Exact Evaluation of Non-Polynomial Subdivision Schemes at Rational Parameter Values

Scott Schaefer
Texas A&M University
3112 Texas A&M University
College Station, TX 77843
schaefer@cs.tamu.edu

Joe Warren
Rice University
6100 Main St
Houston, TX 77005
jwarren@cs.rice.edu

Abstract

In this paper, we describe a method for exact evaluation of a limit mesh defined via subdivision on a uniform grid of any size. Other exact evaluation technique either restrict the grids to have subdivision sampling and are, hence, exponentially increasing in size or make assumptions about the underlying surface being piecewise polynomial (Stam's method is a widely used technique that makes this assumption). As opposed to Stam's technique, our method works for both polynomial and non-polynomial schemes. The values for this exact evaluation scheme can be computed via a simple system of linear equation derived from the scaling relations associated with the scheme or, equivalently, as the dominant left eigenvector of an upsampled subdivision matrix associated with the scheme. To illustrate one possible application of this method, we demonstrate how to generate adaptive polygonalizations of a non-polynomial quad-based subdivision surfaces using our exact evaluation method. Our method guarantees a water-tight tessellation no matter how the surface is sampled and is quite fast. We achieve tessellation rates of over 33.5 million triangles/second using a CPU implementation.

1. Introduction

Curves and surfaces defined via subdivision have become a fixture of the computer modeling and animation industry. Commercial modeling packages such as Maya as well as leading animation studios such as Pixar use subdivision surfaces as one of their basic modeling primitives. As a result of this proliferation, methods for manipulating and processing subdivision surfaces are a topic of practical importance in Computer Graphics.

Perhaps the biggest impediment to developing such methods is the nature of how shapes are defined via subdivision. Given a coarse *base* mesh p_0, a subdivision scheme defines an increasingly detailed sequence of meshes p_1, p_2, p_3, ... via the recurrence

$$p_{j+1} = Sp_j \tag{1}$$

Here, S is an operator that *refines* the mesh p_j to form the new mesh p_{j+1}. For most simple subdivision schemes, the meshes p_j can be modeled as a vector of control points and the operator S can be viewed as a linear operator (i.e; a matrix) that acts on p_j. If the operator S is chosen appropriately, the sequence of meshes $p_0, p_1, p_2, \ldots$ converges to a limit mesh p_∞ that approximates the coarse mesh p_0 [16].

The definition of the final mesh p_∞ in terms of limits appears to be awkward in comparison to other modeling schemes such as B-splines where the associated curve or surface has a direct definition in terms of piecewise polynomials. For schemes with explicit piecewise polynomial definitions, computing the exact position of points on associated curves or surfaces corresponds to just evaluating the appropriate polynomial at a particular parameter value [11]. For subdivision surfaces, this exact evaluation process is more difficult since there is no explicit definition of the limit surfaces in terms of polynomials.

Currently, the authors are only aware of two methods for computing the exact limit positions of points on a subdivision surface. One approach is to simply apply the subdivision scheme several times and then apply a limit *stencil*[1] to reposition the vertices of the resulting mesh so that they lie on the limit surface. This limit stencil can be computed as the dominant left eigenvector of the subdivision matrix S associated with the scheme [4]. The drawback of this approach is that the final mesh size must be compatible with the subdivision process. (In the case of binary subdivision, the mesh size must be uniform and increases ex-

[1] A stencil is a set of weights applied to a set of locally-adjacent vertices in the mesh. Stencils are also sometimes called masks or rules in subdivision terminology.

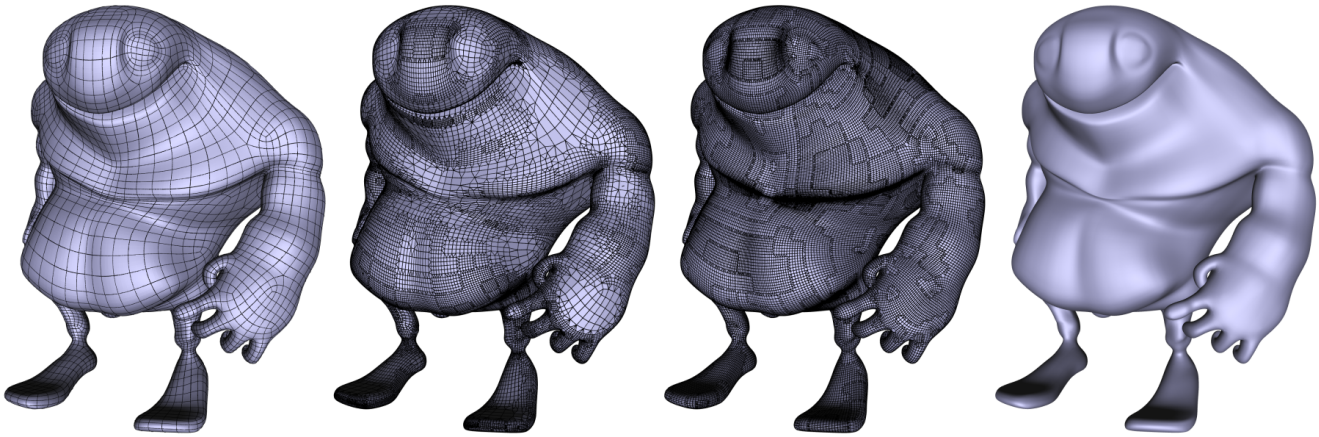

Figure 1. Adaptive tessellation of a subdivision surface that is non-polynomial everywhere using our algorithm. From left to right: patch structure from base surface, adaptive tessellation based on approximate curvature, view-dependent tessellation, and the shaded surface.

ponentially by a factor of 4.) This same type of evaluation also arises in the evaluation of wavelet basis functions in the tensor-product setting that are defined from recursive relationships [15] again with the restriction to the exponential sampling defined by subdivision on grids of spacing $\frac{1}{2^n}$.

