Exact Evaluation Of Catmull-Clark Subdivision Surfaces At Arbitrary Parameter Values

Jos Stam

Alias|wavefront, Inc.
1218 Third Ave, 8th Floor,
Seattle, WA 98101, U.S.A.
`jstam@aw.sgi.com`

Abstract

In this paper we disprove the belief widespread within the computer graphics community that Catmull-Clark subdivision surfaces cannot be evaluated directly without explicitly subdividing. We show that the surface and all its derivatives can be evaluated in terms of a set of *eigenbasis* functions which depend only on the subdivision scheme and we derive analytical expressions for these basis functions. In particular, on the regular part of the control mesh where Catmull-Clark surfaces are bi-cubic B-splines, the eigenbasis is equal to the power basis. Also, our technique is both efficient and easy to implement. We have used our implementation to compute high quality curvature plots of subdivision surfaces. The cost of our evaluation scheme is comparable to that of a bi-cubic spline. Therefore, our method allows many algorithms developed for parametric surfaces to be applied to Catmull-Clark subdivision surfaces. This makes subdivision surfaces an even more attractive tool for free-form surface modeling.

1 Introduction

Subdivision surfaces have emerged recently as a powerful and useful technique in modeling free-form surfaces. However, although in theory subdivision surfaces admit local parametrizations, there is a strong belief within the computer graphics community that these parametrizations cannot be evaluated exactly for arbitrary parameter values. In this paper we disprove this belief and provide a non-iterative technique that efficiently evaluates Catmull-Clark subdivision surfaces and their derivatives up to any order. The cost of our technique is comparable to the evaluation of a bi-cubic surface spline. The rapid and precise evaluation of surface parametrizations is crucial for many standard operations on surfaces such as picking, rendering and texture mapping. Our evaluation technique allows a large body of useful techniques from parametric surfaces to be transferred to subdivision surfaces, making them even more attractive as a free-form surface modeling tool.

Our evaluation is based on techniques first developed to prove smoothness theorems for subdivision schemes [3, 5, 1, 4, 8, 6]. These proofs are constructed by transforming the subdivision into its eigenspace[1]. In its eigenspace, the subdivision is equivalent to a simple scaling of each of its eigenvectors by their eigenvalue. These techniques allow us to compute limit points and limit normals at the vertices of the mesh, for example. Most of the proofs, however, consider only a subset of the entire eigenspace and do not address the problem of evaluating the surface everywhere. We, on the other hand, use the entire eigenspace to derive an efficiently evaluated analytical form of the subdivision surface everywhere, even in the neighborhood of extraordinary vertices. In this way, we have extended a theoretical tool into a very practical one.

In this paper we present an evaluation scheme for Catmull-Clark subdivision surfaces [2]. However, our methodology is not limited to these surfaces. Whenever subdivision on the regular part of the mesh coincides with a known parametric representation [8], our approach should be applicable. We have decided to present the technique for the special case of Catmull-Clark subdivision surfaces in order to show a particular example fully worked out. In fact, we have implemented a similar technique for Loop's triangular subdivision scheme [5]. The details of that scheme are given in another paper in these course notes. We believe that Catmull-Clark surfaces have many properties which make them attractive as a free-form surface design tool. For example, after one subdivision step each face of the initial mesh is a quadrilateral, and on the regular part of the mesh the surface is equivalent to a piecewise uniform B-spline. Also, algorithms have been written to fair these surfaces [4] and to dynamically animate them [7].

In order to define a parametrization, we introduce a new set of *eigenbasis functions*. These functions were first introduced by Warren in a theoretical setting for curves [10] and used in a more general setting by Zorin [11]. In this paper, we show that the eigenbasis of the Catmull-Clark subdivision scheme can be computed analytically. Also, we show that in the regular case the eigenbasis is equal to the power basis and that the eigenvectors then correspond to the "change of basis matrix" from the power basis to the bi-cubic B-spline basis. The eigenbasis introduced in this paper can thus be thought of as a generalization of the power basis at extraordinary vertices. Since our eigenbasis functions are analytical, the evaluation of Catmull-Clark subdivision surfaces can be expressed analytically. As shown in the results section of this paper, we have implemented our evaluation scheme and used it in many practical applications. In particular, we show for the first time high resolution curvature plots of Catmull-Clark surfaces precisely computed around the irregular parts of the mesh.

The paper is organized as follows. Section 2 is a brief review of the Catmull-Clark subdivision scheme. In Section 3 we cast this subdivision scheme into a mathematical setting suitable for analysis. In Section 4 we compute the eigenstructure to derive our evaluation. Section 5 is a discussion of implementation issues. In Section 6 we exhibit results created using our technique, comparing it to straightforward subdivision. Finally in Section 7 we conclude, mentioning promising directions for future research.

This paper is almost equivalent to our SIGGRAPH'98 paper [9]. We have corrected two errors in the Appendices and added a small section devoted to stability issues in Section 5. Also we have included on the CDROM course notes a data file which contains the eigenstructures up to valence 50.

[1] To be defined precisely below.

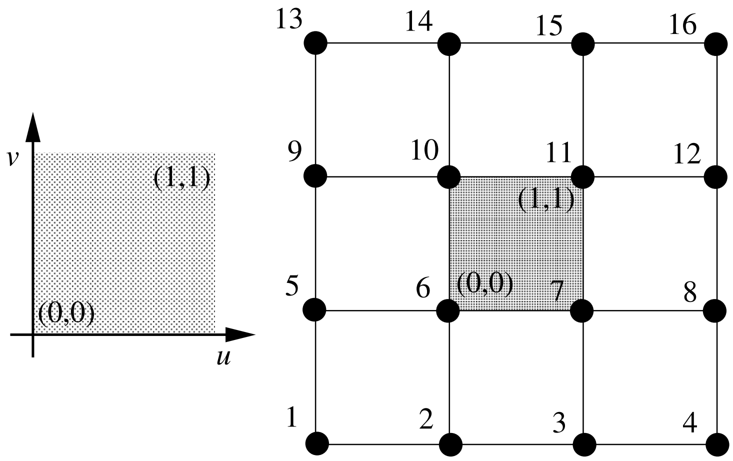

Figure 1: A bi-cubic B-spline is defined by 16 control vertices. The numbers on the right show the ordering of the corresponding B-spline basis functions in the vector $\mathbf{b}(u, v)$.

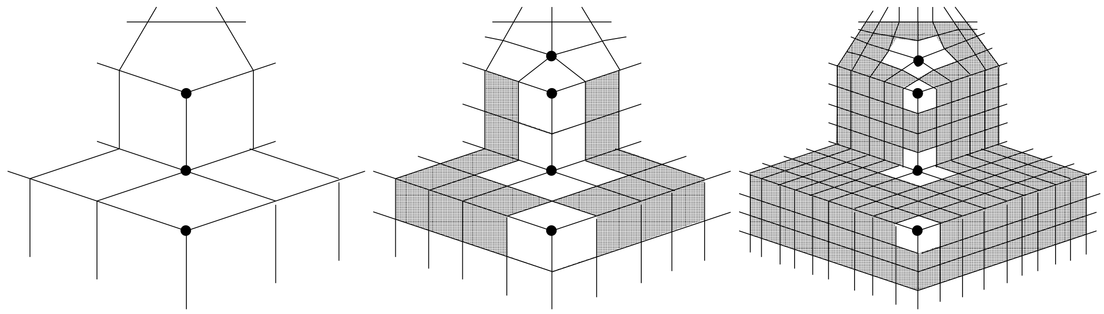

Figure 2: Initial mesh and two levels of subdivision. The shaded faces correspond to regular bi-cubic B-spline patches. The dots are extraordinary vertices.

1.1 Notations

In order to make the derivations below as clear and compact as possible we adopt the following notational conventions. All vectors are assumed to be columns and are denoted by boldface lower case roman characters, e.g., $\mathbf{v}$. The components of the vector are denoted by the corresponding italicized character: the i-th component of a vector $\mathbf{v}$ is thus denoted v_i. The component of a vector should not be confused with an indexed vector such as $\mathbf{v}_k$. Matrices are denoted by uppercase boldface characters, e.g., $\mathbf{M}$. The transpose of a vector $\mathbf{v}$ (resp. matrix $\mathbf{M}$) is denoted by $\mathbf{v}^T$ (resp. $\mathbf{M}^T$). The transpose of a vector is simply the same vector written row-wise. Therefore the dot product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is written "$\mathbf{u}^T\mathbf{v}$". The vector or matrix having only zero elements is denoted by $\mathbf{0}$. The size of this vector (matrix) should be obvious from the context.

