Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

Lofting Curve Networks using Subdivision Surfaces

S. Schaefer[1], J. Warren[1], D. Zorin[2]

[1] Rice University; [2] New York University

Abstract

Lofting is a traditional technique for creating a curved shape by first specifying a network of curves that approximates the desired shape and then interpolating these curves with a smooth surface. This paper addresses the problem of lofting from the viewpoint of subdivision. First, we develop a subdivision scheme for an arbitrary network of cubic B-splines capable of being interpolated by a smooth surface. Second, we provide a quadrangulation algorithm to construct the topology of the surface control mesh. Finally, we extend the Catmull-Clark scheme to produce surfaces that interpolate the given curve network. Near the curve network, these lofted subdivision surfaces are C^2 bicubic splines, except for those points where three or more curves meet. We prove that the surface is C^1 with bounded curvature at these points in the most common cases; empirical results suggest that the surface is also C^1 in the general case.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Lofting is a common technique for constructing smooth surfaces for computer graphics and computer-aided design applications. In this approach, the user defines a curve network and a smooth surface interpolating this network is constructed automatically. The problem of constructing a surface using a curve network can be split into several steps. First, a method for defining a curve network compatible with a smooth surface is needed, as several curves meeting at a point need to have tangent lines in the same plane for such surface to exist. Next, we need a way to define the topology of the surface, as surfaces of different topology may be compatible with a given curve network; finally, we need algorithms for computing the geometry of the surface interpolating the curves.

In our approach, the *curve network* consisting of cubic B-splines is inferred from curve control points connected by line segments in the *polyline network*. Control points where more than two line segments meet are called corners. A curve network compatible with a smooth surface is computed from control points using the *curve network subdivision scheme*.

A set of closed loops of curves are identified as boundaries of surface *patches*. Specifying these loops defines the topology of the surface uniquely. Additional control points are introduced in the interior of patches using a connectivity construction algorithm and fairing; finally, a smooth surface is computed using a modification of Catmull-Clark subdivision scheme. Figure 1 shows an example of a polyline network q^0 defining the curve network q^∞. On the right, the surface control mesh p^0 for this example was computed automatically from the polyline network q^0. When subdivided using our modified Catmull-Clark algorithm, p^0's associated limit surface p^∞ interpolates curve network q^∞.

Previous Work. Early work on lofting [CK83, TS90] focused on its inherent difficulties such as filling n-sided holes and maintaining higher order smoothness. Later work [Her96, Vár91] developed new types of patches suitable for lofting. While there has been considerable success with these approaches, subdivision surfaces provide a simple standard framework for this task, especially for computer graphics applications, as arbitrary meshes with complex constraints at corners can be handled with greater ease.

Unfortunately, Catmull-Clark surfaces [CC78], the standard generalization of bicubic splines, are incapable of interpolating a network of cubic splines since the image of a curve incident on an extraordinary vertex is not piecewise polynomial. The pioneer in this area, Nasri, has developed subdivision methods for lofting based on the concept of a *polygonal complex* [Nas97, Nas00, NA02, Nas03, NKL01].

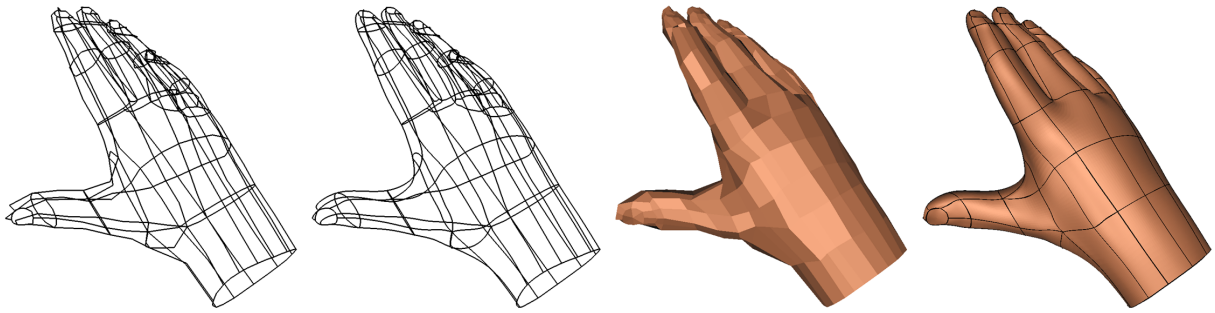

Figure 1: *A network of polygonal lines q^0 and its associated curve network q^∞ (left). The base mesh p^0 for a modified Catmull-Clark surface whose limit surface p^∞ (right) interpolates q^∞. p^0 was automatically computed from q^0 using a combination of skinning, fairing and lofting.*

Such complexes consist of the portion of a surface mesh that controls the shape of a curve on the limit surface. By adjusting the shape of this complex, the designer can adjust the shape of the curve as well as its cross boundary derivatives. Another subdivision approach to lofting is Levin's *combined* subdivision scheme [Lev99]. This method adjusts the surface subdivision rules near the curve network to ensure that the surface smoothly interpolates the resulting network. Combined subdivision produces surfaces that can interpolate arbitrary parametric curves, not just networks of cubic splines.