An alternative approach, developed by Stam [14], allows for direct evaluation of certain subdivision schemes near extraordinary vertices. Stam's basic idea for evaluating a subdivision surface near an extraordinary vertex is to subdivide the mesh until the desired point lies on a locally uniform portion of the mesh and then evaluate using the piecewise polynomial definition associated with the uniform rules for the scheme. For schemes like Catmull-Clark [2] or Loop [9] whose uniform rules generate piecewise polynomial limit surfaces, this approach works well. Unfortunately, methods such the butterfly scheme [18] and $\sqrt{3}$ scheme [7] do not generate piecewise polynomial limit surfaces in the uniform case. For these schemes, Stam's exact evaluation method simply does not apply.

Contributions

We describe a method for exact evaluation of a limit mesh defined via subdivision on a uniform grid of *any* size. As opposed to Stam's method, our technique operates on subdivision schemes that produce polynomial or non-polynomial curves/surfaces. The stencils for this exact evaluation scheme can be computed via a simple system of linear equation derived from the scaling relations associated with the scheme or, equivalently, as the dominant left eigenvector of an upsampled subdivision matrix associated with the scheme. We then show how to use this exact evaluation scheme to create adaptive polygonalizations of non-polynomial quad-based subdivision surfaces.

2. Exact evaluation via scaling relations

The key to our exact evaluation method is that fact that the limit functions associated with a subdivision scheme satisfy a recurrence relation based on the entries of the subdivision matrix S. To explain our method, we consider two cases in this section: uniform curve schemes and non-uniform curve schemes. The extension of our method to the case of surfaces is then relatively straightforward. (Such a surface extension is considered in Section 4.)

In the case of uniform curve schemes, the columns of the subdivision matrix S are two-shifts of a single fundamental sequence of numbers s_i. If the subdivision is convergent, the limit curve p_∞ associated with the coarse curve p_0 can be parameterized by a single variable x yielding the associated limit function $p_\infty[x]$. Due to the linearity of the subdivision process, this limit function $p_\infty[x]$ can be written as a linear combination of translates of a single scaling function $\phi[x]$. Specifically, if the i^{th} point of the j^{th} discrete curve p_j, p_j^i, is treated as lying at the parameter value $x = \frac{i}{2^j}$, the limit function $p_\infty[x]$ can be written as a linear combination of the integer translates of the scaling function

$$p_\infty[x] = \sum_{i \in \mathsf{Z}} p_0^i \phi[x - i] \tag{2}$$

where Z is the set of integers.

Combining Equations 1 and 2 implies that the function $\phi[x]$ satisfies the *scaling relation*

$$\phi[x] = \sum_{i \in \mathsf{Z}} s_i \phi[2x - i] \tag{3}$$

Notice that this scaling function holds for *all* values of x and not just at the points $x = \frac{i}{2^j}$ produced by subdivision.

Our goal is then to use this scaling relation to compute the exact values of the scaling function $\phi[x]$ on a uniform grid; i.e; for a given fixed positive integer n, compute $\phi[\alpha]$ where $\alpha \in \frac{1}{n}\mathsf{Z}$. To this end, let us assume that the scaling function $\phi[x]$ is supported on the interval $[-m, m]$. Substituting $x = \alpha$ into the scaling relation where α lies strictly in the range $(-m, m)$ yields a set of $2mn - 1$ homogeneous equations in $2mn - 1$ variables.

$$\phi[\alpha] = \sum_{i \in \mathsf{Z}} s_i \phi[2\alpha - i].$$

Note that if $\alpha \notin (-m, m)$, the value $\phi[\alpha]$ is zero due to the sparsity assumption.

Since these equations are homogeneous, their solution only specifies the values $\phi[\alpha]$ up to at most a fixed common multiple. To arrive at specific values for $\phi[\alpha]$, we observe that, for stationary schemes, convergence implies that the sum of the integer translates of the scaling function must be identically 1 [16, p.71].

$$\sum_{i \in \mathsf{Z}} \phi[x - i] = 1 \tag{4}$$

Substituting $x = \alpha$ for $\alpha = 0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}$ yields n auxiliary non-homogeneous equations.

$$\sum_{i \in \mathsf{Z}} \phi[\alpha - i] = 1$$

Using these two systems of equations, we then solve for the unknowns $\phi[\alpha]$. If the subdivision scheme is convergent, the actual values of $\phi[\alpha]$ are a solution to this system of equations. Although we have no formal proof that this system of equations always has these values as their unique solution, every example scheme that we have tried always yields a single solution. Developing a proof of uniqueness is one of our topics for future research. Also notice that this evaluation method cannot evaluate the surface at arbitrary parameter values (specifically irrational numbers) but is restricted to rational parameter values.

To complete this subsection, we compute the values of the scaling function for the four-point interpolatory scheme of [3] for $n = 3$. To this end, we observe that the scaling relation for this scheme has the form

$$\begin{aligned} \phi[x] \quad = \quad & \frac{1}{16}(-\phi[2x+3] + 9\phi[2x+1] + 16\phi[2x] + \\ & 9\phi[2x-1] - \phi[2x-3]) \end{aligned}$$

Given this scaling function is support on the interval $[-3, 3]$, we form 17 equations in 17 variable by substituting $x = \alpha$ for $\alpha = \frac{-8}{3}, \frac{-7}{3}, \frac{-6}{3}, \ldots, \frac{6}{3}, \frac{7}{3}, \frac{8}{3}$ into the scaling relation. Adding the three partition of unity constraints

$$\sum_{i=-2}^{3} \phi[\alpha - i] = 1$$

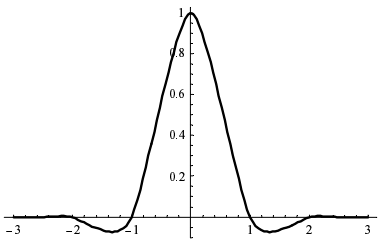

Figure 2. Plot of the basis function values returned from our method for the four-point scheme with $n = 21$.

for $\alpha = 0, \frac{1}{3}, \frac{2}{3}$ yields a system of 20 equation in 17 unknowns. Assembling and solving these equations yields a solution vector $\left\{\phi[\frac{-8}{3}], \phi[\frac{-7}{3}], \ldots, \phi[\frac{7}{3}], \phi[\frac{8}{3}]\right\}$ of the form

$$\begin{aligned} \frac{1}{5589}\{&0, -1, 16, 0, -256, -410, 0, 2000, 4240, \\ &5589, 4240, 2000, 0, -410, -256, 0, 16, -1\} \end{aligned}$$

Figure 2 shows a plot of the solution vector for $n = 21$.

