Tessellation of Curved Surfaces under Highly Varying Transformations

Salim S. Abi-Ezzi and Leon A. Shirman

Graphics Technology Group
Sun Microsystems, Inc.
Mountain View, CA 94043, U.S.A.

We pursue the problem of step size determination for tessellating arbitrary degree polynomial and rational Bézier patches, under highly varying modeling and viewing transformations, to within post-viewing size and/or deviation thresholds specified in display coordinates. The technique involves the computation of derivative bounds of surfaces in modeling coordinates, and the mapping of these bounds into world coordinates (or lighting coordinates), where tessellation takes place by using norms of modeling transformations.

A key result of this work is a closed form expression for the maximum scale a perspective transformation is capable of at an arbitrary point in space. This result allows the mapping of thresholds from DC into WC (LC). In practice, while the step size determination needs to take place during every traversal, the costly operations of finding derivative bounds, computing norms of modeling transformations, and factoring viewing transformations take place at creation time.

1. Introduction

We address the problem of the speedy tessellation of curved surface primitives into appropriately sized triangles for display purposes. By appropriately sized we mean triangles that are not too small, which would result due to oversampling of the surface, nor too big, which would result with a bad quality rendering of the surface. Furthermore, the specific situation we are interested in involves highly varying modeling and viewing transformations; for example, the zoom effect of the view is assumed to vary considerably.

Given our assumption on the highly varying transformations, our approach involves tessellating surfaces (extracting the triangles) for every rendered frame. Thus, we refer to the approach as the dynamic tessellation of curved surfaces. This is in contrast with static tessellation techniques that involve reducing a surface to a bag of triangles, which then get processed for subsequent frames.

Today's graphics accelerators are characterized by having special VLSIs for rendering triangles, complemented with microprogramable floating-point processors for floating-point intensive tasks. In the past, techniques were developed to render surfaces directly without going through the intermediate form of triangles ([Lane80a], [Lien87]); however, for a variety of practical reasons such techniques didn't make it into special VLSI. While it is cost effective to produce special VLSI for the triangle primitive, being a common denominator primitive, it is not so for curved surfaces. Special VLSI for curved surfaces would be limited to a specific class of surfaces, for example cubic Bézies only, which is not consistent with real life requirements. Therefore, it becomes important to tessellate arbitrary classes of curved surfaces into triangles that get fed into triangle processing VLSI. It is best to perform the tessellation in firmware on the special floating-point processors. We discuss the dynamic tessellation technique both as an effective technique for processing curved surfaces, and as a compatible technique with graphics hardware accelerators.

In this paper, we focus on the step size determination aspect of the problem, while leaving the triangulation aspect as a topic for another paper. We use derivative bounds to determine tessellation step sizes to meet specified approximation criterion. Such a criterion may involve placing a threshold on the *deviation* of a surface from its tessellation, or on the *size* of the resulting triangles. For viewing purposes it is most useful to specify the threshold in display coordinates. This is the main reason why dynamic tessellation is so important in situations where the transformation is highly varying: as the transformation varies, the transformed surface changes accordingly (in display coordinates), and hence it needs to be tessellated repeatedly to keep satisfying the given threshold.

Our key contribution is a result that permits us to meet post-transformation thresholds based on pre-transformation derivative bounds; the technique works for general modeling transformations and for general perspectives and parallel views. This makes it possible to apply the costly computation of derivative bounds only once in modeling coordinates, while these bounds will be reusable under varying modeling and viewing transformations to meet size and/or deviation thresholds in display coordinates. Although the ideas we present apply to parametric surfaces in general, we choose to focus on arbitrary degree polynomial and rational Bézier surfaces.

2. The Dynamic Tessellation Approach

In this section we give the general guidelines of our approach. First, we talk about meeting size and deviation thresholds by using uniform tessellation techniques. Second, we discuss the lighting coordinate system as the proper coordinate system for performing the tessellation.

