Near-Optimum Adaptive Tessellation of General Catmull-Clark Subdivision Surfaces

Shuhua Lai
University of Kentucky
shuhua@csr.uky.edu
http://www.csr.uky.edu/~shuhua/

Fuhua (Frank) Cheng
University of Kentucky
cheng@cs.uky.edu
http://www.cs.uky.edu/~cheng/

Abstract. A new adaptive tessellation method for general Catmull-Clark subdivision surfaces is presented. Development of the new method is based on the observation that *optimum adaptive tessellation* for rendering purpose is a *recursive error evaluation* and *globalization* process. The adaptive tessellation process is done by generating an inscribing polyhedron to approximate the limit surface for each individual patch. The inscribing polyhedron is generated through an adaptive subdivision on the patch's parameter space driven by a recursive error evaluation process. This approach generates less faces in the resulting approximating mesh while meeting the given precision requirement. The crack problem is avoided through globalization of new vertices generated in the adaptive subdivision process of the parameter space. No crack-detection or crack-elimination is needed in the adaptive tessellation process. Therefore, no mesh element splitting to eliminate cracks is necessary. The new adaptive tessellation method can precisely measure the error for every point of the limit surface. Hence, it has complete control of the accuracy of the tessellation result.

Keywords: subdivision, Catmull-Clark surfaces, adaptive tessellation, surface evaluation

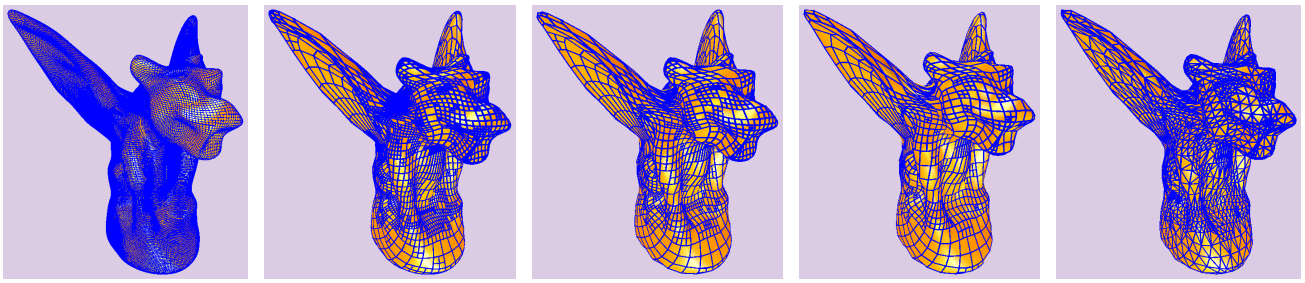
1 Introduction

Catmull-Clark subdivision scheme provides a powerful method for building smooth and complex surfaces. Given a control mesh, a *Catmull-Clark subdivision surface* (CCSS) is generated by iteratively refining (subdividing) the control mesh to form new and finer control meshes [2]. The mesh refining process consists of defining new vertices and connecting the new vertices to form new edges and faces of a new control mesh. A CCSS is the limit surface of the refined control meshes. The limit surface is called a *subdivision surface* because the mesh refining process is a generalization of the uniform B-spline surface *subdivision technique*. Subdivision surfaces can model/represent complex shape of arbitrary topology because there is no limit on the shape and topology of the control mesh of a subdivision surface [3]. But the number of faces in the uniformly refined meshes increases exponentially with respect to subdivision depth. Adaptive tessellation reduces the number of faces needed to yield a smooth approximation to the limit surface and, consequently, makes the rendering process more efficient. See Figure 1 for an example where the control mesh of a Gargoyle is uniformly refined only twice and yet the resulting mesh is already quite dense (Figure 1(a)), while the meshes generated by adaptively tessellating the same model 4, 3, and 2 times (Figure 1(b), 1(c), and 1(d), respectively) have a higher or similar precision but with much less faces.

1.1 Previous Work

A number of adaptive tessellation methods for subdivision surfaces have been proposed. Most of them are *mesh refinement based*, i.e., approximating the limit surface by adaptively refining the control mesh. This approach requires the assignment of a subdivision depth to each region of the surface first. In [20], a subdivision depth is calculated for each patch of the given CCSS with respect to a given error tolerance ϵ. In [9], a subdivision depth is estimated for each vertex of the given CCSS by considering factors such as curvature, visibility, membership to the silhouette, and projected size of the patch. The approach used in [20] is error controllable. An error controllable approach for Loop surface is proposed in [11], which calculates a subdivision depth for each patch of a Loop surface by estimating the distance between two bounding linear functions for each component of the 3D representation.

Several other adaptive tessellation schemes have



(a) Uniform (b) Adaptive (c) Adaptive (d) Adaptive (e) Triangulated

Figure 1: Adaptive tessellation of a surface with arbitrary topology.

been presented as well [15, 14, 10]. In [10], two methods of adaptive tessellation for triangular meshes are proposed. The adaptive tessellation process for each patch is based on angles between its normal and normals of adjacent faces. A set of new error metrics tailored to the particular needs of surfaces with sharp creases is introduced in [14].

In addition to various adaptive tessellation schemes, there are also applications of these techniques. D. Rose et al. used adaptive tessellation method to render terrain [18] and K. Müller et al. combined ray tracing with adaptive subdivision surfaces to generate some realistic scenes [13]. Adaptive tessellation is such an important technique that an API has been designed for its general usage [17]. Actually hardware implementation of this technique has been reported recently as well [12].

A problem with mesh-refinement-based, adaptive tessellation techniques is the possible *over-tessellation problem*. Each region, such as a patch, where a subdivision depth is assigned is uniformly subdivided to the level specified by the subdivision depth. Since the subdivision depth is computed based on the largest possible curvature of the region, parts of the region which do not carry such a large curvature will be unnecessarily subdivided.

Another problem is the so called *crack-prevention requirement*. Because the number of new vertices generated on the boundary of a region depends on the subdivision depth, *cracks* (or, *gaps*) would occur between adjacent regions if these regions are assigned different subdivision depths. Hence, such an adaptive tessellation method needs special mechanism to eliminate cracks. This is usually done by performing additional subdivision or splitting steps on the region with lower subdivision depth. As a result, many unnecessary mesh elements are generated.

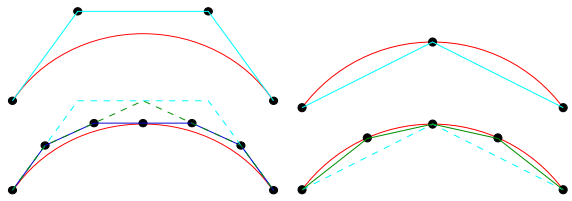
1.2 Overview

In this paper, we will present a new adaptive tessellation method to address the above two problems. The new method is developed for CCSS's. The possible *over-tessellation problem* is addressed by driving the tessellation process by a *recursive error evaluation process* within each patch of the surface and the *crack prevention requirement* is addressed by using a *globalization technique* in the subdivision process to avoid the need of *crack-detection* and *crack-elimination*. To ensure the *over-tessellation problem* is completely addressed, we present an error evaluation process that can precisely measure the error for every point of the limit surface so that accuracy of the tessellation result can be completely controlled. Test results show that, with these new techniques, number of faces in the resulting approximating mesh is significantly reduced and, consequently, rendering of CCSS's can be made much more efficient.