2 Catmull-Clark Subdivision Surfaces

The Catmull-Clark subdivision scheme was designed to generalize uniform B-spline knot insertion to meshes of arbitrary topology [2]. An arbitrary mesh such as the one shown on the upper left hand side of Figure 2 is used to define a smooth surface. The surface is defined as the limit of a sequence of subdivision steps. At each step the vertices of the mesh are updated and new vertices are introduced. Figure 2 illustrates this process. On each vertex of the initial mesh, the *valence* is

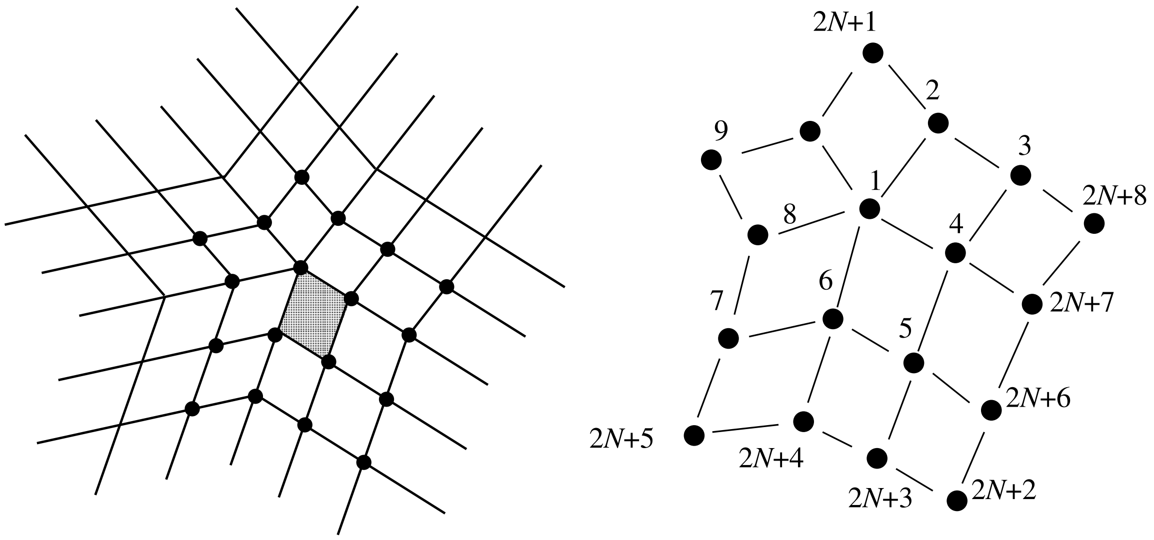

Figure 3: Surface patch near an extraordinary vertex with its control vertices. The ordering of the control vertices is shown on the bottom. Vertex 1 is an extraordinary vertex of valence $N = 5$.

the number of edges that meet at the vertex. A vertex having a valence not equal to four is called an *extraordinary vertex*. The mesh on the upper left hand side of Figure 2 has two extraordinary vertices of valence three and one of valence five. Away from extraordinary vertices, the Catmull-Clark subdivision is equivalent to midpoint uniform B-spline knot insertion. Therefore, the 16 vertices surrounding a face that contains no extraordinary vertices are the control vertices of a uniform bi-cubic B-spline patch (shown schematically in Figure 1). The faces which correspond to a regular patch are shaded in Figure 2. This figure shows how the portion of the surface comprised of regular patches grows with each subdivision step. In principle, the surface can thus be evaluated whenever the holes surrounding the extraordinary vertices are sufficiently small. Unfortunately, this iterative approach is too expensive near extraordinary vertices and does not provide exact higher derivatives.

Because the control vertex structure near an extraordinary vertex is not a simple rectangular grid, all faces that contain extraordinary vertices cannot be evaluated as uniform B-splines. We assume that the initial mesh has been subdivided at least twice, isolating the extraordinary vertices so that each face is a quadrilateral and contains at most one extraordinary vertex. In the rest of the paper, we need to demonstrate only how to evaluate a patch corresponding to a face with just one extraordinary vertex, such as the region near vertex 1 in Figure 3. Let us denote the valence of that extraordinary vertex by N. Our task is then to find a surface patch $\mathbf{s}(u, v)$ defined over the unit square $\Omega = [0, 1] \times [0, 1]$ that can be evaluated directly in terms of the $K = 2N + 8$ vertices that influence the shape of the patch corresponding to the face. We assume in the following that the surface point corresponding to the extraordinary vertex is $\mathbf{s}(0, 0)$ and that the orientation of Ω is chosen such that $\mathbf{s}_u \times \mathbf{s}_v$ points outside of the surface.

A simple argument shows that the influence on the limit surface of the seven "outer control vertices" numbered $2N + 2$ through $2N + 8$ in Figure 3 can be accounted for directly. Indeed, consider the situation depicted in Figure 4 where we show a mesh containing a vertex of valence 5 and a regular mesh side by side. Let us assume that all the control vertices are set to zero except for the seven control vertices highlighted in Figure 4. If we repeat the Catmull-Clark subdivision rules for both meshes we actually obtain the same limit surface, since the exceptional control vertex

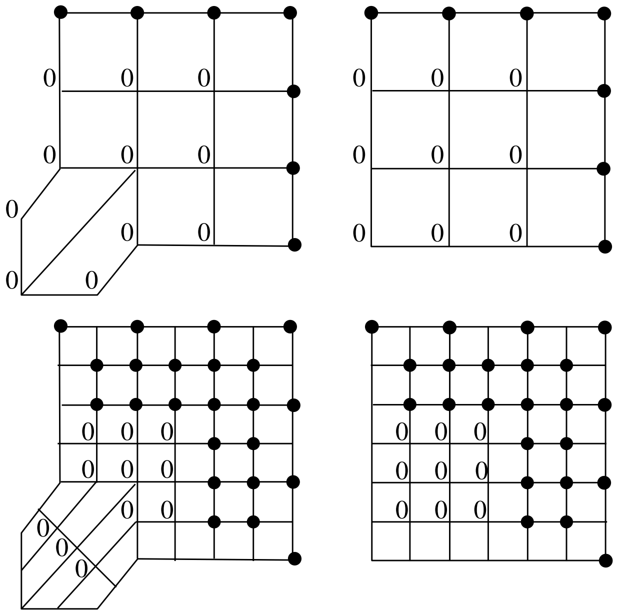

Figure 4: The effect of the seven outer control vertices does not depend on the valence of the extraordinary vertex. When the $2N + 1$ control vertices in the center are set to zero the same limit surface is obtained.

at the center of the patch remains equal to zero after each subdivision step. Therefore, the effect of the seven outer control vertices is simply each control vertex multiplied by its corresponding bi-cubic B-spline tensor product basis function. In the derivation of our evaluation technique we do not need to make use of this fact. However, it explains the simplifications which occur at the end of the derivation.

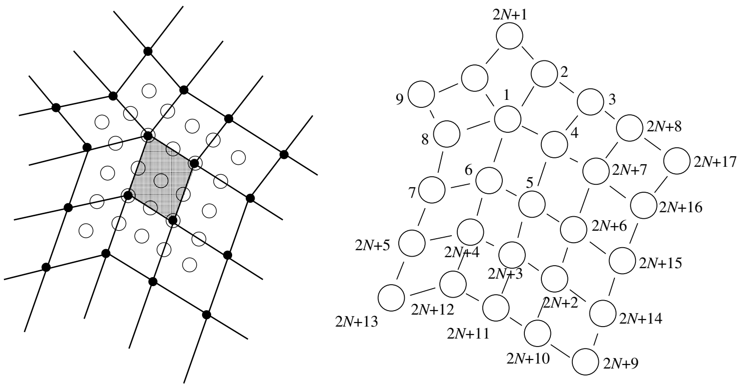

Figure 5: Addition of new vertices by applying the Catmull-Clark subdivision rule to the vertices in Figure 3.

3 Mathematical Setting

In this section we cast the informal description of the previous section into a rigorous mathematical setting. We denote by

$$\mathbf{C}_0^T = (\mathbf{c}_{0,1}, \cdots, \mathbf{c}_{0,K}),$$

the initial control vertices defining the surface patch shown in Figure 3. The ordering of these vertices is defined on the bottom of Figure 3. This peculiar ordering is chosen so that later computations become more tractable. Note that the vertices do not result in the 16 control vertices of a uniform bi-cubic B-spline patch, except when $N = 4$.