2.1. Uniform Tessellation

The uniform tessellation technique involves determining constant step sizes in the two parametric dimensions U and V for meeting a specified threshold. We did consider adaptive techniques such as Bézier subdivision [Farin88] and adaptive forward differencing [Lien87], but decided in favor of the uniform technique for the following reasons:

1. Simplicity: tessellating to a uniform grid results with simplicity and hence speed.
2. Efficiency: uniform tessellation permits the use of forward differencing for fast polynomial evaluation.
3. Adaptive techniques require checking the threshold after every evaluation step, which is a fairly costly operation. In contrast, uniform techniques require the onetime computation of derivative bounds before tessellation starts.

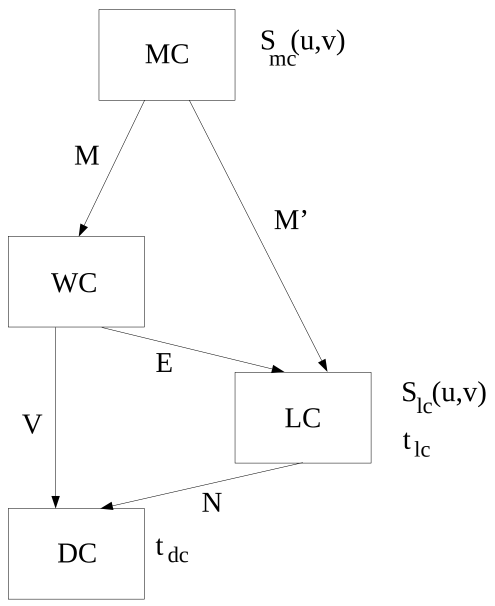

Figure 1: The relation of lighting coordinates to the pipeline. The derivative bounds are computed in MC, the approximation threshold is specified in DC, and tessellation takes place in LC.

In spite of the advantages of the uniform technique there are a few cases in which it doesn't work very well. In real life the majority of surfaces are well parameterized, so for these we use simple uniform tessellation and fast forward differencing. However, for surfaces that are ill-parameterized, uniform tessellation may result in oversampling and a large number of subpixel triangles in certain portions of the surface. We have identified two cases where this could happen: first, if the surface is rational with a wide difference between the weights of its control points, and second, if the eye point is very close to a certain portion of the surface. For these rare cases we recursively subdivide the surface as needed, and then use uniform tessellation on each resulting subsurface. Therefore, the solution is optimized for the common case and is still effective for the special cases.

2.2. Lighting Coordinates

The surfaces are tessellated in lighting coordinates (LC) [Abi-Ezzi90], which exhibit a set of useful properties for accurate lighting, fast transformations, and clipping. The idea is to factor a viewing transformation V into a product of a rigid factor E and a sparse non-rigid factor N. E relates world coordinates to lighting coordinates, while N relates lighting coordinates to display coordinates (Figure 1).

It is advantageous to tessellate in lighting coordinates in order to simultaneously achieve accurate lighting and avoid the costly operation of transforming the resulting triangles with V, which is typically a full 4×4 matrix. E is combined with the composite modeling transformation M to produce $M' = EM$, which is applied to the Bézier control points. Tessellation and lighting take place in LC and the resulting triangles are transformed with the sparse N to DC.

In order to tessellate in LC, we need derivative bounds of $S_{lc}(u,v) = M'S_{mc}(u,v)$ in LC and an approximation threshold t_{lc} in LC. This presents a problem given that we intend to compute derivative bounds of $S_{mc}(u,v)$ in MC, and enforce a threshold t_{dc} specified in

DC. We deal with this issue by producing derivative bounds of $S_{lc}(u,v)$ based on M and on precomputed derivative bounds of $S_{mc}(u,v)$ (Section 4.1). Also, we map t_{dc} to t_{lc}, such that if we meet threshold t_{lc} in LC we guarantee meeting t_{dc} in DC (Section 4.2).

3. Meeting Approximation Criteria

We address two types of approximation criteria: the size criterion which involves placing a threshold on the size of the resulting triangles, and the deviation criterion which involves placing a threshold on the deviation of these triangles from the actual surface. Although these two criteria are useful separately, they could also be very useful when used in a combined fashion. While the deviation criterion insures proximity of the tessellation to the surface, the size criterion insures a certain level of sampling for the lighting equation, which is important if the triangles are to be Gouraud shaded. For example, using the deviation criterion on surfaces that are linear in one direction such as cylinders will result with thin but potentially very long triangles.

