EUROGRAPHICS 2006 / E. Gröller and L. Szirmay-Kalos
(Guest Editors)

*Volume 25 (2006), Number 3*

# Texture Adaptation for Progressive Meshes

Chih-Chun Chen[†] and Jung-Hong Chuang[‡]

Department of Computer Science, National Chiao Tung University, Taiwan, ROC

**Abstract**

*Level-of-detail modeling is a vital representation for real-time applications. To support texture mapping progressive meshes (PM), we usually allow the whole PM sequence to share a common texture map. Although such a common texture map can be derived by using appropriate mesh parameterizations that consider the minimization of geometry stretch, texture stretch, or even the texture deviation introduced by edge collapses, we have found that even with a well parameterized texture map, the texture mapped PM still reveals apparent texture distortion due to geometry changes and the nature of linear interpolation used by texture mapping hardware. In this paper, we propose a novel, simple, and efficient approach that adapts texture content for each edge collapse, aiming to eliminate texture distortion. A texture adaptation and its inverse are local and incremental operations that can be fully supported by texture mapping hardware, the render-to-texture feature, and the fragment shader. Once the necessary correspondence in the partition of texture space is built during the course of PM construction, the texture adaptation or its inverse can be applied on the fly before rendering the simplified or refined model with texture map. We also propose the mechanism of indexing mapping to reduce blurred artifacts due to under-sampling that might be introduced by texture adaptation.*

**Keywords:** texture mapping progressive meshes, mesh simplification, mesh parameterization, texture distortion

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismColor, shading, shadowing, and texture

## 1. Introduction

Mesh simplification has been an active area of research in real-time graphics. The ultimate goal of mesh simplification is to generate a simplified mesh of low polygon count that preserves the fidelity of the original mesh. Texture mapping has been very useful in enhancing shaded images with more surface or color detail. For a given mesh and its associated texture map, there are several possibilities of applying the texture map to the simplified meshes. One way is to have a texture map for each simplified mesh, which requires more artist work on texture design and more storage, especially for progressive meshes (PM). A more practical way is to have the entire PM sequence share a common texture map; however, when applying texture mapping to PMs, serious texture distortions are often observed. To reduce texture distortion, several schemes have been proposed. One is to consider texture deviation as an error metric, implying that edge collapses with higher texture deviation are more likely to be retained. This, however, cannot prevent texture distortion introduced by edge collapses that have been performed. In addition to using the metric of texture deviation, the texture map can be derived by using appropriate parameterizations that take into account the minimization of geometry and texture stretch, as well as the texture deviation introduced by edge collapses. It is observed that even with a very well parameterized texture map, the texture mapped PM still reveals significant texture distortion due to geometry changes and the nature of linear interpolation employed by texture mapping hardware.

Let's consider the 2D case shown in Figure 1, where edges $\overline{\mathbf{AC}}$ and $\overline{\mathbf{BC}}$ are simplified to $\overline{\mathbf{AB}}$. At the bottom, $T$ is the

[†] `chihchun@csie.nctu.edu.tw`

[‡] `jhchuang@csie.nctu.edu.tw`

texture map on which **A** maps to 0, **B** maps to 1, and **C** maps to 0.3. On the textured image of $\overline{\mathbf{AB}}$, we see the blue color shares a portion of $\overline{\mathbf{AB}}$ that is much smaller than expected. Figure 2 shows a planar polygonal mesh $M = M^2$ and the texture map $T$ associated with it. The textured images of simplified meshes $M^1$ and $M^0$ reveal serious texture distortion, as shown in the bottom row of the figure. In this example, geometry remains the same, but the nature of linear interpolation for texture mapping affects the textured image. Essentially, the texture coordinate within a triangle is piecewise linear but is no longer linear when crossing edges of the triangle. Such texture distortion has been routinely observed for texture mapping PMs using any existing technique.

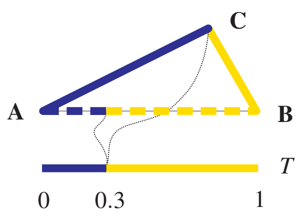

**Figure 1:** *Texture distortion introduced by geometry simplification.*

We present a novel, simple, and efficient approach that adapts texture content for each edge collapse, aiming to effectively eliminate the texture distortion introduced by geometry changes and the nature of linear interpolation employed by texture mapping hardware. The texture adaptation applied for each edge collapse is local to the region affected by the edge collapse and is applied to the adapted texture resulting from the previous edge collapse. The texture adaptation is invertible, that is, the backward texture adaptation can be performed for a vertex split. Both texture adaptation and backward texture adaptation can be fully supported by texture mapping hardware. Once the necessary correspondence in the partition of texture space is built during the course of PM construction, the texture adaptation or its inverse can be applied on the fly before rendering the simplified or refined model with texture map. We have observed that the proposed texture adaptation is capable of eliminating texture distortion in a time that is almost negligible. Figure 3 depicts that the textured images of $M^1$ and $M^0$ with texture adaptation are indistinguishable from the textured image of the original mesh.