Given that our ultimate goal is to perform exact evaluation on subdivision surfaces with extraordinary vertices, we must eventually move from the uniform case to the non-uniform case. To conclude this section, we consider a stationary, but non-uniform curve example. Consider a non-uniform cubic B-spline whose knots t_i satisfy $t_i = i$ if $i > 0$ and $t_i = 2i$ if $i < 0$. If we subdivide this spline by inserting a new knot between every pair of existing knots, the resulting subdivision scheme is a stationary one that satisfies $p_{j+1} = Sp_j$. Restricted to the two-ring of the origin, this matrix S has the form

$$\begin{pmatrix} \frac{1}{8} & \frac{25}{32} & \frac{3}{32} & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ 0 & \frac{5}{24} & \frac{29}{40} & \frac{1}{15} & 0 \\ 0 & 0 & \frac{3}{5} & \frac{2}{5} & 0 \\ 0 & 0 & \frac{3}{20} & \frac{29}{40} & \frac{1}{8} \end{pmatrix}$$

The remaining rules for the scheme away from the origin are the standard rules for cubic B-spline subdivision.

The first question that we must address in examining this scheme is how should we parameterize the control mesh p_j. The standard technique for parameterizing both curve (and surface meshes) is to assign a uniform parameterization based on mesh spacing. However, the subdivision matrix itself contains none of this information. Therefore, we still assign the mesh point p_i^j the parameter value $x = \frac{i}{2^j}$ even though the associated subdivision rules are derived from non-uniform b-splines. Given this uniform parameterization, we may still consider piecewise linear functions $p_j[x]$ and their associated limit function $p_\infty[x]$. Now, our

goal is to compute the exact values of $p_\infty[\alpha]$ for $\alpha = \frac{i}{n}$ and verify that these values are consistent with the values produced by the piecewise polynomial definition of the underlying B-spline.

The key to this computation is to observe that the infinite subdivision matrix S has five types of columns. Three columns lying in the one-ring of the origin are distinct due to the effect of the non-uniform knot spacing at the origin. The remaining columns to both the left and right of the origin are two-shifts of the columns for uniform cubic B-splines. (We treat the left columns and right columns as being of two different types for the sake of simpler pedagogy.) Each of these column types has its own associated scaling function $\phi_{-2}[x]$, $\phi_{-1}[x]$, $\phi_0[x]$, $\phi_1[x]$, $\phi_2[x]$. Now, the entries of each column specify the scaling relation for the associated scaling function, i.e.

$$\begin{aligned}
\phi_{-2}[x] &= \frac{1}{8}\phi_{-2}[2x+2] + \frac{1}{2}\phi_{-2}[2x+1] + \\
&\quad \frac{3}{4}\phi_{-2}[2x] + \frac{1}{2}\phi_{-2}[2x-1] + \frac{1}{8}\phi_{-2}[2x-2] \\
\phi_{-1}[x] &= \frac{1}{8}\phi_{-2}[2x+2] + \frac{1}{2}\phi_{-2}[2x+1] + \\
&\quad \frac{25}{32}\phi_{-2}[2x] + \frac{5}{8}\phi_{-1}[2x-1] + \frac{5}{24}\phi_0[2x-2] \\
\phi_0[x] &= \frac{3}{32}\phi_{-2}[2x+2] + \frac{3}{8}\phi_{-1}[2x+1] + \\
&\quad \frac{29}{40}\phi_0[2x] + \frac{3}{5}\phi_1[2x-1] + \frac{3}{20}\phi_2[2x-2] \\
\phi_1[x] &= \frac{1}{15}\phi_0[2x+2] + \frac{2}{5}\phi_1[2x+1] + \\
&\quad \frac{29}{40}\phi_2[2x] + \frac{1}{2}\phi_2[2x-1] + \frac{1}{8}\phi_2[2x-2] \\
\phi_2[x] &= \frac{1}{8}\phi_2[2x+2] + \frac{1}{2}\phi_2[2x+1] + \\
&\quad \frac{3}{4}\phi_2[2x] + \frac{1}{2}\phi_2[2x-1] + \frac{1}{8}\phi_2[2x-2]
\end{aligned}$$

To solve for the exact values of these scaling functions $\phi_n[x]$, we simply repeat the same construction used in the uniform case. In particular, we substitute $x = \alpha$ where $\alpha = \frac{i}{n}$ into these scaling relations, construct an auxiliary system of equations enforcing a partition of unity and solve the resulting system of equations.

Clearly, the scaling functions $\phi_{-2}[x]$ and $\phi_2[x]$ are exactly the scaling functions for uniform cubic B-splines. And indeed, the exact values produced by our method concur with the exact values for the uniform cubic B-splines. The three remaining scaling functions $\phi_{-1}[x+1]$, $\phi_0[x]$ and $\phi_1[x-1]$ agree with the scaling functions associated with the non-uniform cubic B-splines with one apparent exception: the scaling functions are not smooth at the origin. In fact, this tangent discontinuity is due to the reparameterization of $x = 2t$ for $t > 0$ used in relating the B-spline parameterization and the uniform parameterization for our curve mesh. Accounting for this reparameterization, our exact evaluation method produced the same values as generated by non-uniform cubic B-splines.