In this section we discuss step size determination methods for meeting both size and deviation criterion, which require computing bounds on first and second order derivatives, respectively. Also we discuss methods for computing these derivative bounds.

3.1. Meeting a Size Criterion

For a parametric curve segment $C(t), t \in [0,1]$ that is first-order differentiable in the domain $[0,1]$, we seek a number n_s of uniformly spaced evaluations on this curve in order to meet a size threshold, t_s. The number of evaluations n_s corresponds to an increment δ in parameter space: $\delta = 1/n_s$. Let $||C(t)||$ be the Euclidean norm of $C(t)$, and $||C(t)||_{\max}$ be the maximum of that norm for $t \in [0,1]$.

The Mean Value Theorem from calculus states that

$$\forall\delta,\ \exists\xi < \delta: \quad C(t+\delta) - C(t) = \delta C'(t+\xi).$$

Therefore,

$$\delta \ \le \frac{t_s}{||C'(t)||_{\max}} \quad \Rightarrow \quad n_s \ = \ \left\lceil \frac{||C'(t)||_{\max}}{t_s} \right\rceil.$$

Lane [Lane80b] and Rockwood [Rockwood87] proved special cases of this general and simple result. They addressed the case of Bezier polynomials; the result above is applicable to any parametric function that is first-order continuous. Rockwood's generalization of his result to the case of Bezier rationals is informal and inaccurate based on the discussion below.

For a polynomial Bézier curve $P(t)$ of degree d and control points B_i, the first derivative bound can be comptuted using the convex hull property:

$$||P'(t)||_{\max} \quad \le \quad \max_{i=0}^{d-1} ||d(B_{i+1} - B_i)||.$$

In the case of a rational curve we have:

$$R(t) = \frac{P(t)}{w(t)} \quad \Rightarrow \quad R'(t) = \frac{w(t)P'(t) - w'(t)P(t)}{w^2(t)}.$$

We require such a rational to have only positive weights in order to ensure that it is differentiable in the domain of interest; i.e., $w(t) \neq 0, t \in [0,1]$. Recognizing that $R'(t)$ is the ratio of a degree $2d-1$ vector polynomial and a degree $2d$ scalar polynomial. We use degree elevation, multiplication, addition, and differentiation of Bézier polynomials [Farouki88] to express $R'(t)$ as a rational Bézier of degree $2d$, which will have positive weights as well. The upper bound for the norm of this rational Bézier polynomial can again be found using the convex hull property.

In the case of surfaces we use bounds on partial derivatives to compute the step sizes in both parametric directions. We have two size thresholds, t_s^u and t_s^v, in the U and V directions, which we use to compute n_s^u and n_s^v, respectively. n_s^u is dependent on the partial derivative in the U direction as follows:

$$n_s^u = \left\lceil \frac{\|S_u(u,v)\|_{\max}}{t_s^u} \right\rceil.$$

The expression for n_s^v is analogous. For both polynomial and rational surfaces, we use techniques similar to the ones above in order to determine $||S_u(u,v)||_{\max}$ and $||S_v(u,v)||_{\max}$ by using the convex hull property of Bézier surfaces.

3.2. Meeting a Deviation Criterion

The deviation criterion turns out to be related to the second order derivative of parametric curves and surfaces. In a fashion similar to the size criterion, we discuss how we can enforce the deviation criterion by computing bounds on derivatives.

We start by stating a powerful result from approximation theory that was first proven by de Boor [deBoor78, p. 39] for the case of scalar functions, then generalized by Wang [Wang84] for vector valued functions, and later enhanced by Filip et al [Filip86].