## 2. Related work

Luebke et al. [LWC*02] already gave a complete and intensive review of model simplification. Here, we review the works that address the reduction of texture distortion resulting from model simplification. Hoppe [Hop96] proposed an continuous level-of-detail representation, called progressive mesh (PM). Garland and Heckbert [GH98] introduced a quadric error metric for measuring vertex-to-plane distances, which works for meshes with attributes. Bajaj and Schikore [BS96] presented a method to simplify meshes with multivariate data to within a given error bound. Cohen et al. [COM98] presented texture deviation to measure geometric accuracy when simplifying textured meshes. Lindstrom and Turk [LT00] proposed the image-based metrics that compare images of the original model with images of the simplified model. Xu et al. [XSX05] proposed a texture information driven simplification in which texture image frequency distribution and texture mapping distortion energy are combined to present the simplification error.

Cignoni et al. [CMR*99] presented a method to preserve surface attributes onto texture maps for each level of detail. Requiring a dedicated texture map for each level of detail implies the need for large storage space and long processing time. Sander et al. [SSGH01] proposed TMPM to parameterize the input mesh by taking into account the geometry and texture stretch as well as texture deviation introduced by edge collapses. The texture map derived from such a parameterization is then shared by the entire PM sequence. Kim and Wohn [KW01] proposed a method to minimize texture distortion by generating a texture map that can be mapped to all levels of detail in a pre-processing stage. A distortion metric is then used to guide the mesh simplification. Sander et al. [SGSH02] proposed a signal-specialized parameterization to minimize texture stretch by allocating more texture samples for areas of higher signal frequency. Khodakovsky et al. [KLS03] proposed a globally smooth parameterization with low distortion.

Several papers have been proposed to minimize texture distortion problem in a coarse-to-fine process. Eckstein et al. [ESG01] proposed a coarse-to-fine optimization to generate a proper texture coordinate for a newly refined vertex for supporting texture mapping multi-resolution meshes, and it always guarantees a solution by adding Steiner vertices. Zhou et al. [ZWT*05] proposed a technique called *Texture-Montage* to seamlessly map multiple texture images onto an arbitrary 3D model. One part of their work is to derive texture coordinates through optimization for all the vertices of the original mesh from the coarse texture coordinate assignments on the base mesh. Both texture stretch and texture color continuity are taken into account in this process.

## 3. Texture adaptation

### 3.1. Overview

For a given polygon model $M$ with a texture map $T$, the texture adaptation locally and incrementally adapts the texture map in the course of edge collapses that construct a PM of $M$. The goal is to eliminate the texture distortion introduced by the edge collapses. For a PM sequence, we then have a texture map $T^i$ associated with each reduced model $M^i$, that

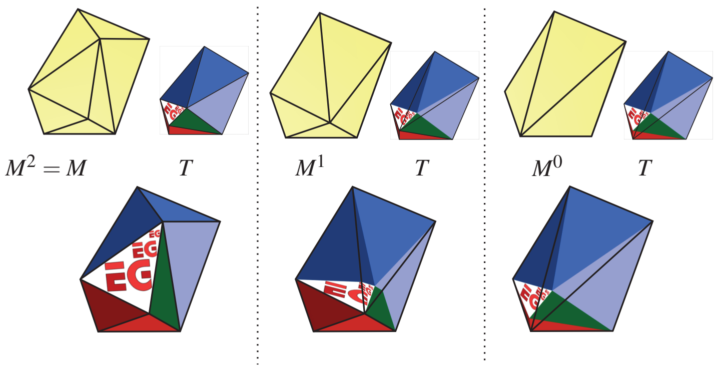

**Figure 2:** *Texture distortion introduced by edge collapses.*

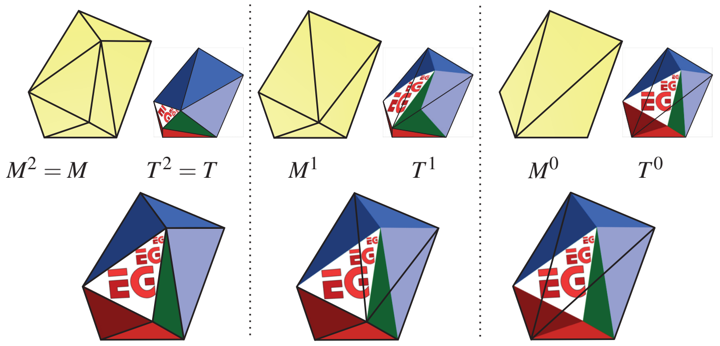

**Figure 3:** *Texture mapping progressive meshes with texture adaptation.*

is,

$$\begin{matrix} M = M^n \\ T = T^n \end{matrix} \xrightarrow{ecol_{n-1}} \cdots \xrightarrow{ecol_1} \begin{matrix} M^1 \\ T^1 \end{matrix} \xrightarrow{ecol_0} \begin{matrix} M^0 \\ T^0 \end{matrix} .$$