3 Exact evaluation via left eigenvectors of an upsampled subdivision matrix

The previous section described a method for deriving a set of equations whose solution was the exact values of the scaling function on the grid $\frac{1}{n}\mathsf{Z}$. In this section, we sketch the theoretical underpinnings of our method by relating the linear system generated by these equations to a traditional method for computing the exact values of a subdivision scheme on Z. For those interested solely in implementing our method for surface schemes, we suggest skipping this section.

In the uniform curve case, computing the exact values of the scaling function $\phi[x]$ on the integer grid Z is well-known and relatively straight forward [16]. Recall the fundamental subdivision process of Equation 1. Iterating this process yields

$$p_j = S^j p_0.$$

Observe that if we set $p_0^i = 1$ and 0 otherwise, the limit function $p_\infty[x]$ is simply the scaling function $\phi[x]$. In this case, the value of the scaling function $\phi[x]$ on Z are simply the entries of the row of S with index zero. For convergent subdivision schemes, this row is simply the dominant left eigenvector associated with the subdivision scheme. In particular, the values satisfy

$$\{.,\phi[-1],\phi[0],\phi[1],.\} = \{.,\phi[-1],\phi[0],\phi[1],.\}S.$$

Notice that this matrix formulation consists of exactly the same equations that instantiating the fundamental scaling relation with $x \in \mathsf{Z}$ generates.

Interestingly, instantiating the fundamental scaling relation with $x \in \frac{1}{n}\mathsf{Z}$ can also be interpreted in terms of a subdivision process. If we reparameterize via $x = \frac{\hat{x}}{n}$, the scaling function satisfies the relation

$$\phi[\frac{\hat{x}}{n}] = \sum_{i\in\mathsf{Z}} s_i \phi[2\frac{\hat{x}}{n} - i].$$

Now, if we define a new scaling function $\hat{\phi}[\hat{x}] = \phi[\frac{\hat{x}}{n}]$, this reparameterized scaling function is a stretched copy of ϕ with its support widened by a factor of n. Applying this definition, we arrive at a new recurrence relation of the form

$$\hat{\phi}[\hat{x}] = \sum_{i\in\mathsf{Z}} s_i \hat{\phi}[2\hat{x} - in].$$

Observe that due to the indexing, only every n^{th} translate of $\hat{\phi}[\hat{x}]$ is used in the recurrence.

To define the effect of this upsampling, we introduce a generating function notation for the subdivision mask. Given a uniform subdivision matrix S representing binary subdivision, the columns of S will be two-shifts of a fundamental set of numbers s_i that encode the scaling relationship for the basis function. Collecting these numbers as coefficients of a polynomial yields the *subdivision mask* $s[z]$ associated with the scheme:

$$s[z] = \sum_{i \in \mathsf{Z}} s_i z^i.$$

One advantage of expressing the subdivision process in terms of a subdivision mask is that the subdivision process can be succinctly expressed via the recurrence $p_{j+1}[z] = s[z]p_j[z^2]$ where $p_j[z]$ is the generating function with terms of the form $p_j^i z^i$.

If we define a new subdivision mask $\hat{s}[z]$ in terms of an upsampled version of the original subdivision mask $s[z]$ via $\hat{s}[z] = s[z^n]$, the subdivision process corresponding to this mask has a scaling relation of the form

$$\hat{\phi}[\hat{x}] = \sum_{i \in \mathsf{Z}} \hat{s}_i \hat{\phi}[2\hat{x} - i].$$

Now, the exact values of new scaling function $\hat{\phi}[\hat{x}]$ on Z are exactly the values of the original scaling function $\phi[x]$ on the grid $\frac{1}{n}\mathsf{Z}$. To compute these values, we simply construct the subdivision matrix $\hat{S}$ corresponding to the upsampled mask $\hat{s}[z]$. Each column of this upsampled subdivision matrix $\hat{S}$ is a two-shift of a column of S with $n-1$ zeros inserted between each entry.

To compute the desired exact values, we construct a local version of the matrix $\hat{S}$ and compute its dominant left eigenvectors. If n is a power of 2, this dominant eigenvector is unique up to a scale factor. We can derive the exact value by constraining the scaled entries to satisfy the partition of unity property. Similarly, when $\hat{S}$ has several dominant eigenvectors (which is the usual case), we solve for a linear combination of these eigenvectors that satisfies the partition of unity property.

In the case of non-uniform schemes, the derivation of this upsampled subdivision matrix whose dominant left eigenvectors contain the exact values is subtle and would take this paper in a theoretical direction too far from the main focus of this conference. Instead, we leave this topic to a future paper. However, to illustrate that such a construction is possible, we consider the case of the non-uniform cubic B-spline curve scheme of the previous section.

In this case, the curve scheme may be expressed as a vector-valued subdivision scheme using the ideas of [10]. The basic idea behind vector-valued subdivision is to treat the initial control mesh p_0 as being a vector consisting of separate several distinct meshes. The subdivision process then mixes entries in the coarse vector via matrix multiplication to form each entry of the refined vector. This process can be modeled quite succinctly using generating functions. If $p_0[z]$ is a vector consisting of m generating functions, the subdivision mask for a vector subdivision scheme is an $m \times m$ matrix of generating functions $s[z]$. The vector-valued subdivision process is then modeled by the standard recurrence $p_{j+1}[z] = s[z]p_j[z^2]$.