The remaining part of the paper is arranged as follows. In Section 2, motivation and basic idea of the new method are presented. The evaluation techniques for a CCSS are shown in Section 3. The error evaluation process used in the adaptive tessellation process is shown in Section 4. The globalization technique is shown in Section 5. A discussion of *absolute flatness* vs. *relative flatness* is given in Section 6. Implementation issues and test results are presented in Section 7. The concluding remarks are given in Section 8.

2 Basic Idea

Given the control mesh of a CCSS and an error tolerance, ϵ, the goal is to generate an approximating polyhedron mesh close enough to the limit surface $\mathbf{S}(u, v)$, i.e., within the error tolerance of $\mathbf{S}(u, v)$, but with as



(a) Circumscribed (mesh refinement base)

(b) Inscribed (parameter space refinement based)

Figure 2: Inscribed and Circumscribed Approximation.

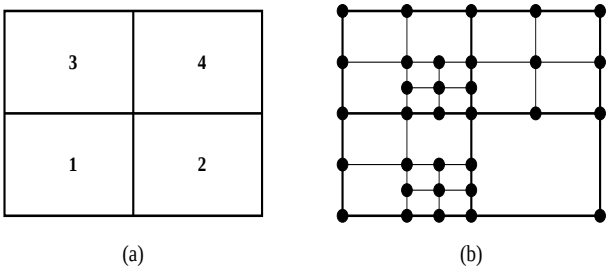


Figure 3: Adaptive subdivision on parameter spaces of patches.

few mesh faces as possible, so that the rendering process of $\mathbf{S}(u, v)$ can be performed efficiently. An approximating polyhedron mesh with the least number of mesh faces is called an *optimum approximating polyhedron mesh*. We assume each face of the control mesh is a quadrilateral and each face has at most one extraordinary vertex (a vertex with a *valence* different from 4). If this is not the case, simply perform Catmull-Clark subdivision on the control mesh of the CCSS twice.

Our first goal is to avoid the possible *over-tessellation problem* discussed in Section 1.1. It is easy to see that, to achieve such a goal, tessellation process within each patch should also be performed based on the flatness of each local region. This can be accomplished by doing *adaptive subdivision* on the parameter space of each patch that is driven by a *recursive error evaluation process*. Contrary to the *mesh refinement based approaches* which generate approximating polyhedra from "outside" the limit surface that usually do not interpolate the limit surface (see Figure 2(a) for an example in the curve case), the approximating polyhedron generated by this approach is an *inscribing polyhedron* whose vertices interpolate the limit surface (see Figure 2(b) for an example in the curve case). The process is illustrated below.

For a patch of $\mathbf{S}(u, v)$ defined on $[u_1, u_2]\times[v_1, v_2]$, we approximate it with the *base quadrilateral* formed by its four vertices $\mathbf{V}_1 = \mathbf{S}(u_1, v_1)$, $\mathbf{V}_2 = \mathbf{S}(u_2, v_1)$, $\mathbf{V}_3 = \mathbf{S}(u_2, v_2)$ and $\mathbf{V}_4 = \mathbf{S}(u_1, v_2)$. If the *distance* (error) (to be defined below) between the patch and its base quadrilateral is small than ϵ, the patch is considered *flat* enough and is replaced with the base quadrilateral in the tessellation process. Otherwise, we perform a *midpoint subdivision* on the parameter space by setting

$$u_{12} = \frac{u_1 + u_2}{2} \quad \text{and} \quad v_{12} = \frac{v_1 + v_2}{2}$$

to get four subpatches: $[u_1, u_{12}] \times [v_1, v_{12}]$, $[u_{12}, u_2] \times [v_1, v_{12}]$, $[u_{12}, u_2] \times [v_{12}, v_2]$, $[u_1, u_{12}] \times [v_{12}, v_2]$, and repeat the *flatness testing* (error evaluation) process on each of the subpatches. The process is recursively repeated until the distances (errors) between all the subpatches and their corresponding base quadrilaterals are smaller than ϵ. The vertices of the resulting subpatches are then used as vertices of the inscribing polyhedron that approximates the limit surface. For example, if the four rectangles in Figure 3(a) are the parameter spaces of four adjacent patches of $\mathbf{S}(u, v)$, and if the rectangles shown in Figure 3(b) are the parameter spaces of the resulting subpatches when the above *recursive flatness testing (error evaluation) process* stops, then the limit surface will be *evaluated* at the points marked with small solid circles to form vertices of an inscribing approximating polyhedron of the limit surface.

This is a simple and straightforward process, it is by no means new, but the result could be very significant. Note that each face in the inscribed approximating polyhedron for a patch is built with the expectation that it is just close enough to the limit surface but with the maximum possible size. Therefore, if the recursive error evaluation process can indeed provide precise error estimate, then the approximating polyhedron mesh generated by this process is *optimum* or *near-optimum* (in case some faces from different sides of a common boundary of two patches can be merged into a bigger face with the same error size). So, the point now is, can we precisely evaluate the error for any part of a CCSS? To ensure that the approximating polyhedron mesh is precisely constructed, we must also be able to precisely evaluate a CCSS at any given parameter point. It is known how to do these tasks for regular patches of a CCSS. We will show that these tasks are possible for extra-ordinary patches of a CCSS as well in Sections 3 and 4.

Our second goal is to avoid the *crack prevention requirement* discussed in Section 1.1. Due to the fact that adjacent patches are usually approximated by

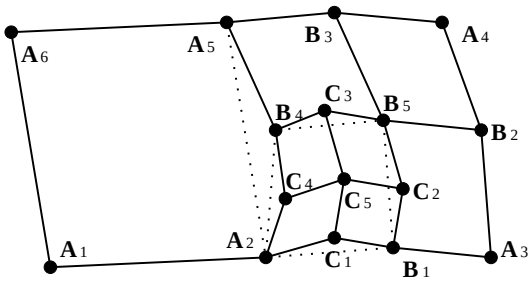


Figure 4: Cracks between adjacent patches (subpatches).

base quadrilaterals from different levels of the midpoint subdivision process, cracks could occur between adjacent patches. For instance, in Figure 4, the left patch is approximated by one base quadrilateral but the right patch is approximated by 7 base quadrilaterals. Consider the boundary shared by the left patch and the right patch. On the left side, that boundary is approximated by a line segment defined by two vertices, $\mathbf{A}_2$ and $\mathbf{A}_5$. But on the right side, the boundary is approximated by a polyline defined by four vertices, $\mathbf{A}_2$, $\mathbf{C}_4$, $\mathbf{B}_4$, and $\mathbf{A}_5$. They do not coincide unless $\mathbf{C}_4$ and $\mathbf{B}_4$ lie on the line segment defined by $\mathbf{A}_2$ and $\mathbf{A}_5$. But this usually is not the case. Hence, a crack would appear between the left patch and the right patch. The points shown in Figure 4 are points of the limit surface, not points in the parameter space of the limit surface.