The texture adaptation operation from $T^i$ to $T^{i-1}$ is invertible, that is, for a $T^{i-1}$ associated with $M^{i-1}$, $T^i$ of $M^i$ can be derived by doing the inverse of the texture adaptation that brings $T^{i-1}$ to $T^i$, which we call *backward texture adaptation*. Consequently, for the sequence of vertex splits from $M^0$, we have

$$\begin{matrix} M^0 \\ T^0 \end{matrix} \xrightarrow{vsplit_0} \begin{matrix} M^1 \\ T^1 \end{matrix} \xrightarrow{vsplit_1} \cdots \xrightarrow{vsplit_{n-1}} \begin{matrix} M^n = M \\ T^n = T \end{matrix} .$$

We will see that the texture adaptation and its inverse, backward texture adaptation, involve the same operations, which can be done very efficiently by graphics hardware. And the texture adaptation or its inverse is applied to texture map and is performed on the fly while the model is simplified or refined, respectively.

For an edge collapse $ecol_{i-1}$ that reduces $M^i$ to $M^{i-1}$, the texture adaptation is local to the region $R_i$ that is the neighborhood of the collapsed edge in texture space and is performed incrementally from $T^i$. Since the boundary of $R_i$ remains fixed, what the texture adaptation does is basically find an appropriate texel of $T^i$ for each texel of $T^{i-1}$, all in the region $R_i$. We will see in the next subsection that the region $R_i$ can be respectively partitioned into the same number of cells for $T^i$ and $T^{i-1}$, and within each of these cells, the texture coordinates are piecewise linear. The correspondence between texels of $T^i$ and $T^{i-1}$ is then simplified to the correspondence between cells of $T^i$ and $T^{i-1}$. Once all pairs of corresponding cells are found, the texture adaptation can be performed by hardware texture mapping the cell of $T^i$ to the corresponding cell in $T^{i-1}$. Since cell correspondence is identical for each texture adaptation and its inverse, backward texture adaptation texture maps the cell of $T^{i-1}$ to the corresponding cell in $T^i$.

Essentially, texture adaptation is a re-sampling process that might pose problems of under sampling and introduce blurred artifacts. *Indexing mapping* is proposed to minimize these blurred artifacts. Indexing mapping involves an additional texture map, called the *indexing map*, that stores texture coordinates to the original texture map. During edge collapsing, texture adaptation is applied to this indexing map, leaving the original texture map alone. Only during rendering, the original texture map is mapped to the model via texture coordinates derived from the indexing map.

The cell partition and cell correspondence are derived while constructing the progressive meshes. The cell partition and cell correspondence are stored along with the information for each edge collapse and vertex split. In rendering the progressive meshes, the texture adaptation or backward texture adaptation is performed on the fly while the model is simplified or refined, respectively.

In the following sections, half-edge collapse is used in the illustration of the proposed method. Extension to the full-edge collapse is straightforward.

### 3.2. Cell correspondence between two consecutive levels

For a given mesh $M$, its texture map $T$ is obtained by parameterizing $M$ onto the texture plane. The parameterization of $M$ is a one-to-one mapping $F$ that maps each vertex $\mathbf{V}$ of $M$ to a point $\mathbf{v}$ on the texture plane. For a PM using half-edge collapse, the parameterization of $M^i$ is a subset of that for $M$. Figure 4 depicts the mapping of the neighborhood of an edge $\overline{\mathbf{UV}}$ on the texture plane, before and after $\overline{\mathbf{UV}}$ is collapsed.

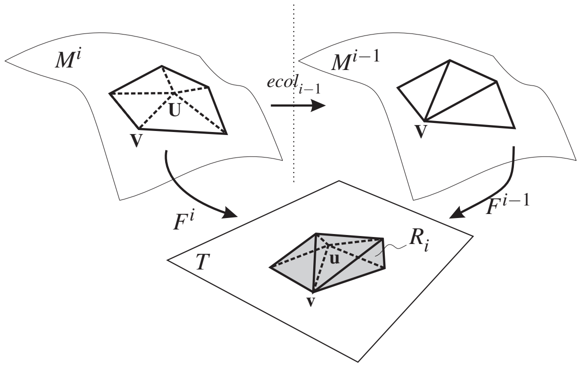

**Figure 4:** *Mesh parameterizations before and after an edge collapse.*

When we overlay the edges in the region $R_i$ before and after edge collapse $ecol_{i-1}$ that reduces $M^i$ to $M^{i-1}$, edges on the overlay may intersect each other and hence partition $R_i$ into cells, as shown in Figure 5, where $\overline{\mathbf{un}}$ intersects $\overline{\mathbf{vw}}$ at $\mathbf{a}$ and another pair of edges intersect at $\mathbf{b}$ and $R_i$ is partitioned into 9 cells. However, the two intersecting edges on the texture plane are generally not coplanar in 3D space. Nevertheless, for each pair of intersecting edges we can compute the pair of nearest points, one on an edge and one on another edge. Taking the mesh in Figure 5 as an example, we compute the pair of the nearest points $\mathbf{A}_i$ and $\mathbf{A}_{i-1}$, where $\mathbf{A}_i$ is on $\overline{\mathbf{UN}}$ of $M^i$ and $\mathbf{A}_{i-1}$ on $\overline{\mathbf{VW}}$ of $M^{i-1}$. The points $\mathbf{A}_i$ and $\mathbf{A}_{i-1}$ can be derived by minimizing the distance of points on the two edges, which amounts to solving the linear system