For a parametric curve $C(t)$ that is second order continuous in the range $[0,1]$ we have the following ([Wang84], [Filip86]):

$$n_d = \left\lceil \sqrt{\frac{\|C''(t)\|_{\max}}{8t_d}} \right\rceil.$$

We can compute close upper bounds of $||C''(t)||_{\max}$ by using methods similar to the case of $||C'(t)||_{\max}$ above. In the case of a Bézier polynomial we have:

$$||P''(t)||_{\max} \leq \max_{\mathrm{i=0}}^{\mathrm{d-2}} (||\mathrm{d(d-1)(B_{i+2} - 2B_{i+1} + B_i)}||).$$

In the case of a Bézier rational $R(t) = \frac{P(t)}{w(t)}$ we have:

$$R''(t) = \frac{w^2 P'' - 2ww'P' + \left(2w'^2 - ww''\right)P}{w^3}.$$

Thus, $R''(t)$ can be expressed as a rational Bézier polynomial of degree $3d$.

For surfaces Filip et al proved the following relationship (see [Filip86]):

$$t_d = \frac{1}{8}\left(\frac{1}{\left(n_d^u\right)^2} D_{uu} + \frac{2}{n_d^u n_d^v} D_{uv} + \frac{1}{\left(n_d^v\right)^2} D_{vv} \right), \qquad (1)$$

where

$$D_{uu} = \sup_{u,v\in[0,1]} \left\| \frac{\partial^2 S(u,v)}{\partial u^2} \right\|, \; D_{uv} = \sup_{u,v\in[0,1]} \left\| \frac{\partial^2 S(u,v)}{\partial u \partial v} \right\|, \; D_{vv} = \sup_{u,v\in[0,1]} \left\| \frac{\partial^2 S(u,v)}{\partial v^2} \right\|.$$

Equation 1 has two unknowns n_d^u and n_d^v. In case $D_{uu} = 0$ or $D_{vv} = 0$, we have a sufficient (but not necessary) condition for the surface to be linear in the corresponding direction, which implies that $n_d^u = 1$ or $n_d^v = 1$, respectively.

Otherwise, we improve Filip's result by deducing a second equation in n_d^u and n_d^v that minimizes the product $n_d^u n_d^v$, which in effect minimizes the number of triangles that it takes to satisfy Equation 1. After substituting $n_d^u \leftarrow k n_d^v$ in Equation 1 and solving for $n_d^u n_d^v$, it follows that

$$n_d^u n_d^v = \frac{D_{uu} + 2kD_{uv} + k^2 D_{vv}}{8kt_d}.$$

Since D_{uu}, D_{uv}, D_{vv}, and t_d are not dependent on k, it can be easily shown that the above function attains its minimum for

$$k = \sqrt{\frac{D_{uu}}{D_{vv}}},$$

and therefore

$$n_d^u = \frac{\sqrt{D_{uu}D_{vv} + D_{uv}\sqrt{D_{uu}D_{vv}}}}{2\sqrt{t_d D_{vv}}}, \;\; n_d^v = \frac{\sqrt{D_{uu}D_{vv} + D_{uv}\sqrt{D_{uu}D_{vv}}}}{2\sqrt{t_d D_{uu}}}. \tag{2}$$

As before, we compute bounds to the above partial derivatives by expressing them as polynomial or rational Bézier surfaces and then exploiting the convex hull property. In the case of rationals and due to the high degree bivariate rational derivatives that result, the computation of the bounds is expensive. This is one of the reasons for our approach to update and not to recompute bounds after changes to transformations.

4. Tessellation in Lighting Coordinates

In the previous section we assumed that the derivative bounds are computed, and the size and deviation thresholds are specified in the same coordinate system. In this section we show how bounds can be computed in MC and mapped into new bounds in LC. We also show how a threshold specified in DC can be scaled into a threshold in LC. This way tessellation can take place in LC, which is desirable (for reasons explained earlier).

4.1. Mapping the Derivative Bounds

In Figure 1, we show $M' = EM$ that maps MC to LC. Typically, M is affine* and hence can be factored into a product of a translation T and a 3×3 linear component L, so that $M' = ETL$. Since both E and T are Euclidean transformations we only need to

* In case M is projective then it gets treated like we treat perspective viewing transformations in Section 4.2.

study the influence of L on the derivative bounds as computed in MC in order to obtain new bounds in LC.