Fortunately Cracks can be removed simply by replacing edges of the base quadrilaterals with appropriate polylines in the tessellation process. Namely, each edge of a base quadrilateral should be replaced with a polyline defined with all the new vertices computed for that edge of the corresponding patch or subpatch. For example, in Figure 4, all the dashed lines should be replaced with the corresponding polylines. In particular, edge $\mathbf{A}_2\mathbf{A}_5$ of the base quadrilateral $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_5\mathbf{A}_6$ should be replaced with the polyline $\mathbf{A}_2\mathbf{C}_4\mathbf{B}_4\mathbf{A}_5$. As a result, the left patch is approximated by the polygon $\mathbf{A}_1\mathbf{A}_2\mathbf{C}_4\mathbf{B}_4\mathbf{A}_5\mathbf{A}_6$, instead of the base quadrilateral $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_5\mathbf{A}_6$, in the tessellation process. For rendering purpose this is fine because graphics systems like OpenGL can handle polygons with any number of vertices and the vertices do not have to be co-planar. Note that, with the above approach, there is no need to perform *crack detection* at all because the resulting approximating polyhedron contains on cracks. Besides, since this process does not increase the number of faces in an approximating polyhedron, the resulting approximating polyhedron is *optimum* or *near-optimum* for the entire CCSS.

The point here is, how do we know which polyline should be used to replace an edge of a base quadrilateral? Currently, all the subdivision surface parametrization and evaluation techniques are patch based [4, 6, 7]. Hence, no matter which method is used in the tessellation process, a patch cannot see vertices evaluated by other patches from its own (local) structure even though the vertices are on its own boundary. For example, in Figure 4, vertices $\mathbf{C}_4$ and $\mathbf{B}_4$ are on the shared boundary of the left and the right patches. But the left patch can not see these vertices from its own structure because these vertices are not evaluated by this patch. So, the key here is how can one make adjacent patches visible to each other so that new vertices computed by one patch for the shared boundary can be accessed by the other patch. We will show in Section 5 that this is possible through a *globalization technique*. For convenience of subsequent reference, the process of replacing edges of base quadrilaterals with new polylines is called a *base quadrilateral replacement process*.

Note that in previous methods for adaptive tessellation of subdivision surfaces [20, 9, 10, 14], the most difficult part is crack prevention. With the above approach, this part becomes the simplest part to handle and implement.

3 Evaluation of a CCSS Patch

In this section we show how to evaluate an extra-ordinary CCSS patch and its tangents at a given point of the parameter space. These techniques are needed in the construction of the approximating polyhedron and the error evaluation process. Several approaches [4, 5, 6, 7] have been presented for exact evaluation of an extra-ordinary patch at any parameter point (u, v). We use the parametrization technique presented in [7] here. This method is more efficient for both surface and tangent evaluation because it employs less eigen basis functions in its representation.

The parametrization technique presented in [7] works for general CCSS's, i.e., for a given vertex point $\mathbf{V}$, a new *vertex point* $\mathbf{V}'$ is computed as:

$$\mathbf{V}' = \alpha_n \mathbf{V} + \beta_n \sum_{i=1}^{n} \mathbf{E}_i + \gamma_n \sum_{i=1}^{n} \mathbf{F}_i$$

where α_n, β_n and γ_n are positive numbers and $\alpha_n + \beta_n + \gamma_n = 1$, and it is based on an $\Omega - partition$ of the parameter space [4, 7]. The value of an extra-ordinary

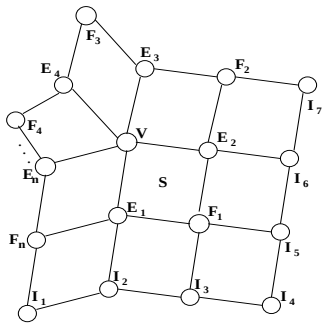


Figure 5: Control vertices of an extra-ordinary patch and their labeling.

patch is evaluated as follows:

$$\mathbf{S}(u,v) = W^T \mathrm{K}^m \sum_{j=0}^{n+5} \lambda_j^{m-1}\ \mathrm{M}_{b,j}\ G \qquad (1)$$

where n is the valance of the extra-ordinary patch [1], W is a vector containing the 16 B-spline power basis functions:

$$\begin{aligned} W^T(u,v) &= [1, u, v, u^2, uv, v^2, u^3, u^2v, uv^2, v^3, \\ &\qquad u^3v, u^2v^2, uv^3, u^3v^2, u^2v^3, u^3v^3]\ , \end{aligned}$$

K is a diagonal matrix:

$$\mathrm{K} = Diag(1,2,2,4,4,4,8,8,8,8,16,16,16,32,32,64),$$

and m and b are defined as follows:

$$m(u,v) = min\{\lceil log_{\frac{1}{2}} u \rceil, \lceil log_{\frac{1}{2}} v \rceil\}\ ,$$

$$b(u,v) = \begin{cases} 1, & if\ \ 2^m u \geq 1\ \ and\ \ 2^m v < 1 \\ 2, & if\ \ 2^m u \geq 1\ \ and\ \ 2^m v \geq 1 \\ 3, & if\ \ 2^m u < 1\ \ and\ \ 2^m v \geq 1\ . \end{cases}$$

λ_j, $0 \leq j \leq n+5$, are eigenvalues of the Catmull-Clark subdivision metrix and $\mathrm{M}_{b,j}$, $1 \leq b \leq 3$, $0 \leq j \leq n+5$, are matrices of dimension $16 \times (2n+8)$. λ_j and $\mathrm{M}_{b,j}$ are independent of (u,v) and their exact expressions are given in [7]. G is the vector of control points (See Fig. 5 for their labeling):

$$G = [\mathbf{V}, \mathbf{E}_1, \cdots, \mathbf{E}_n, \mathbf{F}_1, \cdots, \mathbf{F}_n, \mathbf{I}_1, \cdots, \mathbf{I}_7].$$

One can compute the derivatives of $\mathbf{S}(u,v)$ to any degree simply by differentiating $W(u,v)$ in Eq. (1) accordingly. For example,

$$\frac{\partial}{\partial u}\mathbf{S}(u,v) = (\frac{\partial W}{\partial u})^T\ \mathrm{K}^m \sum_{j=0}^{n+5} \lambda_j^{m-1}\ \mathrm{M}_{b,j}\ G. \qquad (2)$$

[1] Eq. (1) works for regular patches as well, i.e., when $n = 4$.

The value and tangents at an extra-ordinary vertex are simply the limit points of the corresponding equations in (2) when $(u,v) \to (0,0)$:

$$\begin{aligned} \mathbf{S}(0,0) &= [1,0,\cdots,0] \cdot M_{2,n+1} \cdot G \\ \mathbf{D}_u &= [0,1,0,0,\cdots,0] \cdot \mathrm{M}_{2,2} \cdot G \\ \mathbf{D}_v &= [0,0,1,0,\cdots,0] \cdot \mathrm{M}_{2,2} \cdot G \end{aligned} \qquad (3)$$

where $\mathbf{D}_u$ and $\mathbf{D}_v$ are the direction vectors of $\frac{\partial \mathbf{S}(0,0)}{\partial u}$ and $\frac{\partial \mathbf{S}(0,0)}{\partial v}$, respectively.