For our second example, we can model our non-uniform curve scheme as a vector-valued uniform curve scheme. In this framework, $p_j[z]$ is a vector of five generating functions. The matrix mask $s[z]$ associated with our vector-valued scheme has the form

$$\begin{pmatrix} b[z] & \frac{25}{32} + \frac{1}{2z} + \frac{1}{8z^2} & \frac{3}{32z^2} & 0 & 0 \\ 0 & \frac{5z}{8} & \frac{3}{8z} & 0 & 0 \\ 0 & \frac{5z^2}{24} & \frac{29}{40} & \frac{1}{15z^2} & 0 \\ 0 & 0 & \frac{3z}{5} & \frac{2}{5z} & 0 \\ 0 & 0 & \frac{3z^2}{20} & \frac{z}{2} + \frac{29}{40} + \frac{1}{8z^2} & b[z] \end{pmatrix}$$

where $b[z] = \frac{z^2}{8} + \frac{z}{2} + \frac{3}{4} + \frac{1}{2z} + \frac{1}{8z^2}$. Given this uniform scheme, we can now compute the exact values of the scaling functions $\phi_{-2}[x]$, $\phi_{-1}[x]$, $\phi_0[x]$, $\phi_1[x]$, and $\phi_2[x]$ on Z. To this end, we simply construct the subdivision matrix S associated with this scheme and compute its dominant left eigenvector. In this case, the matrix S is a block matrix whose entries are the subdivision matrices S_{ij} corresponding to the entries $s_{ij}[z]$ of the matrix subdivision mask $s[z]$. If each of these block is chosen to be of size 3×3 (due to the support of the scaling functions), the appropriate finite portion of S is

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & \frac{1}{8} & \frac{25}{32} & 0 & \frac{3}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{5}{8} & 0 & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{5}{8} & 0 & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{5}{24} & 0 & \frac{29}{40} & 0 & \frac{1}{15} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{5} & 0 & \frac{2}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{5} & 0 & \frac{2}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{20} & 0 & \frac{29}{40} & \frac{1}{8} & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Now, the dominant left eigenvector for this matrix is a multiple of the block vector containing the exact values of the functions $\phi_h[x]$ for $h = -2, \ldots, 2$ and $x = -1, 0, 1$. Enforcing the partition of unity constraints on this eigenvector yields the desired exact values.

In general, we compute the exact values for our non-uniform curve scheme on the grid $\frac{1}{n}\mathsf{Z}$ by simply construct-

ing the upsampled matrix subdivision mask $\hat{s}[z] = s[z^n]$ and computing the dominant left eigenvectors of its corresponding upsampled subdivision matrix $\hat{S}$ (i.e; a block matrix whose entries are subdivision matrices corresponding to the mask $s_{ij}[z^n]$). Applying the partition of unity constraints to these left eigenvectors yields the desired exact values.

4 Exact evaluation stencils for non-polynomial surface schemes

So far we have only considered exact evaluation for curve subdivision schemes. But subdivision surfaces are far more prevalent in graphics and we would like to extend the evaluation techniques for curves to non-polynomial subdivision surfaces as well. In theory, the upsampling method presented in Section 3 can be extended to the bivariate case. However, the size of the matrix becomes large quickly and, unlike curves, there is no simple ordering of the basis functions that produces the upsampled matrix structure that was so readily visible with curves.

In this section, we explain how to modify the construction in Section 2 for surfaces. More importantly, we pay special attention to extraordinary vertices (valence $\neq 4$ for quad schemes or valence $\neq 6$ for triangle schemes). We then show how we can use this exact evaluation method to create a fast, adaptive tessellation of quadrilateral subdivision surfaces.

4.1 Uniform Case

In general, surface subdivision schemes are very similar to curve subdivision schemes except that the basis function $\phi[x]$ is now parameterized by a bivariate parameter x that represents a vector of two numbers. Equations 3 and 4 are identical except that the summations are over the 2D grid $i \in \mathsf{Z} \times \mathsf{Z}$. Evaluating the equations on the grid $[-m, m] \times [-m, m]$ at intervals of $\frac{1}{n}$ yields a finite set of equations whose solution is the value of the basis function over that grid.

This method works well for most uniform subdivision schemes, however, some subdivision schemes such as $\sqrt{3}$ introduce a rotation into the uniform grid complicating the parameterization. As long as the change in parameterization can be encoded in the right-hand side of Equation 3, the method presented here will still work. An easier solution is to realize that, after two levels of subdivision, $\sqrt{3}$ becomes a ternary subdivision scheme and aligns with the primal grid again simplifying the parameterization. Figure 3 shows the exact evaluation stencil for $\sqrt{3}$ subdivision at the midpoint of an edge computed using this technique. Notice that because two rounds of $\sqrt{3}$-subdivision produces a ternary subdivision scheme, no vertex will ever lie at this position for any finite level of subdivision. However we can still solve for the exact evaluation stencil despite the fact that the surface is non-polynomial.

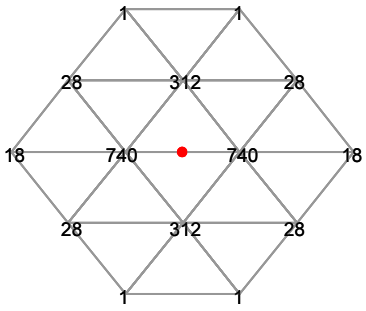

Figure 3. The exact evaluation stencil for the midpoint of an edge for the $\sqrt{3}$ subdivision scheme (with an implicit normalization of 2256).

4.2 Extraordinary vertices

Surfaces with extraordinary vertices behave similar to the non-uniform curve case in Section 2 in that we have special rules in a small region around the extraordinary vertex. Also, similar to non-uniform curves, we will have different basis functions corresponding to different vertices to account for the modified subdivision rules in the vicinity of the extraordinary vertex.