Since derivatives of surfaces are vectors, then we need to study the impact of a linear transformation on vectors. From linear algebra, the norm of L is defined and can be computed as follows:

$$||L|| \;=\; \max_{\vec{v}\in\Re^3-[0,0,0]} \frac{||L\vec{v}||}{||\vec{v}||} \;=\; \sqrt{\lambda_{\max}(L^T L)},$$

where $\lambda_{\max}$ is the maximum eigenvalue of $L^T L$. In the case where L is Euclidean, $L^T L = I$ and $||L|| = 1$. Also, in the case where L is a scale matrix, then $||L|| = \max(s_x, s_y, s_z)$. In general, to find $||L||$ we need to find the maximum eigenvalue of a 3×3 matrix, which involves finding the roots of the cubic characteristic polynomial:

$$\det\left(L^T L - \lambda I\right) = 0.$$

There are closed form expressions for these roots [Press88, p. 157]. Furthermore, since $L^T L$ is symmetric, then its characteristic polynomial always has 3 real roots; this is a well-known fact from linear algebra.

Obviously the norm of a modeling transformation can be computed at creation time. During traversal a bound to the norm of the composite modeling transformation can be obtained by using the inequality $\|AB\| \le \|A\|\,\|B\|$. In order to obtain the derivative bounds in LC, the derivative bounds in MC are multiplied by the norm of the composite modeling transformation. Therefore, the costly computations of the derivative bounds and of the norms are done at creation time, and minimal computational effort takes place at traversal time.

4.2. Scaling of the Threshold

Here we need to map t_{dc} to t_{lc}[†], so that if we tessellate to within threshold t_{lc} in LC, then we guarantee meeting t_{dc} in DC. This can be done by computing the maximum scale the viewing transformation is capable in the convex hull of the primitive being tessellated. Assuming this scale is s_V then $t_{lc} = t_{dc}/s_V$.

Let A and B be two points in 3D space; $[AB]$ be the segment from A to B; and $|AB|$ the length of $[AB]$. We define the scaling function of a viewing transformation V as follows:

$$s_V(A,B) = \frac{|(VA)(VB)|}{|AB|}.$$

If V is linear, then $s_V(A,B) \le ||L||$, where L is the linear part of V; this case is treated in Section 4.2.1. On the other hand, the scaling behavior of a perspective transformation is considerably more complicated. In this case, $s_V(A,B)$ is dependent on the orientation, magnitude, and location of segment $[AB]$. Also V can map certain finite segments into infinite ones. Thus, the concept of a norm is not applicable in this case; however, we can study the maximum scale of V within a certain subspace that doesn't get wrapped around infinity. This is the approach we take in Section 4.2.2.

[†] Note that if the threshold t is specified and is to be enforced in WC, then we can use it as is in LC.

4.2.1. Parallel View

Based on [Abi-Ezzi90] we can factor a parallel viewing transformation $V = LE$ as follows:

$$V = \begin{bmatrix} s_x & 0 & s_x h_x & 0 \\ 0 & s_y & s_y h_y & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & t_x \\ a_{10} & a_{11} & a_{12} & t_y \\ a_{20} & a_{21} & a_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where linear transformation L maps LC to DC. Therefore, we use the norm of L to scale the threshold; i.e. $t_{lc} = t_{dc}/||L||$ (Section 4.1). In the typical case where $h_x = h_y = 0$ then $||L|| = \max(s_x, s_y, s_z)$.

4.2.2. Perspective View

Based on [Abi-Ezzi90] we can factor a perspective viewing transformation $V = TQE$ as follows:

$$V = \begin{bmatrix} 1 & 0 & 0 & t'_x \\ 0 & 1 & 0 & t'_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & -1/P_z & 1 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & t_x \\ a_{10} & a_{11} & a_{12} & t_y \\ a_{20} & a_{21} & a_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where Q is perspective (non-linear) that maps LC to an intermediate coordinate (IC) system, which is a translation away from DC (Figure 2). P_z is the distance from the eye point P, which lies on the positive Z axis of LC. Furthermore, the far plane coincides with the XY plane, and the near plane is between P and the far plane.

The mathematical analysis of the scaling behavior of Q is rather complicated, and is addressed in detail in another paper [Abi-Ezzi91]. Therefore, here we only describe the final results of this analysis.