4 Flatness Testing (Error Evaluation)

In the *flatness testing process*, to measure the difference between a patch (or subpatch) and its *base quadrilateral*, we need to parametrize the base quadrilateral as well. The base quadrilateral can be parametrized with a simple bilinear interpolation:

$$\begin{aligned} \mathbf{Q}(u,v) &= \tfrac{v_2-v}{v_2-v_1}(\tfrac{u_2-u}{u_2-u_1}\mathbf{V}_1 + \tfrac{u-u_1}{u_2-u_1}\mathbf{V}_2) \\ &\quad + \tfrac{v-v_1}{v_2-v_1}(\tfrac{u_2-u}{u_2-u_1}\mathbf{V}_4 + \tfrac{u-u_1}{u_2-u_1}\mathbf{V}_3) \end{aligned} \qquad (4)$$

where $u_1 \leq u \leq u_2$, $v_1 \leq v \leq v_2$. The *difference* between the patch (or subpatch) and the base quadrilateral at (u,v) is defined as

$$\begin{aligned} d(u,v) &= \parallel \mathbf{Q}(u,v) - \mathbf{S}(u,v) \parallel^2 \\ &= (\mathbf{Q}(u,v) - \mathbf{S}(u,v)) \cdot (\mathbf{Q}(u,v) - \mathbf{S}(u,v))^T \end{aligned} \qquad (5)$$

where $\parallel \cdot \parallel$ is the second norm and $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. The *distance* between the patch (or subpatch) and the base quadrilateral is the maximum of all the differences:

$$D = \max\{\ \sqrt{d(u,v)}\ \mid\ (u,v) \in [u_1,u_2] \times [v_1,v_2]\ \}.$$

To measure the distance between a patch (or subpatch) and the corresponding base quadrilateral, we only need to measure the norms of all local minima and maxima of $d(u,v)$. Note that $\mathbf{Q}(u,v)$ and $\mathbf{S}(u,v)$ are both C^1-continuous, and $d(\mathbf{V}_1)$, $d(\mathbf{V}_2)$, $d(\mathbf{V}_3)$ and $d(\mathbf{V}_4)$ are equal to 0. Therefore, by Mean Value Theorem, the local minima and maxima must lie either inside $[u_1,u_2] \times [v_1,v_2]$ or on the four boundary curves. In other words, they must satisfy at least one of the

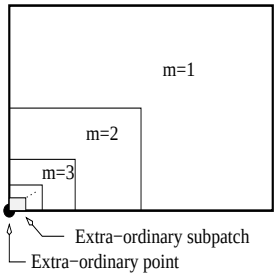


Figure 6: Partitioning of the unit square [7].

following three conditions:

$$\begin{cases} \frac{\partial d(u,v)}{\partial u} = 0 \\ v = v_1 \text{ or } v = v_2 \\ u_1 \le u \le u_2 \end{cases} \qquad \begin{cases} \frac{\partial d(u,v)}{\partial v} = 0 \\ u = u_1 \text{ or } u = u_2 \\ v_1 \le v \le v_2 \end{cases}$$

$$\begin{cases} \frac{\partial d(u,v)}{\partial u} = 0 \\ \frac{\partial d(u,v)}{\partial v} = 0 \\ (u,v) \in (u_1,u_2) \times (v_1,v_2) \end{cases} \tag{6}$$

For a patch (or subpatch) that is not adjacent to an extraordinary point (i.e., $(u_1, v_1) \neq (0,0)$), m is fixed and known ($m(u,v) = min\{\lceil log_{\frac{1}{2}} u \rceil, \lceil log_{\frac{1}{2}} v \rceil\}$). Hence Eq. (6) can be solved explicitly. With the valid solutions, we can find the difference for each of them using Eq. (5). Suppose the one with the biggest difference is $(\hat{u}, \hat{v})$. Then $(\hat{u}, \hat{v})$ is also the point with the biggest distance between the patch (or subpatch) and the corresponding base quadrilateral. The patch (or subpatch) is considered *flat* enough if

$$\sqrt{d\ (\ \hat{u},\ \hat{v})} \quad \le \quad \epsilon \tag{7}$$

where ϵ is a given error tolerance. In such a case, the patch (or subpatch) is replaced with the corresponding base quadrilateral in the tessellation process.

For a patch (or subpatch) that is adjacent to an extraordinary point (i.e. $(u_1, v_1) = (0,0)$ in Eq. (6)), m is not fixed and m tends to ∞ (see Figure 6). As a result, Eq. (6) can not be solved explicitly. One way to resolve this problem is to use nonlinear numerical method to solve these equations. But numerical approach cannot guarantee the error is less than ϵ everywhere. For precise error control, a better choice is needed. In the following, an alternative method is given for that purpose.

Eq. (3) shows that $\mathbf{S}(u,v)$ and $\mathbf{Q}(u,v)$ both converge to $\mathbf{S}(0,0)$ when $(u,v) \to (0,0)$. Hence, for any given error tolerance ϵ, there exists an integer m_ϵ such that if $m \ge m_\epsilon$, then the distance between $\mathbf{S}(u,v)$ and $\mathbf{S}(0,0)$ is smaller than $\epsilon/2$ for any $(u,v) \in [0, 1/2^m] \times [0, 1/2^m]$, and so is the distance between $\mathbf{Q}(u,v)$ and $\mathbf{S}(0,0)$. Consequently, when $(u,v) \in [0, 1/2^m] \times [0, 1/2^m]$, the distance between $\mathbf{S}(u,v)$ and $\mathbf{Q}(u,v)$ is smaller than ϵ. The value of m_ϵ, in most of the cases, is a relatively small number and can be explicitly calculated. In next subsection, we will show how to calculate m_ϵ.

For other regions of the unit square with $\lceil \log_{\frac{1}{2}} u_2 \rceil \le m < m_\epsilon$ (see Figure 6), eq. (6) can be used directly to find the difference between $\mathbf{S}(u,v)$ and $\mathbf{Q}(u,v)$ for any fixed $m \in (\lceil \log_{\frac{1}{2}} u_2 \rceil, m_\epsilon)$. Therefore, by combining all these differences, we have the distance between the given extra-ordinary patch (or subpatch) and the corresponding base quadrilateral. If this distance is smaller than ϵ, we consider the given extra-ordinary patch (or subpatch) to be flat, and use the base quadrilateral to replace the extra-ordinary patch (or subpatch) in the tessellation process. Otherwise, repeatedly subdivide the patch (or subpatch) and perform flatness testing on the resulting subpatches until all the subpatches satisfy Eq. (7).

4.1 Calculating m_ϵ

For a given $\epsilon > 0$, an integer k_ϵ will first be computed so that if k is bigger than k_ϵ, then the subpatch of $\mathbf{S}(u,v)$ with $0 \le u, v \le 1/2^k$ is contained in a sphere with center $\mathbf{S}(0,0)$ and diameter ϵ (called an ϵ-sphere). A subpatch is contained in an ϵ-sphere if all points of the subpatch are $\epsilon/2$ away from $\mathbf{S}(0,0)$.