Contributions. We present a new method for lofting curve networks which has the following features:

- The resulting surfaces are standard Catmull-Clark away from corners of the curve network;
- Curves of the network are cubic splines embedded in the surfaces; curves can be tagged as creases;
- Curves can terminate or continue through corner vertices;
- At corner vertices without creases, curves lie on a common C^2 surface.

Our lofting method is fully automatic and requires only the minimal necessary input from the user designing the curve network. In particular, the method automatically quadrangulates patches formed by the network of polygonal lines and fairs the resulting mesh. Nasri [NAH03] considers building quadrangulations specific to subdivision surfaces by optimizing the number and valence of extraordinary vertices. However, only simple configurations for skinning between two disjoint curves are considered.

In comparison to the work of Levin, we consider a less general setting; as a result we gain the advantage of preserving the piecewise polynomial nature of the surface away from extraordinary vertices and do not require cross boundary information for curves to produce high-quality results. Furthermore, Levin restricts the topology of the curve network where we consider arbitrary connectivity of the curve network. Finally, in contrast to Nasri's approach, we do not use a polygonal complex, which simplifies construction of the fully automatic method.

2. Overview of the approach

The input to our method is a polyline network, a collection of open polygonal lines that terminate in a set of common corners. Pairs of polylines incident on a common corner may be tagged as modeling a single curve passing through the corner. Each polyline in the network is tagged as being either smooth or creased. We note that there are many ways to associate a curve network with the polyline network. Our curve network subdivision scheme is one convenient way to ensure that the limit curves are cubic splines, and, at the same time, for *arbitrary* positions of control points we obtain a curve network compatible with a smooth surface.

To build a surface, a designer specifies a polyline network and a set of patches as a list of topological faces (lists of indexed corner vertices). It is important to note that topological specification of patches is a necessary part of the information needed for lofting. In many configurations it is possible to infer this information from curves, for example, if the projection of a part of the network to a plane is forms a tessellation of an area of the plane. Unfortunately, there exist local rotationally symmetric configurations, such as six edges lying on the coordinate axes and meeting at the origin, for which there appears no general way to infer surface topology.

Given this information, our automated lofting process consists of three steps:

- *Skinning:* Compute the connectivity for a base mesh by quadrangulating the cycle of polylines bounding each patch (Section 6.1);
- *Fairing:* Position the vertices of this base mesh such that its limit surface interpolates the curve network and has a visually pleasing appearance (Section 6.2);
- *Subdivision:* Apply modified Catmull-Clark subdivision to this base mesh and construct a fine mesh that approximates the limit surface. (Sections 4 and 5).

We start with a detailed description of our subdivision scheme for curve networks and its related surface scheme, as these are the central elements of our approach. Then we conclude by discussing our method for skinning and fairing.

Formulation of subdivision rules. One of the difficulties of designing subdivision schemes with sufficient flexibility for realistic modeling applications is the large number of rules one needs to handle. Traditionally we specify subdivision schemes as a collection of masks determined by the local connectivity and tag configurations of the surface. However, at corners of the polyline networks, which are also vertices of the associate surface control mesh, the number of different choices of continuity constraints on incident curves grows rapidly, making the fixed mask approach very cumbersome.

For our purposes, we view subdivision as an operator $\mathbf{S}$ that acts on control meshes and produces refined control meshes. The restriction of this operator to a fixed-size neighborhood of a vertex can be represented as a matrix S for the type of subdivision schemes that we consider since $\mathbf{S}$ uses affine combinations of control points to compute the control points on the next level. These matrices depend only on the local mesh connectivity and tags. It is often useful to consider the neighborhoods of vertices of the same size on all subdivision levels. In this case such matrices are square, and do not depend on subdivision level for the type of schemes we consider. We call such matrices *local subdivision matrices*.

Such matrices are often used to analyze smoothness. We also use them in implementation: for corner vertices, we adopt the approach of precomputing appropriate subdivision matrices based on the parameters of each different corner configuration using simple linear algebra and use these matrices directly instead of subdivision masks. Since these matrices act on vectors of control points in the two-ring of the corner, the matrices are small and easy to compute.