Assume that point A is chosen behind the near plane, and point B is arbitrary. Then, the scale of Q at point A in the direction of B can be defined as follows:

$$\xi_Q(A, B) = \lim_{|AB| \to 0} s_Q(A, B).$$

We seeked a closed form expression for the maximum attained by this function under all possible directions in 3D (i.e. $\forall B \in \Re^3 - A$). The result is a fairly cheap expression to compute ([Abi-Ezzi91]), as follows:

$$\xi^2_{\max,Q}(A) = \frac{P_z^2}{2d^4}\left(X_4 + X_1 + \sqrt{(X_4 - X_1)^2 + 4X_1X_2}\right), \quad \text{where} \qquad (3)$$
$$X_1 = d^2s^2,\ X_2 = h^2s^2,\ X_3 = P_z^2 s_z^2,\ X_4 = X_2 + X_3.$$

Here, d is the distance between the Z coordinate of A and the eye point (which lies on the Z axis), h is the distance from A to the Z axis, and $s = \max(s_x, s_y)$; usually $s_x = s_y$ in practice. Clearly the quantity under the square root is always positive, and also the scale factor increases as d decreases and h increases.

Now suppose we have a Bézier patch $S(u, v)$ with control points B_{ij} in LC, and a perspective transformation Q with a viewing volume (truncated pyramid) determined by a

Figure 2: The factoring of a perspective transformation.

near and far planes, and a window on a viewing plane. The near plane is at distance d_n from the eye point P, and the window is determined by its four corner points W_k.

We need a bound to the maximum scale that Q is capable of on the non-clipped portion of $S(u,v)$, since there is no need to meet the threshold within the clipped away portion. For point $A, \quad A_z \neq P_z$ we define the following function:

$$\tau(A) = \tan^2\left(\angle(A,P,O)\right) = \frac{A_x^2 + A_y^2}{\left(P_z - A_z\right)^2}.$$

We get this bound by substituting the following values for d and h in Equation 3:

$$\begin{aligned} t &= \min\left(\max_{i,j}\left(\tau(B_{ij})\right), \max_{k}\left(\tau(W_k)\right)\right), \\ d &= \max\left(\min_{i,j}\left(P_z - B_{ij,z}\right), d_n\right), \\ h^2 &= d^2 t. \end{aligned} \tag{4}$$

So, even if a surface is wraped around infinity by the perspective transformation, the approach we take here will guarantee a bounded maximum scale. Also, in case the viewing volume is a double truncated pyramid [Abi-Ezzi90], the above choice of d and h can be generalized in a straight forward fashion.

Patch Degree	Patch Type								
	Polynomial			Rational nominal weights			Rational large weights		
2	0.96	1.34	1.65	1.05	1.11	1.18	1.07	1.18	1.46
3	1.32	1.78	2.37	1.05	1.35	1.90	1.22	1.54	1.85
4	1.47	1.93	2.18	1.20	1.45	1.86	1.22	1.53	1.84
5	1.58	1.92	2.12	1.21	1.48	1.73	1.21	1.52	1.83

Table 1: In this table we have the accumulated results of running the above experiment at least for ten different cases per category. We list the minimum, average, and maximum that the ratio of number of triangles n_{lc}/n_{dc} attains over the number of experiments. By rationals with large weights we mean a ratio of at least 3 between the largest weight and the smallest weight of the control points.

4.2.3. Discussion

In order to study the appropriateness of the approach in practice, we implemented a prototype system *SurfTool* written in C++, running on SPARCstations using SunPHIGS. *SurfTool* allows to interactively create surfaces, change views, and run experiments and generate statistics on the different aspects of the dynamic tessellation process. We designed a particular experiment to study the appropriateness of using the perspective scaling approach in order to avoid computing derivative bounds for every change in the view.

Suppose we have a patch S_{lc} in LC, a perspective viewing transformation Q that maps LC to DC, and a deviation threshold t_{dc} to be enforced in DC. For different settings of S_{lc} and Q (that we varied interactively), we compared the following two different methods for determining the number of triangles needed to meet the specified deviation threshold:

1. Map the threshold t_{dc} into t_{lc}, and determine step sizes based on pre-computed derivative bounds of S_{lc}; this is the method we are advocating.
2. Transform S_{lc} into S_{dc} via Q, compute derivative bounds of S_{dc}, and determine step sizes based directly on threshold t_{dc}.