To find such k_ϵ, we need a few properties from [7]. Recall that an extra-ordinary patch $\mathbf{S}(u,v)$ can be expressed as

$$\mathbf{S}(u,v) = \sum_{j=0}^{n+5} \Phi_{b,j}(u,v) \cdot G$$

where $\Phi_{b,j}$ are *eigen basis functions* defined in [7] and G is the vector of control points of $\mathbf{S}$. The eigen basis functions satisfy the *scaling relation* [4, 7], i.e.,

$$\Phi_{b,j}(u/2^k, v/2^k) = \lambda_j^k \Phi_{b,j}(u,v)$$

for any positive integer k, where λ_j are eigen values of the Catmull-Clark subdivision matrix [7]. The eigen values are indexed so that

$$1 = \lambda_{n+1} > \lambda_2 \ge \lambda_i > 0$$

where $0 \le i \le n+5$ and $i \neq n+1$. Also recall that $\Phi_{b,j}(0,0) = 0$ when $j \neq n+1$, and $\Phi_{b,n+1}(u,v)$ is a constant vector, its value is independent of (u,v) [7]. Hence,

$$(\Phi_{b,n+1}(u,v) - \Phi_{b,n+1}(u',v')) \cdot G_r = 0$$

for any (u,v) and (u',v') where $r \in \{x,y,z\}$ and G_r is the x-, y- or z-component of G.

Hence for any $1/2 \le u \le 1$ or $1/2 \le v \le 1$, and for any k we have

$$\begin{aligned}
&|\mathbf{S}_r(u/2^k, v/2^k) - \mathbf{S}_r(0,0)| \\
&= |\textstyle\sum_{j=0}^{n+5}(\lambda_j^k \Phi_{b,j}(u,v) - \Phi_{b,j}(0,0)) \cdot G_r| \\
&\le \textstyle\sum_{j\neq n+1} \lambda_j^k |(\Phi_{b,j}(u,v) \cdot G_r| \\
&< \lambda_2^k \textstyle\sum_{j\neq n+1} |(\Phi_{b,j}(u,v) \cdot G_r|
\end{aligned}$$

Similarly, the three conditions in Eqs. (6) can be used to find the maxima of $|(\Phi_{b,j}(u,v) \cdot G_r|$ for any j. Note that because here $(u,v) \notin [0,1/2] \times [0,1/2]$, the corresponding m is equal to 1 (See figure 6). Hence we can easily find the maximum in its domain $\{(u,v) | 1/2 \le u \le 1 \ or \ 1/2 \le v \le 1\}$. Let the maximum of $|(\Phi_{b,j}(u,v) \cdot G_r|$ be F_{rj} and $F_r = \sum_{j\neq n+1} F_{rj}$. Then, for any $k > 0$ we have

$$|\mathbf{S}_r(u/2^k, v/2^k) - \mathbf{S}_r(0,0)| \le \lambda_2^k F_r.$$

Therefore if $(\lambda_2^k F_x)^2 + (\lambda_2^k F_y)^2 + (\lambda_2^k F_z)^2 \le (\epsilon/2)^2$, we have

$$\| \mathbf{S}(u/2^k, v/2^k) - \mathbf{S}(0,0) \| \le \epsilon/2.$$

If we define k_ϵ as follows

$$k_\epsilon = \lceil \log_{\lambda_2} \frac{\epsilon}{2\sqrt{F_x^2 + F_y^2 + F_z^2}} \rceil$$

then it is easy to see that when $k \ge k_\epsilon$, the subpatch $\mathbf{S}(u,v)$ with $(u,v) \in [0, 1/2^k] \times [0, 1/2^k]$ is inside an ϵ-sphere whose center is $\mathbf{S}(0,0)$.

In addition, $\mathbf{S}(0,0)$ is a fixed point and can be explicitly evaluated for any patch (see eq. 3), and $\mathbf{Q}(u,v)$ also has an explicit parametrization (See eq. (4)). Hence, similarly, by using the method of Eqs. (6), it is easy to find an integer $\widetilde{k}_\epsilon$, such that for any given $\epsilon > 0$, when $k \ge \widetilde{k}_\epsilon$, we have $\| \mathbf{Q}(u,v) - \mathbf{S}(0,0) \| \le \epsilon/2$, where $(u,v) \in [0, 1/2^k] \times [0, 1/2^k]$. Once we have k_ϵ and $\widetilde{k}_\epsilon$, simply set m_ϵ as the maximum of k_ϵ and $\widetilde{k}_\epsilon$.

$$m_\epsilon = \max\{k_\epsilon, \widetilde{k}_\epsilon\}$$

With this m_ϵ, it is easy to see that when $m \ge m_\epsilon$, we have $\| \mathbf{S}(u,v) - \mathbf{Q}(u,v) \| \le \epsilon$, where $(u,v) \in [0, 1/2^m] \times [0, 1/2^m]$.

4.2 Flatness Testing Revisited

A potential problem with the *base quadrilateral replacement process* is the new polygon that replaces the base quadrilateral might not satisfy the flatness requirement. To ensure the flatness requirement is also satisfied by the polygons that replace the base quadrilaterals, we need to change the test condition in Eq. (7) to the following one:

$$\sqrt{d\ (\ \bar{u},\ \bar{v})\ } + \sqrt{d\ (\ \hat{u},\ \hat{v})\ } \le \epsilon \qquad (8)$$

where $(\hat{u}, \hat{v})$ and $(\bar{u}, \bar{v})$ are solutions of Eq. (6) and they satisfy the following conditions:

- Among all the solutions of Eq. (6) that are located on one side of $\mathbf{Q}(u,v)$, i.e. solutions that satisfy $(\mathbf{Q} - \mathbf{S}) \cdot ((\mathbf{V}_1 - \mathbf{V}_3) \times (\mathbf{V}_2 - \mathbf{V}_4)) \ge 0$, $d(\hat{u}, \hat{v})$ is the biggest. If there does not exist any solution such that this condition holds, then $d(\hat{u}, \hat{v})$ is set to 0

- Among all the solutions of Eq. (6) that are located on the other side of $\mathbf{Q}(u,v)$, i.e. solutions that satisfy $(\mathbf{Q} - \mathbf{S}) \cdot ((\mathbf{V}_1 - \mathbf{V}_3) \times (\mathbf{V}_2 - \mathbf{V}_4)) < 0$, $d(\bar{u}, \bar{v})$ is the biggest. If there does not exist any solution such this condtion holds, then $d(\bar{u}, \bar{v})$ is set to 0

From the definition of $(\hat{u}, \hat{v})$ and $(\bar{u}, \bar{v})$, we can see that satisfying Eq. (8) means that the patch being tested is located between two quadrilaterals that are ϵ away.