$$\begin{bmatrix} \vec{u}\cdot\vec{u} & -\vec{v}\cdot\vec{u} \\ \vec{u}\cdot\vec{v} & -\vec{v}\cdot\vec{v} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_{i-1} \end{bmatrix} = \begin{bmatrix} (\mathbf{V}-\mathbf{U})\cdot\vec{u} \\ (\mathbf{V}-\mathbf{U})\cdot\vec{v} \end{bmatrix},$$

where

$$\begin{aligned} \vec{u} &= \mathbf{N}-\mathbf{U}, \\ \vec{v} &= \mathbf{W}-\mathbf{V}. \end{aligned}$$

The solution of the above system are the parameters $\alpha_i$ and $\alpha_{i-1}$, $0<\alpha_i,\alpha_{i-1}<1$. The texture coordinate of $\mathbf{A}_i$, denoted as $\mathbf{a}_i$, is derived by interpolating $\mathbf{u}$ and $\mathbf{n}$, while the texture coordinate of $\mathbf{A}_{i-1}$, denoted as $\mathbf{a}_{i-1}$, is derived by interpolating $\mathbf{v}$ and $\mathbf{w}$, as follows:

$$\begin{aligned} \mathbf{a}_i &= \mathbf{u} + \alpha_i(\mathbf{n}-\mathbf{u}), \\ \mathbf{a}_{i-1} &= \mathbf{v} + \alpha_{i-1}(\mathbf{w}-\mathbf{v}). \end{aligned} \tag{1}$$

It is apparent that $\mathbf{a}_i$ is generally not equal to $\mathbf{a}_{i-1}$. Note that $\mathbf{v}_j$ represents the texture coordinate of vertex $\mathbf{V}$ on $M^j$, for $j=n,\ldots,1,0$, and since every vertex $\mathbf{V}$ of $M$ remains fixed in the course of half-edge collapsing, $\mathbf{v}_j$ remains the same for all $\mathbf{V}$ of $M$. Since $\mathbf{U}$ is collapsed to $\mathbf{V}$, we need to find its texture coordinate $\mathbf{u}_{i-1}$ on $T^{i-1}$. First, we find the point $\mathbf{U}_{i-1}$ on $M_{i-1}$ that is nearest to $\mathbf{U}$ and identify the triangle containing $\mathbf{U}_{i-1}$. Then we compute the barycentric coordinates of $\mathbf{U}_{i-1}$ with respect to the triangle and finally derive the texture coordinate $\mathbf{u}_{i-1}$ from the texture coordinates of triangle vertices using the barycentric coordinates.

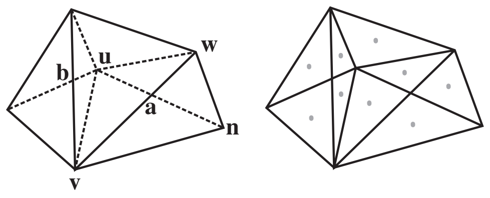

**Figure 5:** *Edges overlay in the one-ring neighborhood of u (left), and the edges partition the neighborhood into 9 cells (right).*

After deriving $\mathbf{a}_i$, $\mathbf{a}_{i-1}$, $\mathbf{b}_i$, $\mathbf{b}_{i-1}$, and $\mathbf{u}_{i-1}$, we move $\mathbf{a}$ to $\mathbf{a}_i$ and $\mathbf{b}$ to $\mathbf{b}_i$ to form the partition for $M^i$ and, similarly, move $\mathbf{a}$ to $\mathbf{a}_{i-1}$, $\mathbf{b}$ to $\mathbf{b}_{i-1}$, and $\mathbf{u}$ to $\mathbf{u}_{i-1}$ to form the partition for $M^{i-1}$. See Figure 6. Two cells in these two region partitions are said to be in correspondence if they correspond to the same cell before moving the points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{u}$ to designated texture coordinates. For example, $\triangle\mathbf{a}_i\mathbf{vn}$ and $\triangle\mathbf{a}_{i-1}\mathbf{vn}$ are in correspondence.