3. Lofting two intersecting cubic splines

This section considers the fundamental problem of interpolating two intersecting cubic splines using a bicubic surface. The possibility of such interpolation is well-known. Our goal is to illustrate three main ideas: (a) a subdivision schemes for networks of cubic splines interpolating corner control points; (b) the change of basis operator; (c) the commutative relation satisfied by curve and surface subdivision operators and the change of basis operator.

A change of basis operator $\mathbf{M}$, extracts a polyline network from a surface control mesh and updates the positions of the control points for this network. A curve subdivision operator $\mathbf{C}$, which maps a polyline network to a refined polyline network and a surface subdivision operator $\mathbf{S}$ are said to *commute* with respect to $\mathbf{M}$ if

$$\mathbf{CM} = \mathbf{MS}. \tag{1}$$

Note this relation implies that repeated applications of $\mathbf{C}$ and $\mathbf{S}$ are also related via $\mathbf{C}^k\mathbf{M} = \mathbf{MS}^k$. It is easy to show that the limit surface defined by $\mathbf{S}$ interpolates the limit curves defined by $\mathbf{C}$, assuming both schemes converge. Furthermore, local subdivision matrices corresponding to the operators $\mathbf{C}$, $\mathbf{M}$ and $\mathbf{S}$ must also satisfy the equation 1; this fact is used to compute the rules for neighborhoods of corner points for arbitrary curve networks.

Interpolating a single point on a cubic spline. Before we consider the surface, case, we examine the simple problem of modifying the subdivision rules for a uniform cubic spline in the neighborhood of a single vertex v such that the resulting scheme interpolates the control point v, but the limit curve remains a cubic B-spline. These rules are an important component in the construction of change of basis operators and curve network subdivision schemes.

Recall that the subdivision matrix C_u (acting on v and its two neighbors on each side) for uniform cubic splines has the form

$$C_u = \left(\begin{array}{c|cc|cc} \frac{3}{4} & \frac{1}{8} & \frac{1}{8} & 0 & 0 \\ \hline \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ \hline \frac{1}{8} & \frac{3}{4} & 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & 0 & \frac{3}{4} & 0 & \frac{1}{8} \end{array}\right) \tag{2}$$

where we choose the indices of vertices along the curve to be 5,3,1,2,4, with v having index 1, to make the structure of the matrix more apparent.

Given the control points q_u^0 for a uniform cubic spline, we can reposition v to lie at its limit position by computing $q^0 = Nq_u^0$ where

$$N = \left(\begin{array}{c|cc|cc} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right).$$

The mask $(\frac{1}{6}\ \frac{2}{3}\ \frac{1}{6})$ is the standard limit position mask for cubic splines. We compute the rules for the modified curve C using the commutative relation $CN = NC_u$ where N is the change of basis matrix. Solving for C yields

$$C = \left(\begin{array}{c|cc|cc} 1 & 0 & 0 & 0 & 0 \\ \hline \frac{3}{4} & \frac{3}{8} & -\frac{1}{8} & 0 & 0 \\ \frac{3}{4} & -\frac{1}{8} & \frac{3}{8} & 0 & 0 \\ \hline \frac{3}{16} & \frac{23}{32} & -\frac{1}{32} & \frac{1}{8} & 0 \\ \frac{3}{16} & -\frac{1}{32} & \frac{23}{32} & 0 & \frac{1}{8} \end{array}\right) \tag{3}$$

in the two-ring of v. Outside the two-ring of v, the subdivision rules remain those of uniform cubic splines. Due to the commutative relation, starting with the control points Nq_u^0 and applying C to vertices in two-ring of v and uniform spline rules elsewhere yield the same cubic spline in the limit.

Interpolating a single cubic spline. To demonstrate a simple example of the commutative relation 1 for surfaces, we examine the case of a uniform bicubic B-spline surface interpolating an isolated B-spline curve or a set of isolated curves (e.g. Figure 3, left).

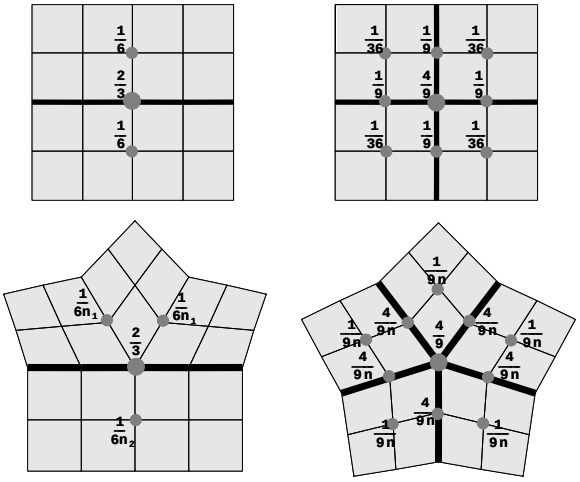

Figure 2: *Change of basis mask M for an interior vertex of q^0 (top-left), a corner vertex of q^0 of valence four (top-right), a general interior vertex (bottom-left) and a corner vertex of valence n (bottom-right). Lofted edges are shown in bold.*

The curve subdivision operator **C** in this case is determined by the uniform cubic B-spline subdivision rules, as there are no corners. Similarly, the surface subdivision operator **S** is just the subdivision operator for tensor-product bicubic splines. If the change of basis operator **M** is chosen to apply the limit mask across the curve as shown in figure 2 (top left), then the commutative relation in equation 1 is satisfied.