Like Stam's exact evaluation method [14], we will assume that we have a surface whose extraordinary vertices are sufficiently separated (no other extraordinary vertices in the $(m-1)$-ring of an extraordinary vertex). For subdivision schemes with basis functions supported over the two-ring, this means that each quad of the surface can contain only one extraordinary vertex. Under this assumption, the m-ring of an extraordinary vertex can be radially parameterized by the sector number k as well as a uniform, bivariate parameter x. Therefore, the basis functions are of the form $\phi_h[k, x]$. Furthermore, we do not use a shifted parameterization as for curves in Equation 3 but use a global parameterization in the vicinity of the extraordinary vertex. This parameterization then leads to the basis function refinement rules

$$\phi_h[k, x] = \sum_{j=0}^{\infty} \sum_{r} s_{h,j,r} \phi_j[k - r, 2x].$$

where $\phi_h[k, x]$ is the h^{th} basis function associated with the h^{th} vertex and $s_{h,j,r}$ encodes the coefficients in the h^{th} column of the subdivision matrix. Notice that we can reduce the number of basis functions in the equation by noting that

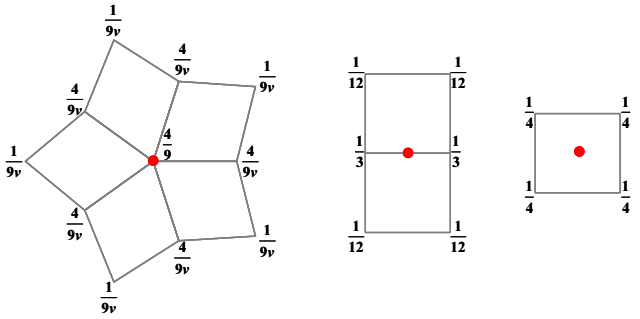

Figure 4. Subdivision rules for our non-polynomial subdivision scheme at vertices of valence v.

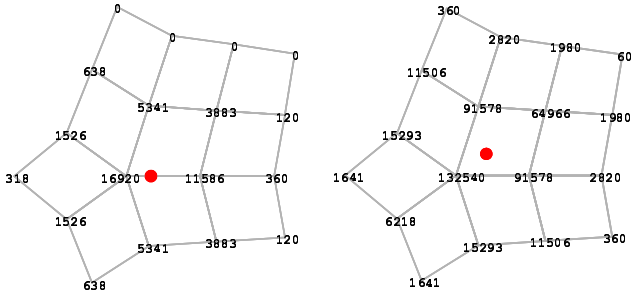

Figure 5. Exact evaluation stencils for our non-polynomial subdivision scheme at $x = \left(\frac{1}{3}, 0\right)$ and $x = \left(\frac{1}{3}, \frac{1}{3}\right)$.

the functions are only non-zero in their m-ring and that basis functions outside the m-ring of the extraordinary vertex are simply translates of basis functions on the edge of the m-ring yielding a finite summation. Unfortunately, these equations are not sufficient to uniquely determine the values of the basis functions but, like curve subdivision, convergent surface subdivisions schemes must also have the property that the basis functions at a single point sum to 1. This property adds the additional constraint that

$$\sum_{j=0}^{\infty} \sum_{r} \phi_j[k-r, x] = 1.$$

To find the exact value of the basis functions, we simply evaluate these equations on the grid $[0, m^2] \times [0, m^2]$ at intervals of $\frac{1}{n}$ producing a finite set of equations in a finite number of variables. The solution to this system of equations is the exact values of the different basis functions.

4.3 A non-polynomial quad subdivision example

To illustrate our algorithm, we will construct a non-polynomial, approximating subdivision scheme for quadrilateral surfaces whose basis functions are supported over the two-ring. In the ordinary case (valence $= 4$), the subdivision scheme will be the tensor-product of a non-polynomial curve subdivision scheme with the rules

$$\begin{array}{rcl} p_{k+1}^{2i} & = & \frac{1}{6}p_k^{i-1} + \frac{2}{3}p_k^i + \frac{1}{6}p_k^{i+1} \\ p_{k+1}^{2i+1} & = & \frac{1}{2}p_k^i + \frac{1}{2}p_k^{i+1} \end{array}$$

The eigenvalues of this curve subdivision scheme are of the form $1, \frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, 0, \ldots$ and the resulting curves are C^1 with bounded curvature but have no piecewise polynomial representation. We then generalize these rules to the extraordinary case resulting in the surface subdivision rules shown in Figure 4.

Like the corresponding curve subdivision scheme, this surface subdivision scheme creates surfaces that are C^1 everywhere, non-polynomial and have bounded curvature everywhere except at extraordinary vertices. We have also verified the smoothness of the subdivision scheme at extraordinary vertices by analyzing the eigenvalues and eigenvectors using the techniques of Reif [12] and Levin and Levin [8] but omit the details here for the sake of brevity. All of the surfaces in this paper were generated using this non-polynomial subdivision scheme.

Notice that, in general, this subdivision scheme is strictly worse than Catmull-Clark subdivision [2] (which produces C^2 surfaces almost everywhere) and would not be used in practice. Other subdivision schemes such as Kobbelt's interpolatory quad scheme [6] or even butterfly subdivision [18] have no polynomial representation. However, their basis functions are supported over the three-ring. This does not cause problems for our method, but the increased support means larger equations and more complexity. For the purposes of illustration, we have opted for a simpler, non-polynomial subdivision scheme.

If we naively attempt to solve the equations from Section 4.2 to perform exact evaluation, the number of variables produced is extremely large. For our example, a valence 6 vertex with $n = 3$ yields a system of equations in 781 variables. However, much of this work is unnecessary. Many of these equations redundantly solve for the ordinary basis function. By taking advantage of the fact that the basis functions in the two-ring are actually ordinary and that the one-ring basis functions become ordinary outside the one-ring, we can reduce the number of equations dramatically. Since we know the value of the basis functions outside the one-ring, we can also reduce the grid that we evaluate the equations on to $[0, 1] \times [0, 1]$. Finally, the basis functions will express a reflection symmetry because the subdivision rules are rotationally symmetric reducing the equations further. Combining these methods produces a drastically re-

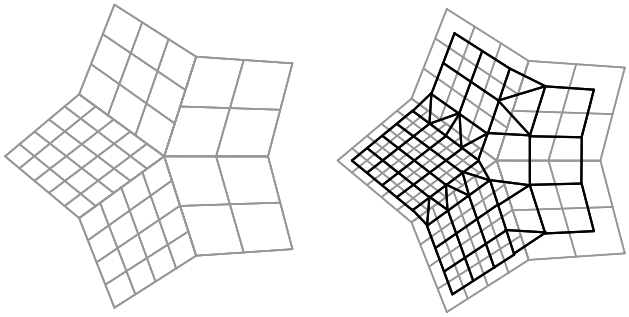

Figure 6. Left: uniform sampling grids generated by exact evaluation on rational grids do not necessarily align along their boundaries. Right: our adaptive tessellation takes the dual of the mesh and produces watertight tessellations.