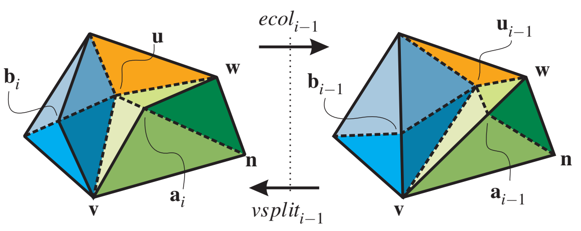

**Figure 6:** *Cell correspondence between partitions of $T^i$ (left) and $T^{i-1}$ (right).*

### 3.3. Texture adaptation

After cell correspondences for all cells in $R_i$ are found, the texture adaptation from $T^i$ to $T^{i-1}$ is performed for each pair of corresponding cells. To adapt the texture from cell $\triangle \mathbf{a}_i\mathbf{b}_i\mathbf{c}_i$ of $T^i$ to cell $\triangle \mathbf{a}_{i-1}\mathbf{b}_{i-1}\mathbf{c}_{i-1}$ of $T^{i-1}$, we let the former be the source texture, the latter be the target polygon, and then apply hardware texture mapping. Similarly, for the backward texture adaptation from $T^{i-1}$ to $T^i$, $\triangle \mathbf{a}_{i-1}\mathbf{b}_{i-1}\mathbf{c}_{i-1}$ is the source texture, and $\triangle \mathbf{a}_i\mathbf{b}_i\mathbf{c}_i$ is the target polygon. Care must be taken to prevent the so called *parametric folding* in the process of texture adaptation by ensuring that areas of both cells $\triangle \mathbf{a}_i\mathbf{b}_i\mathbf{c}_i$ and $\triangle \mathbf{a}_{i-1}\mathbf{b}_{i-1}\mathbf{c}_{i-1}$ are positive. Note that, since we maintain all $T_i$, $i = n, \ldots, 1, 0$, in a single physical texture map, we need a temporary map to support the texture adaptation.

It is worth mentioning that the texture adaptation for all pairs of corresponding cells can be accelerated by using triangle-fan and triangle strip setups, as shown in Figure 7. It's cost can be further minimized by packing triangle fans and strips into arrays and using draw array commands, such as glDrawArrays() or glDrawElements(). Moreover, it can be run in parallel with the edge collapse operation.

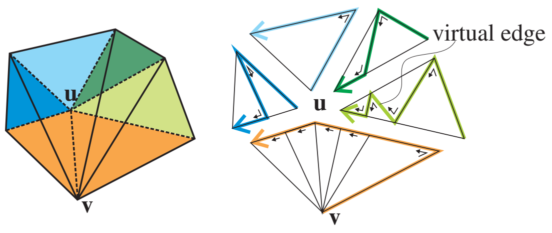

**Figure 7:** *Accelerate texture adaptation by triangle-fan and triangle strip setups.*

### 3.4. Indexing map

Texture adaptation is basically a re-sampling process of a texture map. When a texture area of higher frequency gets adapted to a smaller texture area, blurred artifacts due to under-sampling of texture samples might appear, as shown in Figure 8. To prevent such problems, we propose the mechanism of *indexing mapping* that uses an indexing map $I$ to store in each texel the texture coordinate referring to the original texture map $T$. All the texture adaptation operations are applied to the indexing map, leaving the original texture map alone. Initially, the indexing map $I^n$ for $M^n$ stores in each texel the coordinate itself, that is, $I^n(x,y) = (\frac{x+0.5}{w}, \frac{y+0.5}{h})$, where $w \times h$ is the resolution of the indexing map. For texture adaptation, we derive $I^{i-1}$ from $I^i$ in the same way as we do for deriving $T^{i-1}$ from $T^i$, for $i = n, \ldots, 2, 1$.

Figure 9 illustrates an example of indexing mapping. The surface $S$ is texture mapped with $T$. When $S$ is simplified to $S'$, $T$ is adapted to $T'$ and, in this example, the texels on $T$ are re-sampled to two texels on $T'$. As a result, one gets a blurred textured image while texture mapping $T'$ onto $S'$. With the proposed indexing mapping, the texel values of $I'$ are the coordinates of $T$, and are used to access to the original texels on $T$. Therefore, blurred artifacts are reduced.

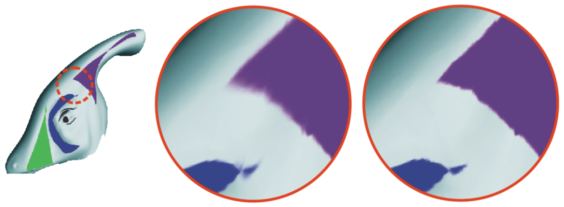

**Figure 8:** *Blurred artifacts introduced by texture adaptation (left) and minimized by indexing mapping (right).*

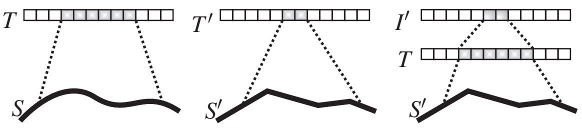

**Figure 9:** *Indexing mapping.*

The use of indexing map is also advantageous when more than one texture map, such as a combination of a color map, a normal map, and a bump map, are associated with the model since texture adaptation is applied to the indexing map only, leaving all the maps untouched.

Indirect accessing of a texture map is not supported by graphics APIs such as OpenGL. Fortunately, it can be implemented using the fragment shader. Another issue is the precision of the indexing map. In our experience, a texture map of 16-bit precision is sufficient to deliver acceptable image quality and performance for texture adaptation. See the example shown in Figure 10.