Through subdivision we can generate a new set of $M = K + 9$ vertices shown as circles superimposed on the initial vertices in Figure 5. Subsets of these new vertices are the control vertices of three uniform B-spline patches. Therefore, three-quarters of our surface patch is parametrized, and could be evaluated as simple bi-cubic B-splines (see top left of Figure 6). We denote this new set of vertices by

$$\mathbf{C}_1^T = (\mathbf{c}_{1,1}, \cdots, \mathbf{c}_{1,K}) \text{ and } \bar{\mathbf{C}}_1^T = \left(\mathbf{C}_1^T, \mathbf{c}_{1,K+1}, \cdots, \mathbf{c}_{1,M}\right).$$

With these matrices, the subdivision step is a multiplication by an $K \times K$ *(extended) subdivision matrix* $\mathbf{A}$:

$$\mathbf{C}_1 = \mathbf{A}\mathbf{C}_0. \tag{1}$$

Due to the peculiar ordering that we have chosen for the vertices, the extended subdivision matrix has the following block structure:

$$\mathbf{A} = \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{S}_{11} & \mathbf{S}_{12} \end{pmatrix}, \tag{2}$$

where $\mathbf{S}$ is the $2N+1 \times 2N+1$ subdivision matrix usually found in the literature [4]. The remaining two matrices correspond to the regular midpoint knot insertion rules for B-splines. Their exact definition can be found in Appendix A. The additional points needed to evaluate the three B-spline patches are defined using a bigger matrix $\bar{\mathbf{A}}$ of size $M \times K$:

$$\bar{\mathbf{C}}_1 = \bar{\mathbf{A}}\mathbf{C}_0,$$

where

$$\bar{\mathbf{A}} = \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{pmatrix}. \tag{3}$$

The matrices $\mathbf{S}_{21}$ and $\mathbf{S}_{22}$ are defined in Appendix A. The subdivision step of Equation 1 can be repeated to create an infinite sequence of control vertices:

$$\begin{aligned} \mathbf{C}_n &= \mathbf{A}\mathbf{C}_{n-1} = \mathbf{A}^n\mathbf{C}_0 \text{ and} \\ \bar{\mathbf{C}}_n &= \bar{\mathbf{A}}\mathbf{C}_{n-1} = \bar{\mathbf{A}}\mathbf{A}^{n-1}\mathbf{C}_0, \quad n \geq 1. \end{aligned}$$

As noted above, for each level $n \geq 1$, a subset of the vertices of $\bar{\mathbf{C}}_n$ becomes the control vertices of three B-spline patches. These control vertices can be defined by selecting 16 control vertices from $\bar{\mathbf{C}}_n$ and storing them in 16×3 matrices:

$$\mathbf{B}_{k,n} = \mathbf{P}_k\bar{\mathbf{C}}_n,$$

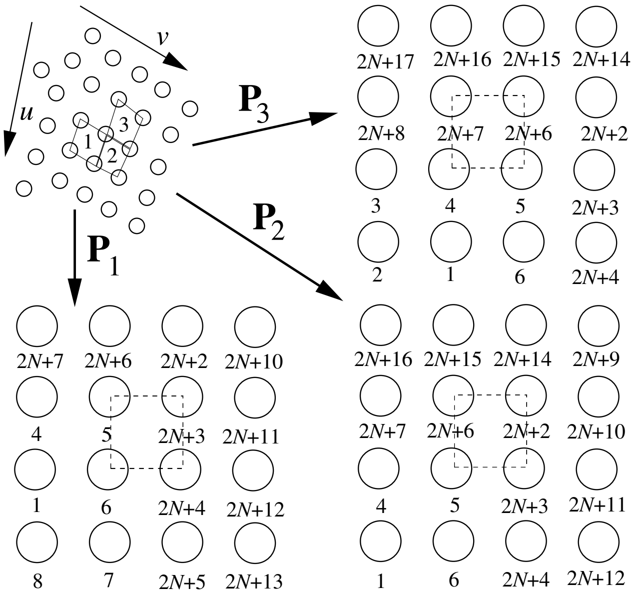

Figure 6: Indices of the control vertices of the three bi-cubic B-spline patches obtained from $\bar{\mathbf{C}}_n$.

where $\mathbf{P}_k$ is a $16 \times M$ "picking" matrix and $k = 1, 2, 3$. Let $\mathbf{b}(u, v)$ be the vector containing the 16 cubic B-spline basis functions (see Appendix B). If the control vertices are ordered as shown on the left of Figure 1, then the surface patch corresponding to each matrix of control vertices is defined as

$$\mathbf{s}_{k,n}(u, v) = \mathbf{B}_{k,n}^T \mathbf{b}(u, v) = \bar{\mathbf{C}}_n^T \mathbf{P}_k^T \mathbf{b}(u, v), \tag{4}$$

where $(u, v) \in \Omega$, $n \geq 1$ and $k = 1, 2, 3$. Using the ordering convention for the B-spline control vertices of Figure 1, the definition of the picking matrices is shown in Figure 6. Each row of $\mathbf{P}_k$ is filled with zeros except for a one in the column corresponding to the index shown in Figure 6 (see Appendix B for more details). The infinite sequence of uniform B-spline patches defined by Equation 4 form our surface $\mathbf{s}(u, v)$, when "stitched together". More formally, let us partition the unit square Ω into an infinite set of tiles $\{\Omega_k^n\}$, $n \geq 1, k = 1, 2, 3$, as shown in Figure 7. Each tile with index n is four times smaller than the tiles with index $n - 1$. More precisely:

$$\begin{aligned} \Omega_1^n &= \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right] \times \left[0, \frac{1}{2^n}\right], \\ \Omega_2^n &= \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right] \times \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right], \\ \Omega_3^n &= \left[0, \frac{1}{2^n}\right] \times \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right]. \end{aligned} \tag{5}$$

A parametrization for $\mathbf{s}(u, v)$ is constructed by defining its restriction to each tile Ω_k^n to be equal to the B-spline patch defined by the control vertices $\mathbf{B}_{k,n}$:

$$\mathbf{s}(u, v)\,\big|_{\Omega_k^n} = \mathbf{s}_{k,n}(\mathbf{t}_{k,n}(u, v)). \tag{6}$$

The transformation $\mathbf{t}_{k,n}$ maps the tile Ω_k^n onto the unit square Ω:

$$\mathbf{t}_{1,n}(u, v) = (2^n u - 1, 2^n v), \tag{7}$$

$$\mathbf{t}_{2,n}(u, v) = (2^n u - 1, 2^n v - 1) \quad \text{and} \tag{8}$$

$$\mathbf{t}_{3,n}(u, v) = (2^n u, 2^n v - 1). \tag{9}$$

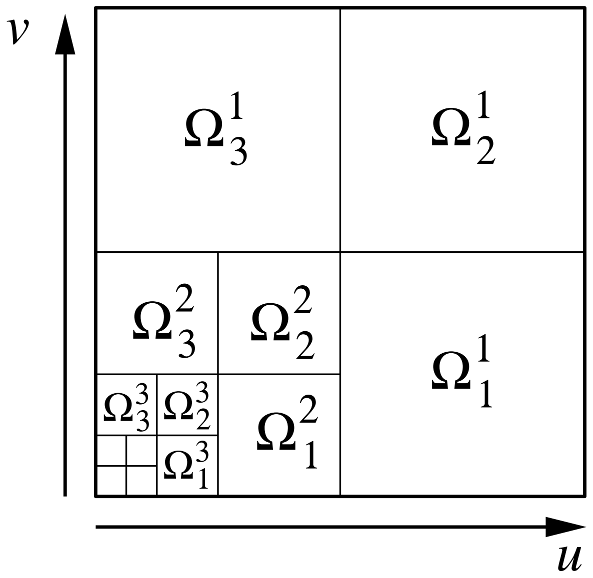

Figure 7: Partition of the unit square into an infinite family of tiles.

Equation 6 gives an actual parametrization for the surface. However, it is very costly to evaluate, since it involves $n-1$ multiplications of the $K \times K$ matrix $\mathbf{A}$. The evaluation can be simplified considerably by computing the eigenstructure of $\mathbf{A}$. This is the key idea behind our new evaluation technique and is the topic of the next section.

4 Eigenstructure, Eigenbases and Evaluation

The eigenstructure of the subdivision matrix $\mathbf{A}$ is defined as the set of its eigenvalues and eigenvectors. In our case the matrix $\mathbf{A}$ is non-defective for any valence. Consequently, there always exists K linearly independent eigenvectors [4]. Therefore we denote this eigenstructure by $(\mathbf{\Lambda}, \mathbf{V})$, where $\mathbf{\Lambda}$ is the diagonal matrix containing the eigenvalues of $\mathbf{A}$, and $\mathbf{V}$ is an invertible matrix whose columns are the corresponding eigenvectors. The computation of the eigenstructure is then equivalent to the solution of the following matrix equation:

$$\mathbf{AV} = \mathbf{V\Lambda}, \tag{10}$$

where the i-th diagonal element of $\mathbf{\Lambda}$ is an eigenvalue with a corresponding eigenvector equal to the i-th column of the matrix $\mathbf{V}$ $(i = 1, \cdots, K)$. There are many numerical algorithms which can compute solutions for such equations. Unfortunately for our purposes, these numerical routines do not always return the correct eigenstructure. For example, in some cases the solver returns complex eigenvalues. For this reason, we must explicitly compute the eigenstructure. Since the subdivision matrix has a definite block structure, our computation can be done in several steps. In Appendix A we analytically compute the eigenstructure $(\mathbf{\Sigma}, \mathbf{U}_0)$ (resp. $(\mathbf{\Delta}, \mathbf{W}_1)$) of the diagonal block $\mathbf{S}$ (resp. $\mathbf{S}_{12}$) of the subdivision matrix defined in Equation 2. The eigenvalues of the subdivision matrix are the union of the eigenvalues of its diagonal blocks:

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Delta} \end{pmatrix}.$$

Using the eigenvectors of $\mathbf{S}$ and $\mathbf{S}_{12}$, it can be proven that the eigenvectors for the subdivision matrix must have the following form:

$$\mathbf{V} = \begin{pmatrix} \mathbf{U}_0 & \mathbf{0} \\ \mathbf{U}_1 & \mathbf{W}_1 \end{pmatrix}.$$

The matrix $\mathbf{U}_1$ is unknown and is determined from Equation 10. If we replace the matrices $\mathbf{\Lambda}$, $\mathbf{V}$ and $\mathbf{A}$ by their block representations, we obtain the following matrix equation:

$$\mathbf{S}_{11}\mathbf{U}_0 + \mathbf{S}_{12}\mathbf{U}_1 = \mathbf{U}_1\mathbf{\Sigma}. \tag{11}$$

Since $\mathbf{U}_0$ is known, $\mathbf{U}_1$ is computed by solving the $2N+1$ linear systems of Equation 11. In principle, this equation could be solved symbolically. In practice, however, because of the small sizes of the linear systems (7×7) we can compute the solution up to machine accuracy (see the next section for details). The inverse of our eigenvector matrix is equal to

$$\mathbf{V}^{-1} = \begin{pmatrix} \mathbf{U}_0^{-1} & \mathbf{0} \\ -\mathbf{W}_1^{-1}\mathbf{U}_1\mathbf{U}_0^{-1} & \mathbf{W}_1^{-1} \end{pmatrix}, \tag{12}$$

where both $\mathbf{U}_0$ and $\mathbf{W}_1$ can be inverted exactly (see Appendix A). This fact allows us to rewrite Equation 10:

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}.$$

This decomposition is the crucial result that we use in constructing a fast evaluation scheme of the surface patch. Indeed, the subdivided control vertices at level n are now equal to

$$\bar{\mathbf{C}}_n = \bar{\mathbf{A}}\mathbf{A}^{n-1}\mathbf{C}_0 = \bar{\mathbf{A}}\mathbf{V}\mathbf{\Lambda}^{n-1}\mathbf{V}^{-1}\mathbf{C}_0 = \bar{\mathbf{A}}\mathbf{V}\mathbf{\Lambda}^{n-1}\hat{\mathbf{C}}_0,$$

where $\hat{\mathbf{C}}_0 = \mathbf{V}^{-1}\mathbf{C}_0$ is the projection of the K control vertices into the eigenspace of the subdivision matrix. Using this new expression for the control vertices at the n-th level of subdivision, Equation 4 can be rewritten in the following form:

$$\mathbf{s}_{k,n}(u,v) = \hat{\mathbf{C}}_0^T\mathbf{\Lambda}^{n-1}\left(\mathbf{P}_k\bar{\mathbf{A}}\mathbf{V}\right)^T\mathbf{b}(u,v).$$

We observe that the right most terms in this equation are independent of the control vertices and the power n. Therefore, we can precompute this expression and define the following three vectors:

$$\mathbf{x}(u,v,k) = \left(\mathbf{P}_k\bar{\mathbf{A}}\mathbf{V}\right)^T\mathbf{b}(u,v) \quad k = 1,2,3. \tag{13}$$

The components of these three vectors correspond to a set of K bi-cubic splines. In Appendix B we show how to compute these splines. Notice that the splines $x_i(u,v,k)$ depend only on the valence of the extraordinary vertex. Consequently, we can rewrite the equation for each patch more compactly as:

$$\mathbf{s}_{k,n}(u,v) = \hat{\mathbf{C}}_0^T\mathbf{\Lambda}^{n-1}\mathbf{x}(u,v,k) \quad k = 1,2,3. \tag{14}$$

To make the expression for the evaluation of the surface patch more concrete, let $\mathbf{p}_i^T$ denote the rows of $\hat{\mathbf{C}}_0$. Then the surface patch can be evaluated as:

$$\mathbf{s}(u,v)\Big|_{\Omega_k^n} = \sum_{i=1}^{K}(\lambda_i)^{n-1}x_i(\mathbf{t}_{k,n}(u,v),k)\mathbf{p}_i. \tag{15}$$

Therefore, in order to evaluate the surface patch, we must first compute the new vertices $\mathbf{p}_i$ (only once for a given mesh). Next, for each evaluation we determine n and then scale the contribution from each of the splines by the relevant eigenvalue to the power $n-1$. Since all but the first of the eigenvalues are smaller than one, their contribution decreases as n increases. Thus, for large n, i.e., for surface-points near the extraordinary vertex, only a few terms make a significant contribution. In fact for $(u,v) = (0,0)$ the surface point is $\mathbf{p}_1$, which agrees with the definition of a limit point in [4].

Alternatively, the bi-cubic spline functions $\mathbf{x}(u,v,k)$ can be used to define a set of *eigenbasis functions* for the subdivision. For a given eigenvalue λ_i we define the function φ_i by its restrictions on the domains Ω_k^n as follows:

$$\varphi_i(u,v)\,\big|_{\Omega_k^n} = (\lambda_i)^{n-1} x_i(\mathbf{t}_{k,n}(u,v),k),$$

with $i = 1, \cdots, K$. By the above definition these functions satisfy the following scaling relation:

$$\varphi_i(u/2, v/2) = \lambda_i \varphi_i(u,v).$$

The importance of these functions was first noted by Warren in the context of subdivision curves [10]. More recently, Zorin has defined and used eigenbasis functions to prove smoothness conditions for very general classes of subdivision schemes [11]. However, explicit analytical expressions for particular eigenbases have never appeared before. On the other hand, we can compute these bases analytically. Figures 8 and 9 show the complete sets of eigenbasis functions for valences 3 and 5. In the figures we have normalized each function such that its range is bounded within -1 and 1. In particular, the first eigenbasis corresponding to an eigenvalue of one is always a constant function for any valence. A closer look at Figures 8 and 9 reveals that they share seven identical functions. In fact as shown in Appendix B, the last seven eigenbasis functions for any valence are always equal to

$$\left\{ \frac{1}{36}u^3v^3, \frac{1}{6}u^3, \frac{1}{6}u^3v, \frac{1}{2}u^3v^2, \frac{1}{6}v^3, \frac{1}{6}uv^3, \frac{1}{2}u^2v^3 \right\}.$$

Furthermore, by transforming these functions back from the eigenspace using $\mathbf{W}_1^{-1}$ we obtain the seven tensor B-spline basis functions

$$b_4(u,v), b_8(u,v), b_{12}(u,v), \cdots, b_{16}(u,v),$$

i.e., the basis functions corresponding to the "outer layer" of control vertices of Figure 3. This should not come as a surprise since as we noted above, the influence of the outer layer does not depend on the valence of the extraordinary vertex (see Figure 4).

In the regular bi-cubic B-spline case ($N = 4$), the remaining eigenbasis can be chosen to be equal to the power basis

$$\{1, u, v, u^2, uv, v^2, u^2v, uv^2, u^2v^2\}.$$

The scaling property of the power basis is obvious. For example, the basis function u^2v corresponds to the eigenvalue $1/8$:

$$(u/2)^2(v/2) = (1/2)^2(1/2)u^2v = \frac{1}{8}u^2v.$$

This relationship between the Catmull-Clark subdivision and the power basis in the regular case does not seem to have been noted before. Note also that the eigenvectors in this case correspond to the "change of basis matrix" from the bi-cubic B-spline basis to the power basis. The eigenbasis functions at extraordinary vertices can thus be interpreted as a generalization of the power basis. However, the eigenbases are in general not polynomials. In the case of the Catmull-Clark subdivision they are piece-wise bi-cubic polynomials. The evaluation of the surface patch given by Equation 15 can now be rewritten exactly as:

$$\mathbf{s}(u,v) = \sum_{i=1}^{K} \varphi_i(u,v)\mathbf{p}_i. \tag{16}$$

This is the key result of our paper, since this equation gives a parametrization for the surface corresponding to any face of the control mesh, no matter what the valence is. There is no need to subdivide. Equation 16 also allows us to compute derivatives of the surface up to any order. Only the corresponding derivatives of the basis functions appearing in Equation 16 are required. For example, the partial derivative of the i-th eigenbasis with respect to u is:

$$\frac{\partial}{\partial u}\varphi_i(u,v)\Big|_{\Omega_k^n} = 2^n(\lambda_i)^{n-1}\frac{\partial}{\partial u}x_i(\mathbf{t}_{k,n}(u,v),k),$$

where the factor 2^n is equal to the derivative of the affine transformation $\mathbf{t}_{k,n}$. Generally a factor 2^{pn} will be present when the order of differentiation is p.