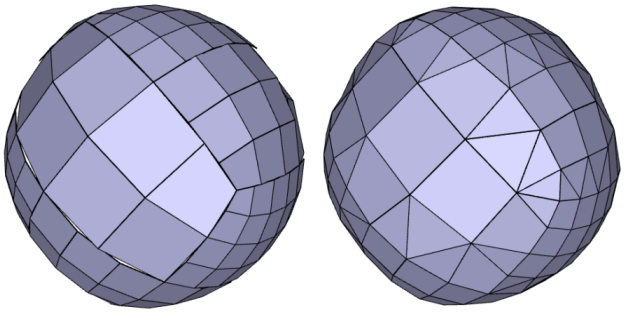

Figure 7. Left: gaps in surface due to different sampling rates. Right: closed, adaptive tessellation using our algorithm.

duced system of equations and, for our valence 6 example with $n = 3$, produces only 51 variables (as opposed to 781 before). Figure 5 shows two exact evaluation stencils for our example subdivision scheme at valence 5 with $n = 3$. Notice that there is an implied normalization that the entries of stencils sum up to 1.

4.4 Adaptive Tessellation of Quad-based subdivision surfaces

Using the exact evaluation technique from Section 4 we can develop a very efficient adaptive polygonalization technique for arbitrary subdivision surfaces (polynomial or non-polynomial). Previous work in adaptive tessellation of subdivision surfaces has largely concentrated on polynomial subdivision schemes. Both Shiue et al. [13] and Bolz et al. [1] focus on adaptive tessellation of subdivision surfaces using the GPU but take very different approaches. Shiue et al. perform subdivision directly on the GPU by encoding two-ring neighborhoods in a spiral fashion. Since the authors are performing subdivision the density of the surface increases exponentially with the level of subdivision. Using this method the authors achieve a tessellation rate of about 2 million triangles/second for a Catmull Clark surface (though later results and a more recent GPU indicate tessellation rates of about 7 million triangles/second).

Bolz et al. take a different approach to adaptive tessellation of subdivision surfaces. Instead of performing subdivision dynamically, the authors precompute samples of the basis functions for each valence and then apply these basis functions to the control points inside the GPU. These basis functions are precomputed using subdivision and, hence, the grid sizes increase exponentially as in [13]. The authors also allow for adaptive tessellation by using different subdivision levels on each patch of the surface. To fill the gaps in the surface caused by different levels of subdivision, the authors simply fan triangles to the neighboring vertices and achieve an impressive tessellation rate of about 24 million vertices/second using the GPU.

Both of these methods also pay special attention to numerical errors because the fact that the same vertex will be created multiple times from different patches possibly producing gaps in the surface. In contrast, we provide an adaptive tessellation technique based on our exact evaluation scheme that does not have to pay special attention to creating a watertight surface due to inaccurate floating point operations. We also do not restrict the tessellation rates of neighboring patches (for example, enforcing that the tessellation rates can only differ by one level) and always guarantee that our surfaces are closed. Furthermore, we can sample the subdivision surface on any rational grid $\left(\frac{1}{n}\right)$ as opposed to the exponential grids $\left(\frac{1}{2^n}\right)$ created via subdivision leading to much finer granularity in the adaptivity and results in less polygons for the same degree of approximation of the surface.

Our adaptive tessellation routine is inspired by another adaptive tessellation technique for implicit surfaces called Dual Contouring [5]. Dual Contouring generates adaptive, watertight tessellations of implicit surfaces defined over octrees by creating the topological dual of edges of the octree crossed by the implicit surface. We take a similar approach for surfaces and note that simply performing exact evaluation for polygons using different grid sizes produces a mesh with many gaps along the edges (see Figure 7). Instead of using these grids as the polygons, we take the dual of this mesh placing vertices at the centroid of each quad to create our adaptive tessellation. The dual of a uniform quadrilateral grid is simply another quadrilateral grid. Along the shared edges between patches, grid vertices that do not align between patches create triangles while coincident vertices create quads. Finally, at the vertices of the mesh, we create

Figure 8. View dependent tessellation using rational sampling (top left) and exponential, subdivision sampling (top right). An alternate view of the same tessellation (bottom).

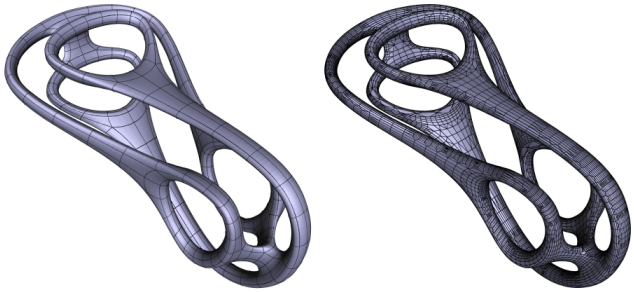

Figure 9. Left: patch structure. Right: adaptive tessellation of a non-polynomial subdivision surface.

a v-gon surrounding the vertex of valence v. Figures 6 and 7 show examples of tessellations produced by this technique. Notice that, since each vertex is created exactly once, we do not have to worry about floating point errors and we always obtain a watertight tessellation of the surface. Zorin and Schroder [17] also developed a related method for adaptively tessellating Doo-Sabin surfaces in the case of grids restricted to subdivision sampling.