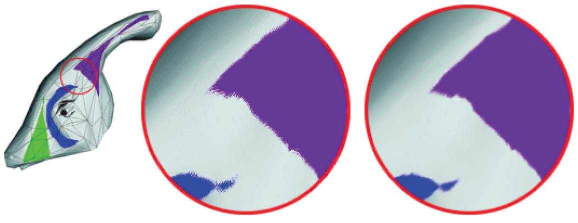

**Figure 10:** *Performance difference of indexing map in* 8*-bit (center) and* 16*-bit (left) precision.*

## 4. Results

In our experiments, we used a P4 3.0Ghz platform with an *n*VIDIA GeForce 6800 GT graphics card. All test models are using a $1024 \times 1024$ 16-bit floating indexing map.

We use the render-to-texture feature to avoid transferring texture maps between graphics hardware and the host CPU. In our experiments, we observed that the render-to-texture setup is the most expensive operation in texture adaptation even with the help of framebuffer objects (FBOs). To minimize the setup cost, we simply collect texture adaptations that are applied to disjoint regions and perform adaptations together. When we simplify the model from $M^j$ to $M^i$, where $j > i$, we start with the set COLLECTION containing only the texture adaptation from $T^j$ to $T^{j-1}$ and the set $R = \{R_j\}$. We then check, in the sequence of $k = j-1, \dots, i+1$, to see if $R_k$ overlays with any element of $R$. If not, $R_k$ is inserted into $R$ and the texture adaptation from $T^k$ to $T^{k-1}$ is put into COLLECTION. After doing this cycle, we perform all the texture adaptations collected in COLLECTION with one common setup. We repeatedly perform the packing of texture adaptations until all the texture adaptations are done. A similar approach is applied to the refinement process. We have observed that 13.73 and 20.20 texture adaptations on average are packed to share a common setup for the parasaur head and bunny head, respectively, and such a simple approach achieves about a 13.66× speedup factor for the parasaur head model and 17.51× for bunny head model, as depicted in Table 1, and the entire run-time cost is also listed in Table 1. Moreover, the setup can be further shared among objects in an environment with multiple objects.

Experimental tests have been done on several models. For PM construction, a quadric error metric (QEM) of five dimensions [GH98] is applied to all tested models. Figure 11 demonstrates the power of texture adaptation on the parasaur head at bottom row. A comparison with QEM of 5D without texture adaptation and APS [COM98] without texture adaptation is shown at top and middle row, respectively. The texture map for the parasaur head is derived using signal-specialized parameterization [SGSH02]. Texture distortion becomes noticeable when the mesh is simplified to 2000 polygons and becomes obvious for meshes of 1000 and 499 polygons. We have observed that texture distortion is almost eliminated by texture adaptation even in the textured image of the simplified model with 499 polygons. A normal mapped parasaur head is also greatly improved by texture adaptation in Figure 12.

The horse model shown in Figures 13 is parameterized by geometric-stretch-minimizing parameterization [SSGH01]. Texture distortion is perceivable in the simplified meshes and is almost eliminated by using texture adaptation.

Figure 14 depicts the performance of texture adaptation on a bunny head model with color map parameterized by geometric-stretch-minimizing parameterization. The bottom row shows the parameterization and texture map. The texture distortion is visualized by texture mapping the check board onto the model. For the textured image without texture adaptation, texture distortion is apparent inside the enlarged image. On the other hand, texture distortion is not noticeable in the textured image with texture adaptation.

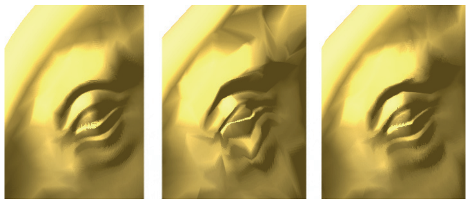
**Figure 12:** *Normal mapped parasaur head. left:* 7685 *polygons, center:* 499 *polygons, and right:* 499 *polygons with texture adaptation.*

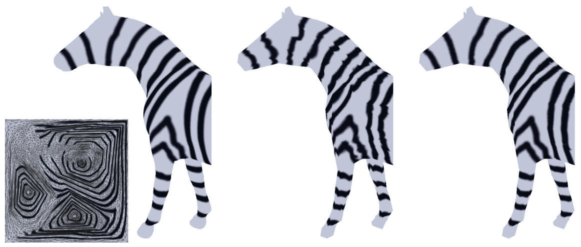
**Figure 13:** *Textured images of horse model. Left: original model of* 8160 *polygons with parameterized texture map, center: simplified model of* 800 *polygons without texture adaptation, right: simplified model of* 800 *polygons with texture adaptation.*

Our last tested model is a swirl model represented by swirled texture coordinates on a curved surface. Texture distortion is also perceived on the simplified swirl model. See Figure 15. The swirl lines get straightened on the simplified mesh but are well preserved by using texture adaptation.