5 Implementation

Although the derivation of our evaluation technique is mathematically involved, its implementation is straightforward. The tedious task of computing the eigenstructure of the subdivision matrix only has to be performed once and is provided in Appendix A. In practice, we have precomputed these eigenstructures up to some maximum valence, say `NMAX=50`, and have stored them in a file. The file and some C code that reads in the data can be found on the course CDROM. Any program using our evaluation technique can read in these precomputed eigenstructures. In our implementation the eigenstructure for each valence `N` is stored internally as

```
typedef
 struct {
  double L[K];          /* eigenvalues */
  double iV[K][K];      /* inv of the eigenvectors */
  double x[K][3][16]; /* coeffs of the splines */
 } EIGENSTRUCT;
EIGENSTRUCT eigen[NMAX];,
```

where `K=2*N+8`. At the end of this section we describe how we computed these eigenstructures. We emphasize that this step has to be performed only once and that its computational cost is irrelevant to the efficiency of our evaluation scheme.

Given that the eigenstructures have been precomputed and read in from a file, we evaluate a surface patch around an extraordinary vertex in two steps. First, we project the control vertices surrounding the patch into the eigenspace of the subdivision matrix. Let the control vertices be ordered as shown in Figure 3 and stored in an array `C[K]`. The projected vertices `Cp[K]` are then easily computed by using the precomputed inverse of the eigenvectors:

```
ProjectPoints(point *Cp,point *C,int N){
 for ( i=0 ; i<2*N+8 ; i++ ){
  Cp[i] = (0,0,0);
  for ( j=0 ; j<2*N+8 ; j++ ){
   Cp[i] += eigen[N].iV[i][j] * C[j];
  }
 }
}
```

This routine is called only whenever one of the patches is evaluated for the first time or after an update of the mesh. This step is, therefore, called at most once per surface patch. The second step of our evaluation, on the other hand, is called whenever the surface has to be evaluated at a particular parameter value `(u,v)`. The second step is a straightforward implementation of the sum appearing in Equation 15. The following routine computes the surface patch at any parameter value.

```
EvalSurf ( point P, double u, double v,
    point *Cp, int N ) {
 /* determine in which domain $\Omega_k^n$ the parameter lies */
 n = floor(min(-log2(u),-log2(v)))+1;
 pow2 = pow(2,n-1);
 u *= pow2; v *= pow2;
 if ( v < 0.5 ) {
  k=0; u=2*u-1; v=2*v;
 }
 else if ( u < 0.5 ) {
  k=2; u=2*u; v=2*v-1;
 }
 else {
  k=1; u=2*u-1; v=2*v-1;
 }
 /* Now evaluate the surface */
 P = (0,0,0);
 for ( i=0 ; i<2*N+8 ; i++ ) {
  P += pow(eigen[N].L[i],n-1) *
     EvalSpline(eigen[N].x[i][k],u,v)*Cp[i];
 }
}
```

The function `EvalSpline` computes the bi-cubic B-spline polynomial whose coefficients are given by its first argument at the parameter value `(u,v)`. When either one of the parameter values `u` or `v` is zero, we set it to a sufficiently small value near the precision of the machine, to avoid an overflow that would be caused by the `log2` function. Because `EvalSpline` evaluates a bi-cubic polynomial, the cost of `EvalSurf` is comparable to that of a bi-cubic surface spline. The extra cost due to the logarithm and the elevation to an integer power is minimal, because these operations are efficiently implemented on most current hardware. Since the projection step is only called when the mesh is updated, the cost of our evaluation depends predominantly on `EvalSurf`.

The computation of the p-th derivative is entirely analogous. Instead of using the routine `EvalSpline` we employ a routine that returns the p-th derivative of the bi-cubic B-spline. In addition, the final result is scaled by a factor `pow(2,n*p)`. The evaluation of derivatives is essential in applications that require precise surface normals and curvature. For example, Newton iteration schemes used in ray surface computations require higher derivatives of the surface at arbitrary parameter values.

We now describe how we compute the eigenstructure of the subdivision matrix. This step only has to be performed once for a given set of valences. The efficiency of this step is not crucial. Accuracy is what matters here. As shown in the appendix, the eigenstructure of the two matrices $\mathbf{S}$ and $\mathbf{S}_{12}$ can be computed analytically. The corresponding eigenstructure of the extended subdivision matrix $\mathbf{A}$ requires the solution of the $2N+1$ linear systems of Equation 11. We did not solve these analytically because these systems are only of size 7×7. Consequently, these systems can be solved up to machine accuracy using standard linear solvers. We used the `dgesv` routine from LINPACK to perform the task. The inverse of the eigenvectors is computed by carrying out the matrix products appearing in Equation 12. Using the eigenvectors, we also precompute the coefficients of the bi-cubic splines $\mathbf{x}(u,v,k)$ as explained in Appendix B. For each valence N we stored the results in the data structure `eigen[NMAX]` and saved them in a file to be read in at the start of any application which uses the routines `ProjectPoints` and `EvalSurf` described above. This data is provided in the file `ccdata50.dat` on the CDROM for valences upto `NMAX=50`. The C program `cctest.c` demonstrates how to read in that data.

5.1 Some Remarks on Stability

As noted previously there is a problem when evaluating the surface at the extraordinary point using `EvalSurf` since the `log` is ill-defined at $(0,0)$. One option as mentioned above is to clamp the (u,v) values below a certain threshold. A better option is to return the limit point `Cp[0]` directly. When computing derivatives other instabilities can occur, although in practice we have not encountered them in our implementation using the clamping of the $u-v$ values. However, instabilities could be a nuisance in other applications of the evaluation method. The problem is that we have to multiply the derivative by a factor `pow(2,n*p)` which diverges for large `n`. One possible solution is to include this factor when taking powers of the eigenvalues, i.e., line

```
P += pow(eigen[N].L[i],n-1) *
```

should be replaced by

```
P += p2*pow(p2*eigen[N].L[i],n-1) * ,
```

where `p2=pow(2,p)`. Although this simple modification reduces instabilities it is not completely satisfactory. The problem is that the inverse of the eigenvalues λ are not always powers of two so that the products $(2^p\lambda)^n$ either converge to zero or diverge.

The cleanest solution is to reparametrize the surface using the characteristic map introduced in [8]. The characteristic map is simply the mapping defined by the eigenbasis functions $\varphi_2(u,v)$ and $\varphi_3(u,v)$. Let

$$\chi(u,v) = (x(u,v), y(u,v)) = (\varphi_2(u,v), \varphi_3(u,v)),$$

be the characteristic map. A more stable implementation would be to evaluate:

$$\mathbf{s}_\chi(x,y) = \mathbf{s}\left(\chi^{-1}(u,v)\right).$$

This reparametrization requires the inversion of the two eigenbasis functions φ_2 and φ_3. Note also that the evaluation of the derivatives requires the computation of the derivatives of the inverse of the characteristic map as well. Also we point out that the evaluation of the curvature near the extraordinary point is inherently unstable since the the curvature at these points is known not to exist (diverge) for Catmull-Clark surfaces. In particular, this implies that Catmull-Clark are in general not C^2 surfaces. As a corollary, for example, a perfect sphere cannot be represented exactly by a Catmull-Clark surface.

6 Results

In Figure 10 we depict several Catmull-Clark subdivision surfaces. The extraordinary vertex whose valence N is given in the figure is located in the center of each surface. The position information within the blue patches surrounding the extraordinary vertex is computed using our new evaluation technique. The remaining patches are evaluated as bi-cubic B-splines. Next to each surface we also depict the curvature of the surface. We map the value of the Gaussian curvature onto a hue angle. Red corresponds to a flat surface, while green indicates high curvature. We have purposely made the curvature plot discontinuous in order to emphasize the iso-contour lines. Both the shaded surface and the curvature plot illustrate the accuracy of our method. Notice especially how the curvature varies smoothly across the boundary between the patches evaluated using our technique and the regular bi-cubic B-spline patches. The curvature plots also indicate that for $N \neq 4$ the Gaussian curvature takes on arbitrarily large values near the extraordinary vertex. The curvature at the extraordinary vertex is in fact infinite, which explains the diverging energy functionals in [4].

Figure 11 depicts more complex surfaces. The blue patches are evaluated using our technique.

7 Conclusion and Future Work

In this paper we have presented a technique to evaluate Catmull-Clark subdivision surfaces. This is an important contribution since the lack of such an evaluation schemes has been sited as the chief argument against the use of subdivision scheme in free-form surface modelers. Our evaluation scheme permits many algorithms and analysis techniques developed for parametric surfaces to be extended to Catmull-Clark surfaces. The cost of our algorithm is comparable to the evaluation of a bi-cubic spline. The implementation of our evaluation is straightforward and we have used it to plot the curvature near extraordinary vertices. We believe that the same methodology can be applied to many other subdivision schemes sharing the features of Catmull-Clark subdivision: regular parametrization away from extraordinary vertices. We have worked out the details for Loop's triangular scheme, and the derivation can be found in the accompanying paper in these course notes. Catmull-Clark surfaces and Loop surfaces (when the valence $\neq 3$) share the property that their extended subdivision matrices are non-defective. In general, this is *not* the case. For example, the extended subdivision matrix of Doo-Sabin surfaces cannot generally be diagonalized. In that case, however, we can use the Jordan normal form of the extended subdivision matrix and employ Zorin's general scaling relations [11].

