any dimension," *ACM Trans. Graph.*, vol. 24, no. 2, pp. 209–239, Apr. 2005.

[15] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes," in *Modeling in Computer Graphics*. Berlin, Germany: Springer, 1993, pp. 455–465.

[16] K. Low and T. Tan, "Model simplification using vertex-clustering," in *Proc. Symp. Interact. 3D Graph.*, 1997, pp. 75–85.

[17] P. Lindstrom, "Out-of-core simplification of large polygonal models," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2000, pp. 259–262.

[18] M. Garland and P. Heckbert, "Simplifying surfaces with color and texture using quadric error," in *Proc. 9th Conf. Vis.*, 1998, pp. 263–269.

[19] H. Hoppe, "New quadric metric for simplifiying meshes with appearance attributes," in *Proc. 10th Conf. Vis.*, 1999, pp. 59–66.

[20] J. Cohen, D. Manocha, and M. Olano, "Simplifying polygonal models using successive mappings," in *Proc. 7th Conf. Vis.*, 1997, pp. 395–402.

[21] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1998, pp. 115–122.

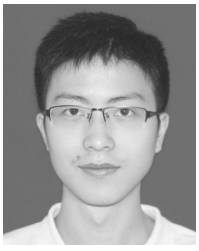
[22] N. Williams, D. Luebke, J. Cohen, M. Kelley, and B. Schubert, "Perceptually guided simplification of lit textured meshes," in *Proc. Symp. Interact. 3D Graph.*, 2003, pp. 113–121.

[23] P. Lindstrom and G. Turk, "Image-driven simplification," *ACM Trans. Graph.*, vol. 19, no. 3, pp. 204–241, Jul. 2000.

[24] Y. Shao, B. Liu, and H. Zhang, "Geometry and texture based simplification of 3d meshes," in *Proc. 7th Int. Conf. Signal Process. (ICSP)*, 2004, pp. 1310–1313.

[25] C. Gonzalez, P. Castello, and M. Chover, "A texture-based metric extension for simplification methods," in *Proc. Int. Conf. Comput. Graph. Theory Appl.*, 2007, pp. 69–76.

[26] C. Gonzalez, P. Castello, M. Chover, M. Sbert, and M. Feixas, "Viewpoint entropy-driven simplification method for textured triangle meshes," in *Proc. Int. Conf. Comput. Graph. Theory Appl.*, 2010, pp. 30–37.
[27] 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.
[28] I. Garcia and G. Patow, "IGT: Inverse geometric textures," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1371–1379, 2008.
[29] C.-C. Chen and J.-H. Chuang, "Texture adaptation for progressive meshes," *Comput. Graph. Forum*, vol. 25, no. 3, pp. 343–350, Sep. 2006.
[30] N. Coll and T. Paradinas, "Accurate simplification of multi-chart textured models," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1842–1853, Sep. 2010.
[31] 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.
[32] 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.
[33] 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.
[34] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
[35] M. Isenburg and J. Snoeyink, "Compressing the property mapping of polygon meshes," *Graph. Models*, vol. 64, no. 2, pp. 114–127, Mar. 2002.
[36] 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.



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

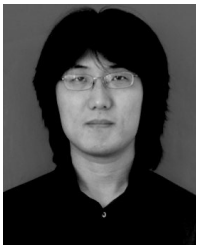
YUXUE FAN received the bachelor's and Ph.D. degrees from the School of Computer Science and Technology, Shandong University, China. She currently works with the Patent Examination Cooperation Center, State Intellectual Property Office. Her research interest focuses on the processing of textures and 3D models.

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.

GUOPING 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.

CHANGHE TU received the bachelor's, master's, and Ph.D. degrees from Shandong University. He is currently a Professor and the Associate Dean of the School of Computer Science and Technology, Shandong University. He has published articles on SIGGRAPH, Eurographics, ACM TOG, the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, and CAGD. His research interests include geometric modeling and processing, computational geometry, and data-driven visual computing.

XIANGXU 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*.



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.