## 5. Conclusion

We have presented a texture adaptation for progressive meshes, aiming to eliminate texture distortion introduced by edge collapses. The proposed scheme adapts the texture for each edge collapse, but only one single texture map is required for entire PM sequence since the actual texture adaptation operation is performed when simplifying or refining the model. The texture adaptation operation is local, incremental, invertible, and can be accelerated by graphics hardware. We have also proposed the mechanism of indexing mapping to reduce blurred artifacts due to under-sampling that might be introduced by texture adaptation. The experiment results have revealed that the texture adaptation scheme is capable of eliminating texture distortion in a very efficient

| | | | w/o *packing* | | w/ *packing* | | | | run-time cost (ms) | |
|---|---|---|---|---|---|---|---|---|---|---|
| model | original model (polygons) | simplified model (polygons) | # of texture adaptations | avg. time (ms) | # of packs | # of texture adaptations/pack | avg. time (ms) | speedup factor | w/ texture adaptation | w/o texture adaptation |
| parasaur head | 7,835 | 499 | 3,611 | 0.7581 | 263 | 13.73 | 0.0555 | 13.66× | 253.42 | 52.99 |
| bunny head | 17,483 | 500 | 8,545 | 0.7616 | 423 | 20.20 | 0.0435 | 17.51× | 508.87 | 137.16 |

**Table 1:** *Speedup by packing texture adaptations, and the entire run-time cost of simplifying PM w/ and w/o texture adaption.*

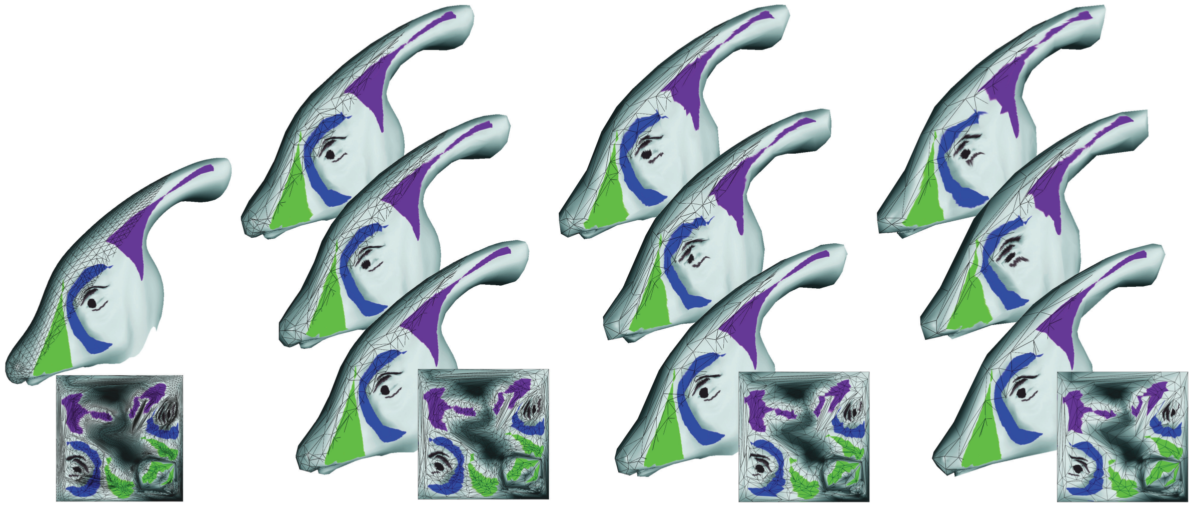
(a) *Original (7685 polygons)* (b) 2000 *polygons* (c) 1000 *polygons* (d) 499 *polygons*

**Figure 11:** *Textured images of parasaur head model. Top row of (b),(c),(d): simplified by QEM of* 5D *without texture adaptation, middle row: by APS without texture adaptation, bottom row: by QEM of* 5D *with texture adaptation.*

manner. As a future work, one may investigate how to design a meaningful error metric for PM construction in the presence of the proposed texture adaptation.

**Acknowledgments**

The authors would like to thank Pedro V. Sander for providing the parasaur head model and texture map, T.-C. Ho, J.-B. Yu, and Y.-C. Chiu for preparing the test models, Y.-G. Lin for creating the animation, and Roger Hong for improving the writing. Finally, we also thank the anonymous reviewers for their suggestions toward improving this paper.

## References

[BS96] BAJAJ C., SCHIKORE D.: Error-Bounded Reduction of Triangle Meshes with Multivariate Data. *SPIE 2656* (1996), 34–45.

[CMR*99] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R., TARINI M.: Preserving Attribute Values on Simplified Meshes by Resampling Detail Textures. *The Visual Computer 15*, 10 (1999), 519–539.

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-Preserving Simplification. In *Proc. SIGGRAPH '98* (New York, NY, USA, 1998), ACM Press, pp. 115–122.

[ESG01] ECKSTEIN I., SURAZHSKY V., GOTSMAN C.: Texture Mapping with Hard Constraints. *Computer Graphics Forum 20*, 3 (2001), 95–104.