Acknowledgments

I wish to thank the following individuals for their help: Eugene Lee for assisting me in fine tuning the math, Michael Lounsbery and Gary Herron for many helpful discussions, Darrek Rosen for creating the models, Pamela Jackson for proofreading the paper, Gregg Silagyi for his help during the submission, and Milan Novacek for his support during all stages of this work. Thanks also to Markus Meister from Brown University who kindly pointed out a mistake in Appendix A of our SIGGRAPH paper.

A Subdivision Matrices and Their Eigenstructures

The matrix S corresponds to the extraordinary rules around the extraordinary vertex. With our choice of ordering of the control vertices the matrix is:

$$\mathbf{S} = \left(\begin{array}{c|cccccccccc}
a_N & b_N & c_N & b_N & c_N & b_N & \cdots & b_N & c_N & b_N & c_N \\ \hline
d & d & e & e & 0 & 0 & \cdots & 0 & 0 & e & e \\
f & f & f & f & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
d & e & e & d & e & e & \cdots & 0 & 0 & 0 & 0 \\
f & 0 & 0 & f & f & f & \cdots & 0 & 0 & 0 & 0 \\
\vdots & & & \vdots & & & \ddots & & \vdots & & \\
d & e & 0 & 0 & 0 & 0 & \cdots & e & e & d & e \\
f & f & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & f & f
\end{array}\right)$$

where

$$a_N = 1 - \frac{7}{4N}, b_N = \frac{3}{2N^2}, c_N = \frac{1}{4N^2}, d = \frac{3}{8}, e = \frac{1}{16}, f = \frac{1}{4}.$$

Since the lower right $2N \times 2N$ block of $\mathbf{S}$ has a cyclical structure, we can use the discrete Fourier transform to compute the eigenstructure of $\mathbf{S}$. This was first used in the context of subdivision surfaces by Doo and Sabin [3]. The discrete Fourier transform can be written compactly by introducing the following $2N \times 2N$ "Fourier matrix";

$$\mathbf{F} = \begin{pmatrix}
1 & 0 & 1 & 0 & \cdots & 1 & 0 \\
0 & 1 & 0 & 1 & \cdots & 0 & 1 \\
1 & 0 & \omega^{-1} & 0 & \cdots & \omega^{-(N-1)} & 0 \\
0 & 1 & 0 & \omega^{-1} & \cdots & 0 & \omega^{-(N-1)} \\
 & & \vdots & & \ddots & \vdots & \\
1 & 0 & \omega^{-(N-1)} & 0 & \cdots & \omega^{-(N-1)^2} & 0 \\
0 & 1 & 0 & \omega^{-(N-1)} & \cdots & 0 & \omega^{-(N-1)^2}
\end{pmatrix},$$

where $\omega = \exp(i2\pi/N)$. Using these notations we can write down the "Fourier transform" of the matrix $\mathbf{S}$ compactly as:

$$\hat{\mathbf{S}} = \left(\begin{array}{c|c|c|c}
\hat{\mathbf{S}}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline
\mathbf{0} & \hat{\mathbf{S}}_1 & \mathbf{0} & \mathbf{0} \\ \hline
\vdots & \mathbf{0} & \ddots & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{S}}_{N-1}
\end{array}\right) = \mathbf{T}\ \mathbf{S}\ \mathbf{T}^{-1},$$

where

$$\mathbf{T} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \frac{1}{N}\mathbf{F} \end{pmatrix}, \quad \mathbf{T}^{-1} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^* \end{pmatrix},$$

$$\hat{\mathbf{S}}_0 = \begin{pmatrix} a_N & Nb_N & Nc_N \\ d & 2f & 2e \\ f & 2f & f \end{pmatrix} \quad \text{and}$$

$$\hat{\mathbf{S}}_l = \begin{pmatrix} e\left(\omega^{-l}+\omega^{l}\right)+d & e\left(1+\omega^{-l}\right) \\ f\left(1+\omega^{l}\right) & f \end{pmatrix},$$

$l = 1, \cdots, N-1$. The eigenstructure of the Fourier transform $\hat{\mathbf{S}}$ is computed from the eigenstructures of its diagonal blocks. The first block $\hat{\mathbf{S}}_0$ has eigenvalues

$$\mu_1 = 1, \quad \mu_2, \mu_3 = \frac{1}{8N}\left(-7+3N \mp \sqrt{49-30N+5N^2}\right)$$

and eigenvectors

$$\hat{\mathbf{K}}_0 = \begin{pmatrix} 1 & 16\mu_2^2-12\mu_2+1 & 16\mu_3^2-12\mu_3+1 \\ 1 & 6\mu_2-1 & 6\mu_3-1 \\ 1 & 4\mu_2+1 & 4\mu_3+1 \end{pmatrix}.$$

Similarly, the two eigenvalues of each block $\hat{\mathbf{S}}_l$ ($l = 1, \cdots, N-1$) are equal to:

$$\lambda_l^{\mp} = \frac{1}{16}\left(5+\cos\left(\frac{2\pi l}{N}\right) \mp \cos\left(\frac{\pi l}{N}\right)\sqrt{18+2\cos\left(\frac{2\pi l}{N}\right)}\right),$$

where we have used some trigonometric relations to simplify the resulting expressions. The corresponding eigenvectors of each block are

$$\hat{\mathbf{K}}_l = \begin{pmatrix} 4\lambda_l^- - 1 & 4\lambda_l^+ - 1 \\ 1+\omega^l & 1+\omega^l \end{pmatrix}.$$

We have to single out the special case when N is even and $l = N/2$. In this case the corresponding block is

$$\hat{\mathbf{K}}_{N/2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The eigenvalues of the matrix $\hat{\mathbf{S}}$ are the union of the eigenvalues of its blocks and the eigenvectors are

$$\hat{\mathbf{K}} = \left(\begin{array}{c|c|c|c} \hat{\mathbf{K}}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \hat{\mathbf{K}}_1 & \mathbf{0} & \mathbf{0} \\ \hline \vdots & \mathbf{0} & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{K}}_{N-1} \end{array}\right).$$

Since the subdivision matrix $\mathbf{S}$ and its Fourier transform $\hat{\mathbf{S}}$ are similar, they have the same eigenvalues. The eigenvectors are computed by inverse Fourier transforming these eigenvectors:

$$\mathbf{K} = \mathbf{T}^{-1}\,\hat{\mathbf{K}}.$$

Consequently, we have computed the eigenvalues and eigenvectors of $\mathbf{S}$. However, in this form the eigenvectors are complex valued and most of the eigenvalues are actually of multiplicity two, since $\lambda_l^- = \lambda_{N-l}^+$ and $\lambda_l^+ = \lambda_{N-l}^-$. We relabel these eigenvalues as follows:

$$\mu_4 = \lambda_1^-, \mu_5 = \lambda_1^+, \mu_6 = \lambda_2^-, \mu_7 = \lambda_2^+, \cdots$$

Since we have rearranged the eigenvalues, we have to rearrange the eigenvectors. At the same time we make these eigenvectors real. Let $\mathbf{k}_1, \cdots, \mathbf{k}_{2N+1}$ be the columns of $\mathbf{K}$, then we can construct the columns of a matrix $\mathbf{U}_0$ as follows:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{k}_1, \quad \mathbf{u}_2 = \mathbf{k}_2, \quad \mathbf{u}_3 = \mathbf{k}_3, \\
\mathbf{u}_{2l+2} &= \frac{1}{2}\left(\mathbf{k}_{l+3} + \mathbf{k}_{2N-l+2}\right) \quad \text{and} \\
\mathbf{u}_{2l+3} &= \frac{1}{2i}\left(\mathbf{k}_{l+3} - \mathbf{k}_{2N-l+2}\right).
\end{aligned}$$

More precisely $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$, $\mathbf{u}_{2l+2}$ and $\mathbf{u}_{2l+3}$ are equal to