Note that all the evaluated points lie on the limit surface. Hence, for instance, in Fig. 4, points $\mathbf{A}_2, \mathbf{C}_4, \mathbf{B}_4$ and $\mathbf{A}_5$ of patch $\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5$ are also points of patch $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_5\mathbf{A}_6$. With the new test condition in Eq. (8), we know that a patch or subpatch is flat enough if it is located between two quadrilaterals that are ϵ away. Because boundary points $\mathbf{A}_2, \mathbf{C}_4, \mathbf{B}_4$ and $\mathbf{A}_5$ are on the limit surface, they must be located between two quadrilaterals that are ϵ away. So is the polygon $\mathbf{A}_1\mathbf{A}_2\mathbf{C}_4\mathbf{B}_4\mathbf{A}_5\mathbf{A}_6$. Now the patch (or subpatch) and its approximating polygon are both located inside two quadrilaterals that are ϵ away. Hence the overall error between the patch (or subpatch) and its approximating polygon is guaranteed to be smaller than ϵ.

5 Making Patches Visible to Each Other

In this section, we show how to use a *globalization process* to make adjacent patches visible to each other.

To make adjacent patches visible to each other and to make subsequent *base quadrilateral replacement process* easier, one should assign a *global index ID* to each evaluated vertex so that

- all the evaluated vertices with the same 3D position have the same index ID;

- the index ID's are sorted in v and then in u, i.e., if $(u_i, v_i) \geq (u_j, v_j)$, then $ID_i \geq ID_j$, unless ID_i or ID_j has been used in previous patch evaluation.

This global indexing technique allows subsequent processing to be performed on individual patches but still with a global visibility. We also need a step called *adaptive marking* to facilitate the *base quadrilateral replacement process*. The purpose of *adaptive marking* is to mark those points in uv space where the limit surface should be evaluated. With the help of the global index ID, this step can be done on an individual patch basis. Initially, all (u, v) points are marked white. If surface evaluation should be performed at a point and the resulting vertex is needed in the tessellation process, then that point is marked in black. This process can be easily implemented as a recursive function. A pseudo code for this step is given below.

```
AdaptiveMarking(P, u1, u2, v1, v2)
1.   Evaluate(P, u1, u2, v1, v2),
2.   AssignGlobalID(P, u1, u2, v1, v2),
3.   if (FlatEnough(P, u1, u2, v1, v2))
4.       MarkBlack(P, u1, u2, v1, v2)
5.   else
6.       u12 = (u1 + u2)/2
7.       v12 = (v1 + v2)/2
8.       AdaptiveMarking(P, u1, u12, v1, v12)
9.       AdaptiveMarking(P, u12, u2, v1, v12)
10.      AdaptiveMarking(P, u12, u2, v12, v2)
11.      AdaptiveMarking(P, u1, u12, v12, v2)
```

This routine adaptively marks points in the parameter space of patch **P**. Function 'Evaluate' evaluates limit surface at the four corners of patch or subpatch **P** defined on $[u_1, u_2] \times [v_1, v_2]$. Function 'AssignGlobeID' assignes global index ID to the four corners of **P**. Function 'FlatEnough' uses the method given in Section 4 and Eq. (7) to tell if a patch or subpatch is flat enough. Function 'MarkBlack' marks the four corners of patch or subpatch **P** defined on $[u_1, u_2] \times [v_1, v_2]$ in black. All the marked corner points will be used in the tessellation process. When a subpatch is ready for the *base quadrilateral replacement process*, simply output in order all the marked points between corners of the base quadrilateral to form the polygon that should be used for this base quadrilateral in the tessellation process.

6 Degree of Flatness

Just like numerical errors have two different settings, the flatness of a patch, which can be viewed as a numerical error from the approximation point of view, has two different aspects as well, depending on if the flatness is considered in the absolute sense or relative sense. The flatness of a patch is called the *absolute flatness* (AF) if the patch is not transformed in any way. In that case, the value of ϵ in Eqs. (7) and (8) is set to whatever precision the flatness of the patch is supposed to meet. AF should be considered for operations that work on physical size of an object such as machining or prototyping.

For operations that do not work on the physical size of an object, such as the rendering process, we need a flatness that does not depends on the physical size of a patch. Such a flatness must be Affine transformation invariant to be a constant for any transformed version of the patch. Such a flatness is called the *relative flatness* of the patch. More specifically, if **Q** is the base quadrilateral of patch **S**, the *relative flatness* (RF) of **S** with respect to **Q** is defined as follows:

$$\mathrm{RF} = \frac{d}{\max\{D_1, D_2\}}$$

where d is the maximal distance between **S** and **Q**, and D_1, D_2 are lengths of the diagonal lines of **Q**. It is easy to see that RF defined this way is Affine transformation invariant. Note that when D_1 and D_2 are fixed, smaller RE means smaller d. Hence, RE indeed measures the flatness of a patch. The difference between RF and AF is that RF measures the flatness of a patch in a global sense while AF measures flatness of a patch in a local sense. Therefore, RF is more suitable for operations that have data sets of various sizes but with a constant size display area such as the rendering process. Using RF is also good for adaptive tessellation process because it has the advantage of keeping the number of polygons low in the tessellation process.

7 Implementation and Test Results

The proposed approach has been implemented in *C++* using *OpenGL* as the supporting graphics system on the Windows platform. Some of the tested results are shown in Figures 1 and Figure 7. We also summarize those tested results in Table 1. The column underneath A|U|T in Table 1 indicates the type of tessellation technique (**A**daptive, **U**niform or **T**riangulated

after adaptive tessellation) used in the rendering process. For instance, Fig. 1(a) is generated using uniform subdivision, while Figs. 1(b), 1(c), 1(d) are tessellated with the adaptive technique presented in this paper, and Fig. 1(e) is the triangulated result of Fig. 1(d). Also Fig. 7(e) and Fig. 7(p) are the triangulated results of Fig. 7(d) and Fig. 7(o), respectively. The term *A/U ratio* means the ratio of number of polygons in an adaptively tessellated CCSS to its counter part in a uniformly tessellated CCSS with the same accuracy. The term *Depth* means the number of iterative uniform subdivisions that have to be performed on the control mesh of a CCSS to satisfy the error requirement. From Table 1 we can see that all the adaptively tessellated CCSS's have relatively low A/U ratios. This shows the proposed method indeed significantly reduces the number of faces in the resulting mesh while satisfying the given error requirement.

The 'Error' column in Table 1 represents absolute error. We can easily see that, for the same model, the smaller the error, the lower the A/U ratio. For example, Fig. 7(b) has lower A/U ratio than Fig. 7(c) and Fig. 7(d) because the former has smaller error tolerance than the last two. However, for the same model, if the difference of two error tolerances is not big enough, the resulting adaptive tessellation would have the same subdivision depth (see information on Figs. 7(g) and 7(h) or Figs. 7(l) and 7(m) in Table 1). As a result, the one with smaller error tolerance would have higher A/U ratio, because the corresponding uniformly subdivided meshes are the same. Another interesting fact is that Fig. 7(k) uses much more polygons than Fig. 7(l) does, while the former is less accurate than the latter. This shows the presented adaptive tessellation method is capable of providing a higher accuracy with less polygons.

From Table 1 we can easily see that for different models the absolute errors differ very much. Therefore, for different models, comparing their absolute errors might not make any practical sense because absolute error is not affine transformation invariant. In the mean while, from Table 1, we can see that RF is a much better and more understandable measurement for users to specify the error requirement in the adaptive tessellation process.