[GH98] GARLAND M., HECKBERT P. S.: Simplifying Surfaces with Color and Texture Using Quadric Error Metrics. In *Proceedings of IEEE Visualization '98* (Oct. 1998), pp. 263–269.

[Hop96] HOPPE H.: Progressive Meshes. In *Proc. SIGGRAPH '96* (Aug. 1996), Rushmeier H., (Ed.), pp. 99–108.

[KLS03] KHODAKOVSKY A., LITKE N., SCHRÖDER P.: Globally Smooth Parameterizations with Low Distortion. *Proc. SIGGRAPH 2003, ACM Trans. on Graphics 22*, 3 (2003), 350–357.

[KW01] KIM H., WOHN K.: Multiresolution Model Generation with Geometry and Texture. In *Seventh Inter-*

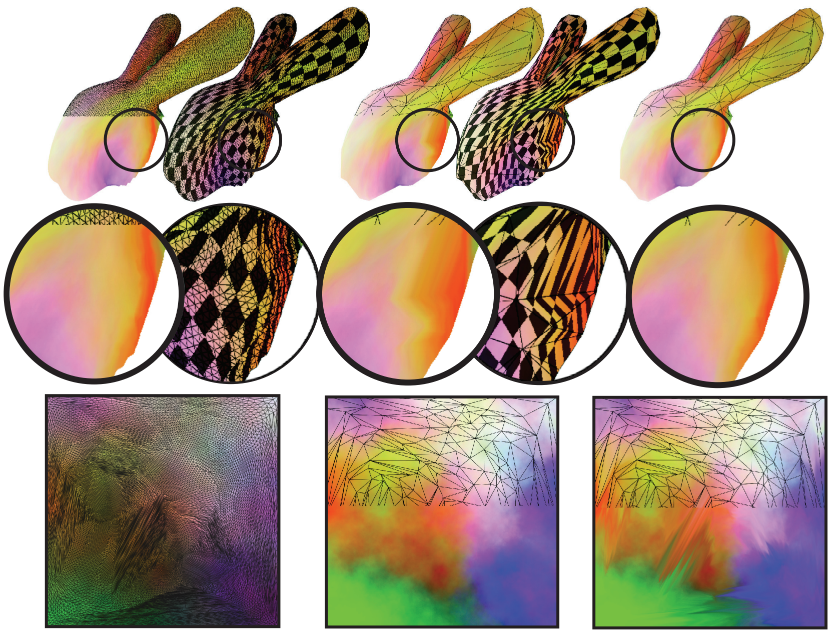

**Figure 14:** *Left: original model of* 17483 *polygons, center: simplified model of* 500 *polygons without texture adaptation, right: simplified model of* 500 *polygons with texture adaptation.*

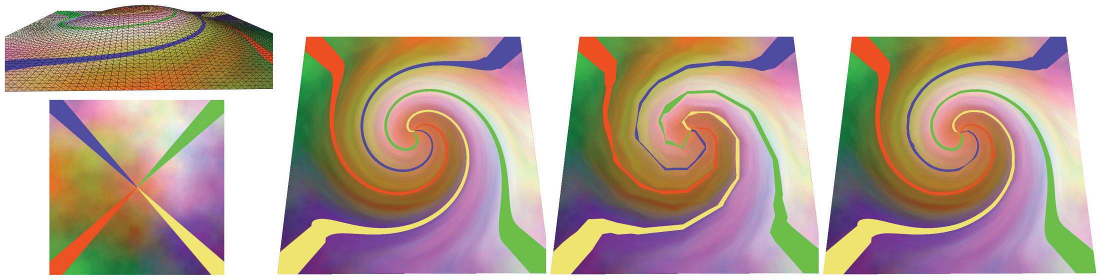

**Figure 15:** *From left to right, mesh with swirled texture coordinates and its color texture map, original mesh (*7688 *polygons), simplified mesh (*399 *polygons), and the same simplified mesh with texture adaptation.*

*national Conference on Virtual Systems and Multimedia (VSMM'01)* (Oct 2001), IEEE, pp. 780–789.

[LT00] LINDSTROM P., TURK G.: Image-Driven Simplification. *ACM Trans. on Graphics 19*, 3 (July 2000), 204–241.

[LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-Specialized Parametrization. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, 2002), Eurographics Association, pp. 87–98.

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture Mapping Progressive Meshes. In *Proc. SIGGRAPH 2001* (New York, NY, USA, 2001), ACM Press, pp. 409–416.

[XSX05] XU A., SUN S., XU K.: Texture Information Driven Triangle Mesh Simplification. In *Computer Graphics and Imaging - CGIM2005* (Honolulu, Hawaii, Aug 2005), Hamza M., (Ed.), ACTA Press, pp. 73–77.

[ZWT*05] ZHOU K., WANG X., TONG Y., DESBRUN M., GUO B., SHUM H.-Y.: TextureMontage: Seamless Texturing of Arbitrary Surfaces From Multiple Images. *Proc. SIGGRAPH 2005, ACM Trans. on Graphics 24*, 3 (2005), 1148–1155.