Note than due to the effect of Q, in the second method S_{dc} is rational even if S_{lc} is polynomial.

The obvious advantage of the first method is the fact that the costly bound computation of rational derivatives is not needed every time the view is changed. We simply determine a new scale factor and reuse the precomputed bounds. The price for this is that the first method will require more triangles in general, since the maximum scale of Q is used. So, the intent of the experiments is to indicate how many more triangles the first method requires as compared to the second method. These results are summarized in Table 1.

By inspecting the entries in Table 1, we make the following observations:

1. The first method results in relatively more triangles for higher degrees.
2. It is rare but possible for the first method to result with fewer triangles than the second method.
3. The numbers in the polynomial column are considerably larger than those in the rational column. This is due to the fact that, for a polynomial patch of degree d, its second derivative is of degree $d - 2$, while for a rational patch of the same degree, its second

derivative is of degree $3d$ (Section 3.2). In practice, for the polynomial case it is advisable to degree elevate the derivatives before the convex hull is used for computing the bounds.

Furthermore, the second method has a problem when S_{dc} intersects the eye plane; i.e. it wraps around infinity and its derivatives are unbounded. The first method will easily handle this case, since the scaling is clamped at the front plane (Equation 4).

5. The case of Ill-parametrized Surfaces

It is possible, though rare in practice, that surfaces are ill-parameterized in DC. This causes uniform step sizes in parameter space to result with wildly varying step sizes in DC. This phenomenon is caused by one of two reasons: the surface could be ill-parameterized in MC to start with or a portion of the surface could be very close to the eye point in LC. In both cases when we uniformly tessellate to meet a specified criteria, we end up oversampling in certain regions of the surface. This results with the generation of a large number of mostly subpixel triangles.

We present a quick technique for the detection of ill-parameterized surfaces, in which case we recursively subdivide the surface, so that uniform tessellation within each subsurface will diminish the number of generated triangles by avoiding subpixel triangles.

We define the function $\eta(u,v)$ as a counterpart to the product $n_u^d n_v^d$ (Equation 2), except that while the latter is a constant based on the bounds D_{uu}, D_{vv}, and D_{uv}, the former is a function based on the scalar functions $||S_{uu}(u,v)||, ||S_{vv}(u,v)||$, and $||S_{uv}(u,v)||$. The function $\eta(u,v)$ measures the relative density of triangles needed in the neighborhood of a point $A = S(u^*,v^*)$ to meet a certain threshold. So, the higher $\eta(u^*,v^*)$ is, the more dense the triangles resulting from a uniform tessellation are in the neighborhood of A. From Equation 2 we define $\eta(u,v)$ as follows:

$$\eta(u,v) = ||S_{uv}(u,v)|| + \sqrt{||S_{uu}(u,v)||\ ||S_{vv}(u,v)||}.$$

In order to decide if a patch needs to be subdivided we compare the triangle density at each of the four corners of the patch to the triangle density $\eta_{\max}$ required by a uniform tessellation across the whole patch, as follows:

$$\begin{aligned}\eta_{\min} &= \min\left(\eta(0,0), \eta(0,1), \eta(1,0), \eta(1,1)\right),\\ \eta_{\max} &= n_d^u n_d^v = D_{uv} + \sqrt{D_{uu} D_{vv}}.\end{aligned}$$

In case $\eta_{\max}/\eta_{\min}$ is lower than a certain threshold then the patch is subdivided. The choice of that threshold depends on the graphics hardware; essentially it represents a trade off between generating more triangles versus the cost of subdivision. In case the hardware can process triangles very fast then the threshold is set lower.

The evaluation of η at the corner points essentially comes for free, since the needed parameters are obtained during the computation of the derivative bounds. Therefore, this approach filters the common case where subdivision is not needed fairly quickly. Furthermore, after a patch is subdivided we subdivide its derivative functions correspondingly to obtain derivative bounds for the subpatches. It is much cheaper to subdivide a derivative patch as opposed to computing the derivatives for each subpatch.