From Table 1, we can also see that triangulated tessellations usually have higher A/U ratio, because triangulation increases the number of polygons by at lease 2 times. Hence triangulation will slow down the rendering process while it does not improve accuracy. From the view point of rendering, triangulation is not really necessary. But for some special applications, such as Finite Element Analysis, triangulation is indispensable. Performing triangulation on the resulting mesh of our adaptive tessellation process is straightforward and fast.

The proposed adaptive tessellation method is good for models that have large flat or nearly flat regions in its limit surface and would save significant amount of time in the final rendering process, but may not have low A/U ratios when it is applied to surfaces with extraordinary curvature distribution or surfaces with very dense control meshes. One main disadvantage of all the current adaptive tessellation methods (including the method proposed here) is that they only eliminate polygons inside a patch. They do not take the whole surface into consideration. For instance, all the flat sides of the rocker arm model in Fig. 7 are already flat enough, yet a lot of polygons are still generated there.

8 Summary

A new adaptive tessellation method for general Catmull-Clark subdivision surfaces is presented. The new method is developed for rendering purpose and is based on the observation that *optimum adaptive tessellation* for rendering purpose is a *recursive error evaluation* and *globalization* process. The new method can precisely measure the error for every point of the limit surface and is performed on the basis of individual patches. Hence, the approximating polyhedron mesh generated by this method is optimum for each patch of the CCSS. Furthermore, since the new method does not increase the number of faces in order to avoid the generation of cracks, the resulting approximating polyhedron mesh is actually near-optimum for the entire CCSS. As the the new method does not require crack detection, it is also computation efficient.

The result of our work can be improved by running a post-processor to see if some faces from different sides of a patch boundary can be merged into a bigger face with the same error size. However, since the improvement is not significant and the computation is costly, it might not be worth the effort to do so.

Acknowledgement. Data sets for Figs. 1 and 7 except the rocker arm are downloaded from the web site: http://research.microsoft.com/~hoppe/ .

References

[1] Austin SP, Jerard RB, Drysdale RL, Comparison of discretization algorithms for NURBS surfaces with application to numerically controlled

machining, *Computer Aided Design* 1997, 29(1): 71-83.

[2] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design*, 1978, 10(6):350-355.

[3] DeRose T, Kass M, Truong T, Subdivision Surfaces in Character Animation, *Proceedings of SIGGRAPH*, 1998: 85-94.

[4] Stam J, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, *Proceedings of SIGGRAPH* 1998:395-404.

[5] Stam J, Evaluation of Loop Subdivision Surfaces, *SIGGRAPH'99 Course Notes*, 1999.

[6] Zorin D, Kristjansson D, Evaluation of Piecewise Smooth Subdivision Surfaces, *The Visual Computer*, 2002, 18(5/6):299-315.

[7] Lai S, Cheng F, Parametrization of General Catmull Clark Subdivision Surfaces and its Application, submitted. www.cs.uky.edu/∼cheng/PUBL/para.pdf.

[8] Garland M, Heckber P, Surface simplification using quadric error metrics, *Proceedings of SIGGRAPH* 1997:209-216.

[9] Settgast V, Müller K, Fünfzig C, et.al., Adaptive Tesselation of Subdivision Surfaces, In *Computers & Graphics*, 2004, pp:73-78.

[10] Amresh A, Farin G, Razdan A, Adaptive Subdivision Schemes for Triangular Meshes, In *Hierarchical and Geometric Methods in Scientific Visualization*, Springer-Verlag, 2002 pp:319-327.

[11] Wu X, Peters J, An Accurate Error Measure for Adaptive Subdivision Surfaces, In *Shape Modeling International*, 2005

[12] M. Bo, M. Amor, M. Doggett, et.al., Hardware Support for Adaptive Subdivision Surface Rendering, In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* 2001, pp:33-40.

[13] Müller K, Techmann T, Fellner D, Adaptive Ray Tracing of Subdivision Surfaces *Computer Graphics Forum* Vol 22, Issue 3 (Sept 2003).

[14] Smith J, Séquin C, Vertex-Centered Adaptive Subdivision, www.cs.berkeley.edu/∼jordans/pubs/vertexcentered.pdf.

[15] Isenberg T, Hartmann K, König H, Interest Value Driven Adaptive Subdivision, In *Simulation und Visualisierung*, March 6-7, 2003, Magdeburg, Germany.

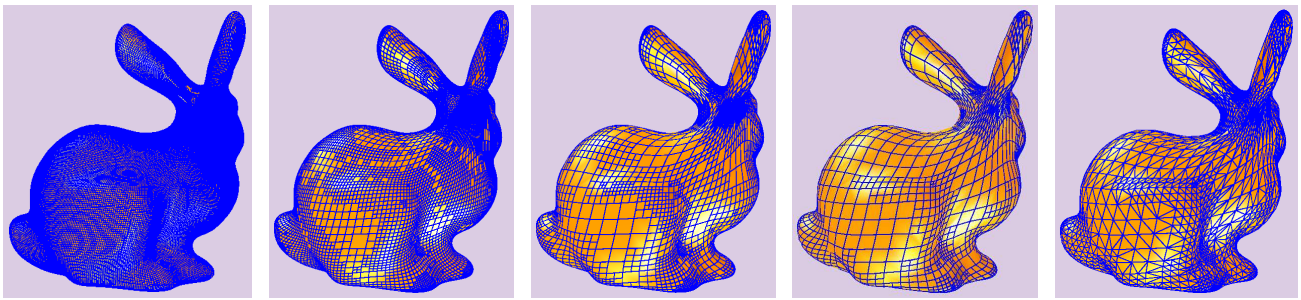
[16] Sederberg TW, Zheng J, Sewell D, Sabin M, Non-uniform recursive subdivision surfaces, *Proceedings of SIGGRAPH*, 1998:19-24.

[17] Sovakar A, Kobbelt L, API Design for adaptive subdivision schemes. 67-72, *Computers & Graphics*, Vol. 28, No. 1, Feb. 2004.

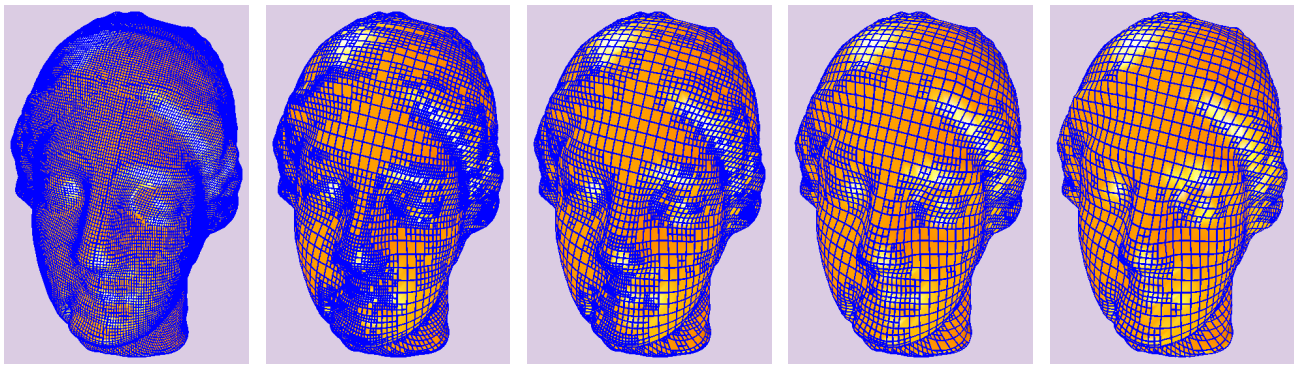
[18] Rose D, Kada M, Ertl T, On-the-Fly Adaptive Subdivision Terrain. In *Proceedings of the Vision Modeling and Visualization Conference*, Stuttgart, Germany, pp: 87-92, Nov. 2001.