Our adaptive tessellation technique works well for quadrilateral surfaces because the dual of a quadrilateral grid is also quadrilateral. If triangle subdivision schemes are desired, the same process can be used. However, the dual of a regular triangle grid is a hexagonal tiling. Therefore, these hexagons must be triangulated before display. We suggest adding a new vertex at the center of the dual polygons and creating a triangle fan to that vertex from the edges. This technique produces a regular grid of triangles in ordinary regions and always creates a watertight tessellation.

5 Implementation and Results

In terms of implementation, we use the technique from Section 4 to pre-compute an exact evaluation table for a restricted set of valences (we use 3-8) at different sampling rates (Figure 5 shows two example entries in the table). These tables are quite small and take up about 2.5 MB of space for $n = 1$-24 and valences 3-8. We also use a half-edge data structure for the base mesh to efficiently collect the vertices in the one-ring of each quad for exact evaluation.

When computing the dual of the evaluation grids to create a water-tight tessellation, we consider the faces, edges and vertices of the base mesh separately. Faces of the base mesh contain a uniform grid whose dual is another uniform grid and is trivial to compute. Along edges, we take the list of dual vertex indices from the two adjacent patches and perform a merge operation based on the two grid sizes to create the edge polygons, which is quite fast. Finally, we collect all of the dual vertex indices adjacent to each vertex for the polygons dual to the control mesh vertices.

Figures 1, 7, 8 and 9 show examples of adaptive tessellations produced by our method for subdivision surfaces that are non-polynomial everywhere. At runtime we compute the tessellation parameters for each patch and many different tessellation criteria could be used such as curvature. We calculate view-dependent tessellation rates by projecting each quad from the base subdivision surface onto the screen, computing its maximal edge length and dividing by the maximal number of pixels we would like edges on the adaptively tessellated surface to be. Using our tessellation method, we are able to achieve tessellation rates of over 33.5 million triangles/second on an Intel Core 2 6700 PC using the Intel compiler. Notice that, unlike other techniques, no GPU is being used to perform tessellation. We expect a GPU implementation to achieve even more dramatic performance gains.

Figure 8 compares our tessellation algorithm using rational sampling $\left(\frac{1}{n}\right)$ on the top left versus samples generated via subdivision on exponentially increasing grids $\left(\frac{1}{2^n}\right)$ on the top right using the same view dependent tessellation criteria. The bottom left and right of the figure show the same tessellations from another viewpoint. The changes in resolution are much more subtle with our tessellations than those produced by subdivision. Hence, our technique is able to achieve much finer granularity when approximating the surface and results in far fewer polygons. In this example, the method based on exponential grids requires over 40% more polygons than our method to achieve the same level of approximation.

6 Conclusions and Future Work

In this paper we have demonstrated that it is possible to evaluate *any* stationary subdivision scheme at rational values. Our technique was based on enumerating a system of equations relating the basis functions to one another using the subdivision matrix. Solving this system of equations provides the samples of the basis functions and leads to the exact evaluation stencils. While this process is somewhat expensive for large sampling grids, the stencils can be precomputed once and saved. We then showed how we could use this exact evaluation method to create adaptive tessellations of quadrilateral subdivision surfaces, which led to impressive tessellation rates.

There are several open problems we would like to consider in the future. One is to attempt to extend the upsampling method in Section 3 from curves to surfaces so that we do not have to manually enumerate the equations for the scaling relationship. Unfortunately, the parameterization of surfaces with extraordinary vertices is complicated and we have not created a simple technique for doing so yet.

While all of the subdivision schemes we tried provided a unique evaluation stencil from our equations, we currently have no proof of uniqueness. In the future, we would like to develop such a proof. One might expect problems to occur when evaluating more degenerate subdivision schemes such as linearly dependent subdivision schemes, but our experiments linearly dependent subdivision schemes did not produce any difficulties and the equations still had unique solutions.

Finally, we intend to explore the idea of using precomputed stencils to facilitate modeling using linear surfaces schemes that have C^2 smoothness. Typically, such schemes require complicated basis functions with support much larger than that of subdivision schemes such as Catmull-Clark. Our idea is to isolate the complexity of the evaluation of the basis functions in a pre-computation phase in which stencils for various values of n and various local neighborhood topologies are constructed. Given these stencils, our task is to then develop a mesh indexing scheme that characterizes the topological structure of the local neighborhood of a quad (or triangle) allowing for fast application of the appropriate stencil to perform exact evaluation.

Acknowledgements

We would like to thank Bay Raitt for the models of the "Monster Frog" and "Big Guy" as well as Ergun Akleman for the shape in Figure 9.

References

[1] J. Bolz and P. Schröder. Evaluation of subdivision surfaces on programmable graphics hardware.

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

[3] N. Dyn, J. Gregory, and D. Levin. A four point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.

[4] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. *Computer Graphics*, 27(Annual Conference Series):35–44, 1993.

[5] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, 2002.

[6] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Computer Graphics Forum (Proc. EUROGRAPHICS '96), 15(3)*, pages 409–420, 1996.

[7] L. Kobbelt. sqrt(3)-subdivision. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112, 2000.

[8] A. Levin and D. Levin. Analysis of quasi uniform subdivision. *Applied and Computational Harmonic Analysis*, 15(1):18–32, 2003.

[9] C. Loop. Smooth subdivision surfaces based on triangles. *Masters Thesis., University of Utah, Dept. of Mathematics*, 1987.

[10] C. Micchelli and J. Sauer. On vector subdivision. *Math. Z.*, 229:621–674, 1998.

[11] L. Ramshaw. Blossoms are polar forms. *Computer Aided Geometric Design*, 6(4):323–358, 1989.

[12] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[13] L.-J. Shiue, I. Jones, and J. Peters. A realtime gpu subdivision kernel. *ACM Trans. Graph.*, 24(3):1010–1015, 2005.

[14] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 395–404, 1998.

[15] G. Strang. Wavelets and dilation equations: a brief introduction. *SIAM Review*, 31(4):614–627, 1989.

[16] J. Warren and H. Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[17] D. Zorin and P. Schröder. Subdivision for modeling and animation, 2000.

[18] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192, 1996.