6. Conclusion

We presented a formal, general, and comprehensive technique for the tessellation of arbitrary degree polynomial and rational Bézier patches under heavily changing modeling and viewing transformations, while enforcing size and/or deviation thresholds specified in DC. The technique involves performing the complex operations of finding derivative bounds, computing norms of transformations, and factoring of views at data creation time, while performing simple operations such as mapping the bounds and the approximation threshold into LC at traversal time.

NURB surfaces are converted into Bézier patches before they are subjected to the technique above. Since adjacent patches may be tessellated at different step sizes, special care needs to be taken along common borders in order to avoid any gaps. Trimming is an orthogonal problem to the one we addressed above; nevertheless, it is a very important problem since trimmed NURBs are the primitives of choice in recent formal and defacto geometric modeling and graphics standards. We will address trimming and the adjacency issues in NURBs in a future paper.

The techniques that we present constitute a testament to the fact that the generality of specification of graphical data does not necessarily mean poor performance. High performance can be achieved through the compilation of the graphical data into forms that can be processed very quickly during traversal. Although all surfaces are specified using the same interface, the fact that a surface is almost planar, or of constant curvature, or ill-behaved can be detected through its derivative bounds. Similarly, essential aspects of a general modeling transformation can be represented through its norm. For more details on this idea of unification between generality, formalism, and efficiency see [Abi-Ezzi89].

One aspect of our technique that we feel needs more work is a faster method for finding derivative bounds of rational surfaces. Although the technique we use is robust and produces tight bounds, it is expensive and it is not practical for high degree patches. This continues to be an area for more work.

7. References

1. [Abi-Ezzi89] Salim Abi-Ezzi: “The graphical Processing of B-splines in a Highly Dynamic Environment,” RPI Ph.D. dissertation, RDRC-TR-89001, Troy, New York (May 1989).

2. [Abi-Ezzi90] Salim Abi-Ezzi and Michael Wozny: “Factoring a Homogeneous Transformation for a more Efficient Graphics Pipeline,” *Proc. EUROGRAPHICS ’90*, (September 1990).

3. [Abi-Ezzi91] Salim Abi-Ezzi and Leon Shirman: “The Scaling Behavior of a Perspective Transformation,” in preparation.

4. [deBoor78] Carl de Boor: *A Practical Guide to Splines*, Springer-Verlag, 1978.

5. [Farin88] Gerald Farin: *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, 1988.

6. [Farouki88] R. Farouki and V. Rajan: “Algorithms for Polynomials in Bernstein form,” *Computer Aided Geometric Design*, 5(1988), pp 1–26, 1988.

7. [Filip86] Daniel Philip, Robert Magedson, and Robert Markot: “Surface Algorithms Using Bounds on Derivatives,” *Computer Aided Geometric Design*, 3(1986), pp 295–311, 1986.

8. [Lane80a] Jeffrey Lane, Loren Carpenter, Turner Whitted, and James Blinn: “Scan Line Methods for Displaying Parametrically Defined Surfaces,” *Communications of the ACM*, 23(1), January 1980.

9. [Lane80b] Jeffrey Lane and Richard Riesenfeld: “A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces,” IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 2(1), pp 35–46, January 1980.

10. [Lien87] Sheue-Ling Lien, Michael Shantz, and Vaughan Pratt: “Adaptive Forward Differencing for Rendering Curves and Surfaces,” *Computer Graphics*, 21(4), pp 111–117, July 1987.

11. [Press88] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling: *Numerical Recipes in C,* Cambridge University Press, New York, 1988.

12. [Rockwood87] Alyn Rockwood: “A Generalized Scanning Technique for Display of Parametrically Defined Surfaces,” IEEE *Computer Graphics & Applications*, pp 15–26, August 1987.

13. [Wang84] G. Wang: "The Subdivision Method for Finding the Intersection Between Two Bézier Curves or Surfaces", Zhejiang University Journal, Special issue on Computational Geometry (in Chinese), 1984.