[19] Wu X, Peters J, Interference detection for subdivision surfaces, *Computer Graphics Forum, Eurographics* 23(3):577-585, 2004.

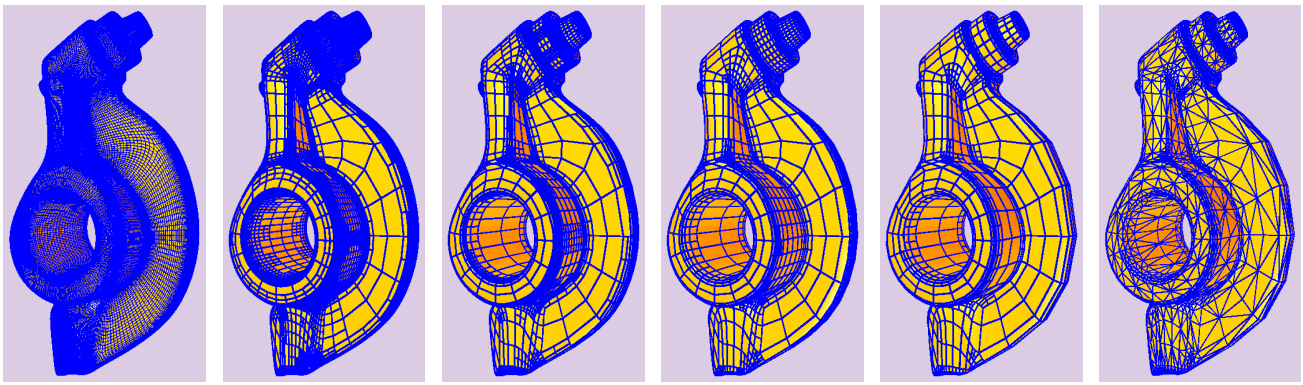
[20] Yong J, Cheng F, Adaptive Subdivision of Catmull-Clark Subdivision Surfaces, *Computer-Aided Design & Applications* 2(1-4):253-261, 2005.



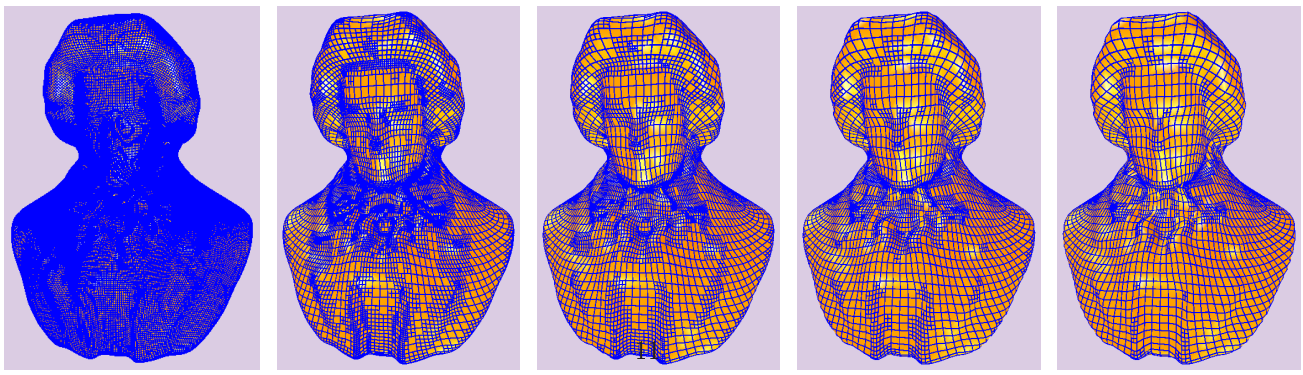
(a) Uniform (b) Adaptive (c) Adaptive (d) Adaptive (e) Triangulated



(f) Uniform (g) Adaptive (h) Adaptive (i) Adaptive (j) Adaptive



(k) Uniform (l) Adaptive (m) Adaptive (n) Adaptive (o) Adaptive (p) Triangulated



(q) Uniform (r) Adaptive (s) Adaptive (t) Adaptive (u) Adaptive

Figure 7: Adaptive rendering on surfaces with arbitrary topology.

Table 1: Extra information on Figures 1 and 7

Figure	Object	A|U|T	polygons	A/U Ratio	Depth	Error	RF
Fig. 1(a)	Gargoyle	U	16384	100.00%	2	0.0055	12%
Fig. 1(b)	Gargoyle	A	14311	5.46%	4	0.0030	6%
Fig. 1(c)	Gargoyle	A	5224	7.97%	3	0.0045	9%
Fig. 1(d)	Gargoyle	A	2500	15.26%	2	0.0055	12%
Fig. 1(e)	Gargoyle	T	6139	37.47%	2	0.0055	12%
Fig. 7(a)	Bunny	U	65536	100.00%	3	0.0008	3%
Fig. 7(b)	Bunny	A	32894	12.55%	4	0.0001	1%
Fig. 7(c)	Bunny	A	9181	14.01%	3	0.0008	3%
Fig. 7(d)	Bunny	A	3412	20.82%	2	0.0010	5%
Fig. 7(e)	Bunny	T	7697	46.98%	2	0.0010	5%
Fig. 7(f)	Venus	U	65536	100.00%	2	0.00095	8%
Fig. 7(g)	Venus	A	29830	2.84%	4	0.00015	3%
Fig. 7(h)	Venus	A	21841	2.08%	4	0.00035	4%
Fig. 7(i)	Venus	A	9763	3.72%	3	0.00060	6%
Fig. 7(j)	Venus	A	6178	9.43%	2	0.00095	8%
Fig. 7(k)	Rocker arm	U	90624	100.00%	4	1.2	3%
Fig. 7(l)	Rocker arm	A	36045	9.94%	5	0.85	1%
Fig. 7(m)	Rocker arm	A	10950	3.02%	5	1.0	2%
Fig. 7(n)	Rocker arm	A	5787	6.39%	4	1.2	3%
Fig. 7(o)	Rocker arm	A	2091	12.80%	3	1.5	5%
Fig. 7(p)	Rocker arm	T	4926	21.74%	3	1.5	5%
Fig. 7(q)	Beethoven	U	65536	100.00%	2	0.041	10%
Fig. 7(r)	Beethoven	A	20893	1.99%	4	0.006	4%
Fig. 7(s)	Beethoven	A	15622	1.48%	4	0.026	6%
Fig. 7(t)	Beethoven	A	7741	2.95%	3	0.035	8%
Fig. 7(u)	Beethoven	A	5230	7.99%	2	0.041	10%