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 16\mu_2^2 - 12\mu_2 + 1 \\ 6\mu_2 - 1 \\ 4\mu_2 + 1 \\ \vdots \\ 6\mu_2 - 1 \\ 4\mu_2 + 1 \end{pmatrix}, \begin{pmatrix} 16\mu_3^2 - 12\mu_3 + 1 \\ 6\mu_3 - 1 \\ 4\mu_3 + 1 \\ \vdots \\ 6\mu_3 - 1 \\ 4\mu_3 + 1 \end{pmatrix},$$

$$\begin{pmatrix} 0 \\ 4\mu_{l+3} - 1 \\ 1 + C_{\gamma(l)} \\ (4\mu_{l+3} - 1)C_{\gamma(l)} \\ C_{\gamma(l)} + C_{2\gamma(l)} \\ \vdots \\ (4\mu_{l+3} - 1)C_{(N-1)\gamma(l)} \\ C_{(N-1)\gamma(l)} + 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 \\ 0 \\ S_{\gamma(l)} \\ (4\mu_{l+3} - 1)S_{\gamma(l)} \\ S_{\gamma(l)} + S_{2\gamma(l)} \\ \vdots \\ (4\mu_{l+3} - 1)S_{(N-1)\gamma(l)} \\ S_{(N-1)\gamma(l)} \end{pmatrix},$$

respectively, where $l = 1, \cdots, N_2$, $N_2 = N - 1$ when N is odd and $N_2 = N - 2$ when N is even, $\gamma(l) = (l+1)/2$ when l is odd and $\gamma(l) = l/2$ when l is even, and

$$C_k = \cos\left(\frac{2\pi k}{N}\right) \quad \text{and} \quad S_k = \sin\left(\frac{2\pi k}{N}\right).$$

When N is even the last two eigenvectors are

$$\begin{aligned}
\mathbf{u}_{2N}^T &= (0, 1, 0, -1, 0, 1, 0, \cdots, -1, 0) \quad \text{and} \\
\mathbf{u}_{2N+1}^T &= (0, 0, 1, 0, -1, 0, 1, \cdots, 0, -1).
\end{aligned}$$

Finally, the diagonal matrix of eigenvalues is

$$\mathbf{\Sigma} = \operatorname{diag}\left(1, \mu_2, \mu_3, \mu_4, \mu_4, \cdots, \mu_{N+2}, \mu_{N+2}\right).$$

The inverse of the eigenvectors $\mathbf{U}_0$ can be computed likewise by first computing the inverses of each block $\hat{\mathbf{K}}_l$ in the Fourier domain and then setting

$$\mathbf{K}^{-1} = \hat{\mathbf{K}}^{-1}\,\mathbf{T}.$$

With the same reshuffling as above we can then compute $\mathbf{U}_0^{-1}$. The resulting expressions are, however, rather ugly and are not reproduced in this paper.

The remaining blocks of the subdivision matrix $\mathbf{A}$ directly follow from the usual B-spline knot-insertion rules.

$$\mathbf{S}_{12} = \begin{pmatrix} c & b & c & 0 & b & c & 0 \\ 0 & e & e & 0 & 0 & 0 & 0 \\ 0 & c & b & c & 0 & 0 & 0 \\ 0 & 0 & e & e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e & e & 0 \\ 0 & 0 & 0 & 0 & c & b & c \\ 0 & 0 & 0 & 0 & 0 & e & e \end{pmatrix}, \ \mathbf{S}_{11} = \begin{pmatrix} c & 0 & 0 & b & a & b & 0 & 0 & \mathbf{0} \\ e & 0 & 0 & e & d & d & 0 & 0 & \mathbf{0} \\ b & 0 & 0 & c & b & a & b & c & \mathbf{0} \\ e & 0 & 0 & 0 & 0 & d & d & e & \mathbf{0} \\ e & 0 & 0 & d & d & e & 0 & 0 & \mathbf{0} \\ b & c & b & a & b & c & 0 & 0 & \mathbf{0} \\ e & e & d & d & 0 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix},$$

where

$$a = \frac{9}{16}, \quad b = \frac{3}{32} \ \text{ and } \ c = \frac{1}{64}.$$

For the case $N = 3$, there is no control vertex $\mathbf{c}_8$ ($\mathbf{c}_8 = \mathbf{c}_2$) and the second column of the matrix $\mathbf{S}_{11}$ is equal to $(0, 0, c, e, 0, c, e)^T$.

The eigenstructure of the matrix $\mathbf{S}_{12}$ can be computed manually, since this matrix has a simple form. Its eigenvalues are:

$$\mathbf{\Delta} = \operatorname{diag}\left(\frac{1}{64}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\right),$$

with corresponding eigenvectors:

$$\mathbf{W}_1 = \begin{pmatrix} 1 & 1 & 2 & 11 & 1 & 2 & 11 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 2 \end{pmatrix}.$$

The inverse $\mathbf{W}_1^{-1}$ of this matrix is easily computed manually.

The other two matrices appearing in $\bar{\mathbf{A}}$ are:

$$\mathbf{S}_{21} = \begin{pmatrix} 0 & 0 & 0 & 0 & f & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & d & e & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & f & f & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & e & d & e & \mathbf{0} \\ 0 & 0 & 0 & 0 & 0 & f & f & \mathbf{0} \\ 0 & 0 & 0 & e & d & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & f & f & 0 & 0 & \mathbf{0} \\ 0 & 0 & e & d & e & 0 & 0 & \mathbf{0} \\ 0 & 0 & f & f & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}, \mathbf{S}_{22} = \begin{pmatrix} f & f & 0 & 0 & f & 0 & 0 \\ e & d & e & 0 & e & 0 & 0 \\ 0 & f & f & 0 & 0 & 0 & 0 \\ 0 & e & d & e & 0 & 0 & 0 \\ 0 & 0 & f & f & 0 & 0 & 0 \\ e & e & 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & 0 & f & f & 0 \\ 0 & 0 & 0 & 0 & e & d & e \\ 0 & 0 & 0 & 0 & 0 & f & f \end{pmatrix}.$$

B Eigenbasis Functions

In this appendix we compute the bi-cubic spline pieces $\mathbf{x}(u,v,k)$ of the eigenbasis defined in Equation 13. The vector $\mathbf{b}(u,v)$ contains the 16 tensor B-spline basis functions ($i = 1, \cdots, 16$):

$$b_i(u,v) = N_{(i-1)\%4}(u) N_{(i-1)/4}(v),$$

where "%" and "/" stand for the remainder and the division respectively. The functions $N_i(t)$ are the uniform B-spline basis functions:

$$\begin{aligned} 6N_0(t) &= 1 - 3t + 3t^2 - t^3, \\ 6N_1(t) &= 4 - 6t^2 + 3t^3, \\ 6N_2(t) &= 1 + 3t + 3t^2 - 3t^3 \quad \text{and} \\ 6N_3(t) &= t^3. \end{aligned}$$

The projection matrices $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are defined by introducing the following three permutation vectors (see Figure 6):

$$\begin{aligned} \mathbf{q}^1 &= (8, 7, 2N+5, 2N+13, 1, 6, 2N+4, 2N+12, \\ &\quad 4, 5, 2N+3, 2N+11, 2N+7, 2N+6, 2N+2, \\ &\quad 2N+10), \\ \mathbf{q}^2 &= (1, 6, 2N+4, 2N+12, 4, 5, 2N+3, 2N+11, \\ &\quad 2N+7, 2N+6, 2N+2, 2N+10, 2N+16, \\ &\quad 2N+15, 2N+14, 2N+9), \\ \mathbf{q}^3 &= (2, 1, 6, 2N+4, 3, 4, 5, 2N+3, 2N+8, 2N+7, \\ &\quad 2N+6, 2N+2, 2N+17, 2N+16, 2N+15, \\ &\quad 2N+14). \end{aligned}$$

Since for the case $N = 3$ the vertices $\mathbf{c}_2$ and $\mathbf{c}_8$ are the same vertex, $q_1^1 = 2$ instead of 8 for $N = 3$. Using these permutation vectors we can compute each bi-cubic spline as follows:

$$x_i(u,v,k) = \sum_{j=1}^{16} \bar{V}_{q_j^k,i}\, b_j(u,v),$$

where $i = 1, \cdots, K$ and $\bar{\mathbf{V}} = \bar{\mathbf{A}}\mathbf{V}$.

References

[1] A. A. Ball and J. T. Storry. Conditions For Tangent Plane Continuity Over Recursively Defined B-spline Surfaces. *ACM Transactions on Graphics*, 7(2):83–102, April 1988.

[2] E. Catmull and J. Clark. Recursively Generated B-Spline Surfaces On Arbitrary Topological Meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[3] D. Doo and M. A. Sabin. Behaviour Of Recursive Subdivision Surfaces Near Extraordinary Points. *Computer Aided Design*, 10(6):356–360, 1978.

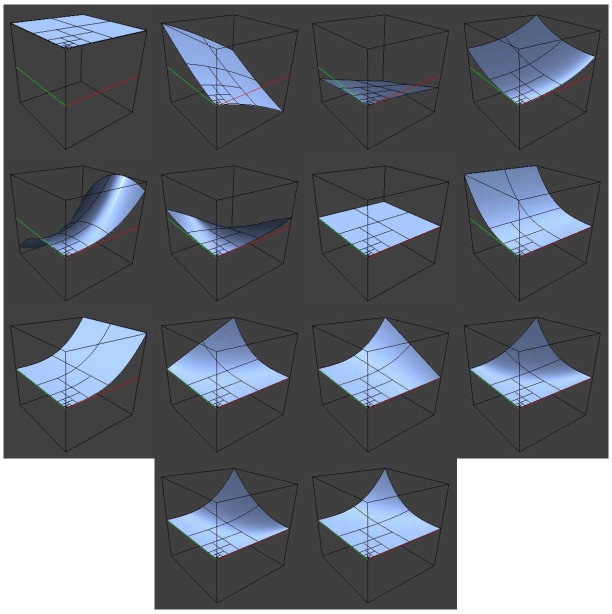

Figure 8: The complete set of 14 eigenbasis functions for extraordinary vertices of valence $N = 3$.

[4] M. Halstead, M. Kass, and T. DeRose. Efficient, Fair Interpolation Using Catmull-Clark Surfaces. In *Proceedings of SIGGRAPH '93*, pages 35–44. Addison-Wesley Publishing Company, August 1993.

[5] C. T. Loop. *Smooth Subdivision Surfaces Based on Triangles*. M.S. Thesis, Department of Mathematics, University of Utah, August 1987.

[6] J. Peters and U. Reif. Analysis Of Generalized B-Splines Subdivision Algorithms. To appear in SIAM Journal of Numerical Analysis.

[7] H. Qin, C. Mandal, and C. Vemuri. Dynamic Catmull-Clark Subdivision Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, July-September 1998.

[8] U. Reif. A Unified Approach To Subdivision Algorithms Near Extraordinary Vertices. *Computer Aided Geometric Design*, 12:153–174, 1995.

[9] J. Stam. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics Proceedings, Annual Conference Series, 1998*, pages 395–404, July 1998.

[10] J. Warren. Subdivision Methods For Geometric Design. Unpublished Manuscript. Preprint available on the web at `http://www.cs.rice.edu/~jwarren/papers/book.ps.gz`, 1994.

[11] D. N. Zorin. *Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, Pasadena, California, 1997.

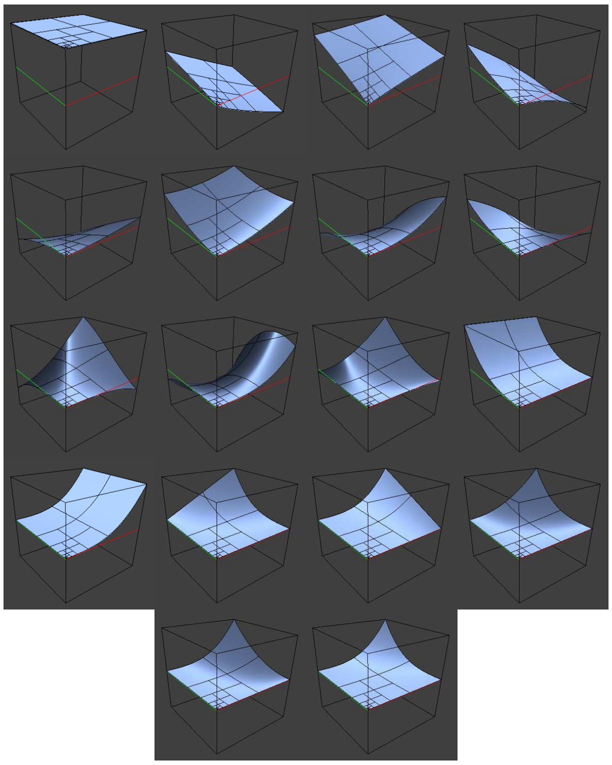

Figure 9: The complete set of 18 eigenbasis functions for extraordinary vertices of valence $N = 5$.

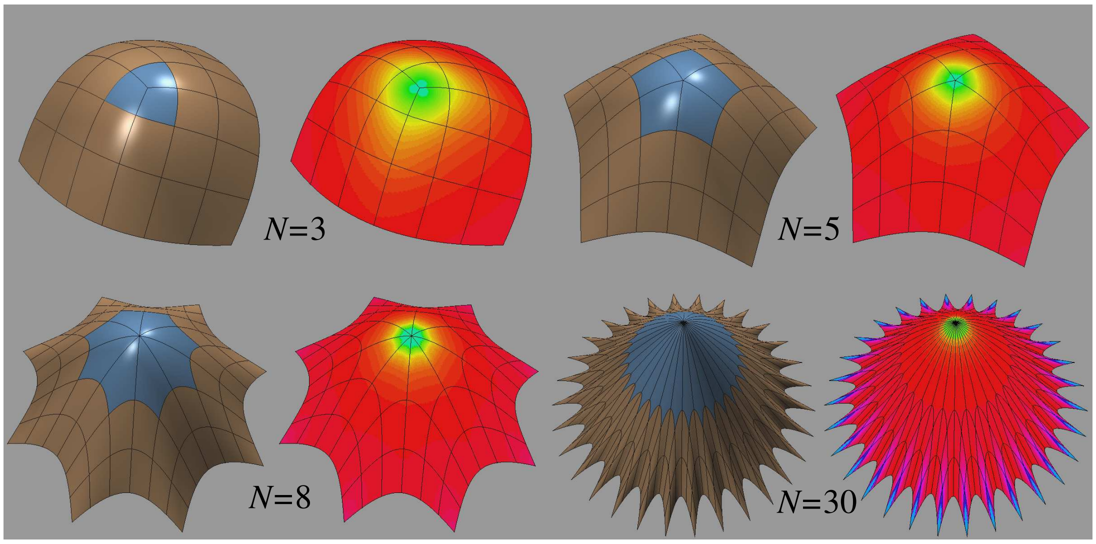

Figure 10: Surfaces having an extraordinary vertex in the center. For each surface we depict the patches evaluated using our technique in blue. Next to them is a curvature plot. Derivative information for curvature is also computed near the center vertex using our technique.

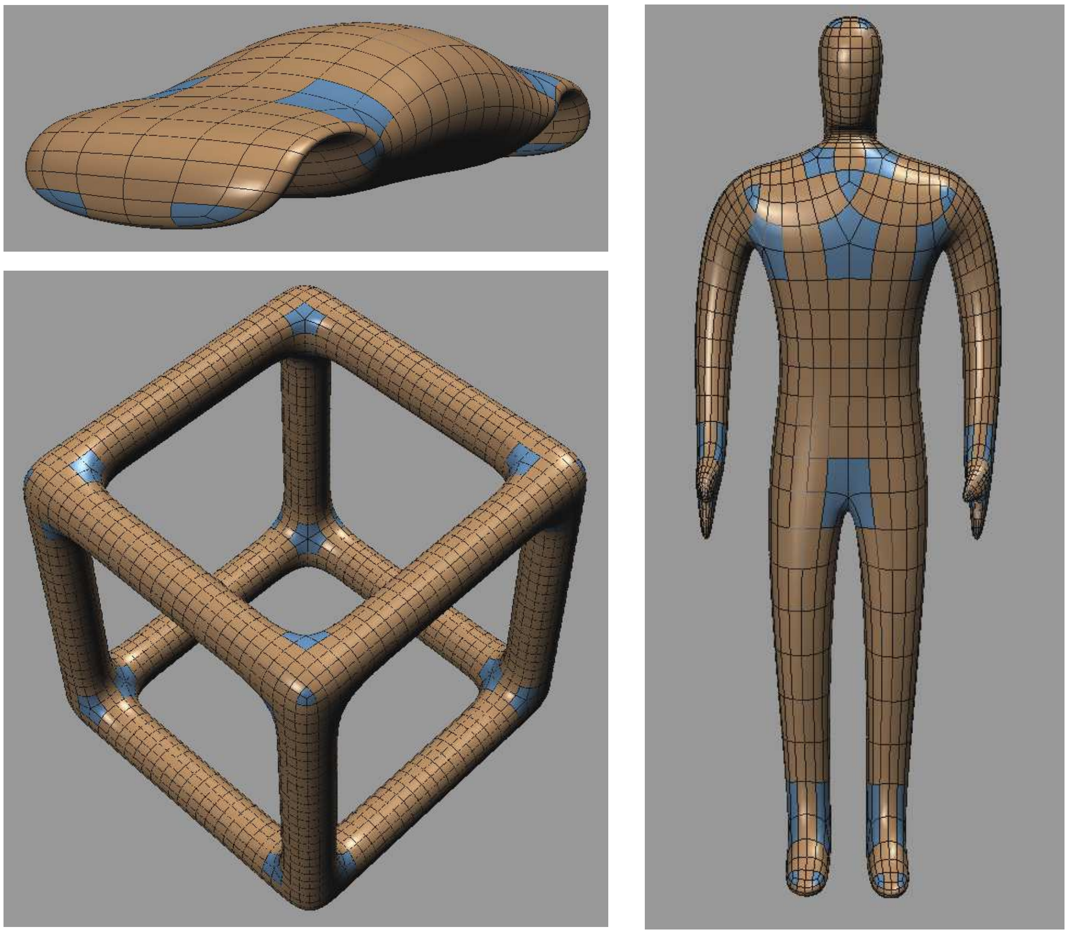

Figure 11: More complex surfaces rendered using our evaluation technique (in blue).