Interpolating two intersecting cubic splines. Finally, we consider a tensor-product bicubic uniform B-spline surface that interpolates two intersecting cubic splines.

The surface subdivision operator **S** in this case remains the same; the curve subdivision operator is obtained using the rules from equation 3 for each curve. This ensures that the common control point is interpolated. The local subdivision matrix C for the 2-ring of the corner vertex has the form

$$C = \left(\begin{array}{c|cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \frac{3}{4} & \frac{3}{8} & 0 & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & 0 & \frac{3}{8} & 0 & -\frac{1}{8} & 0 & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{1}{8} & 0 & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & 0 & -\frac{1}{8} & 0 & \frac{3}{8} & 0 & 0 & 0 & 0 \\ \hline \frac{3}{16} & \frac{23}{32} & 0 & -\frac{1}{32} & 0 & \frac{1}{8} & 0 & 0 & 0 \\ \frac{3}{16} & 0 & \frac{23}{32} & 0 & -\frac{1}{32} & 0 & \frac{1}{8} & 0 & 0 \\ \frac{3}{16} & -\frac{1}{32} & 0 & \frac{23}{32} & 0 & 0 & 0 & \frac{1}{8} & 0 \\ \frac{3}{16} & 0 & -\frac{1}{32} & 0 & \frac{23}{32} & 0 & 0 & 0 & \frac{1}{8} \end{array}\right) \quad (4)$$

where the order of the vertices in each block is cyclic and the block ordering is given in figure 10.

The change of basis operator **M** in this case extracts a row and a column of control points corresponding to the curves, and uses the masks which are tensor products of the masks encoded as rows in the curve change of basis matrix N (Figure 2, top right). Figure 3 (right) shows an example of a spherical surface interpolating three circular cubic splines that have six pair wise intersections.

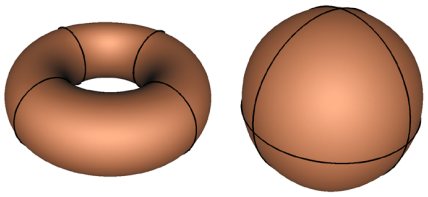

Figure 3: *Lofting four non-intersecting curves with a toroidial surface (left). Lofting three intersecting curves with a spherical surface (right).*

In the examples considered in this section, the surface subdivision scheme was known, and the curve subdivision schemes were either known or could be easily deduced from the corner vertex interpolation constraint. In the next two sections, we consider the problem of constructing curve subdivision schemes for arbitrary numbers of cubic splines meeting at a common corner, possibly with continuity constraints. Unlike the case of two intersecting curves, which always share a tangent plane, special care must be taken to ensure that the resulting networks are compatible with a smooth surface. Such rules are constructed in matrix form, generalizing the matrix from equation 4.

Afterwards, we generalize the change of basis operator **M** to arbitrary curve networks. Finally, knowing the curve network scheme and the change of basis operator, we use the commutative relation to construct a surface subdivision scheme interpolating the curve network which reduces to tensor product B-spline subdivision rules away from corners.

4. Subdivision for curve networks

As defined in the overview, a polyline network q^0 consists of a set of open polylines q_i^0 that terminate at a set of shared corners. Pairs of polylines incident on a common corner v may be tagged as forming a single polyline crossing the corner v. Individual polylines may be tagged as being either smooth or creased. In addition, topological faces (ordered lists of corner vertices) are specified, which uniquely define the topology of a surface associated with the network.

For the two-ring of each corner in q^0, we construct a subdivision matrix C with the following properties:

- Each polyline q_i^0 converges to a cubic spline q_i^∞. Tagged pairs of polylines incident on a common corner converge to a single uniform C^2 cubic spline passing through that corner, a *cross* curve.
- If none of the polylines incident on a corner are creased, the associated curve network has a unique tangent plane as well as a common best approximating quadric surface at the corner. (Having such a quadric makes lofting with a bounded curvature surface easier.)

- If several of the polylines incident on a corner are creased, each portion of the network between two consecutive creased curves (a *sector*) has a common tangent plane (and best approximating quadric).

If a polyline contains only one interior vertex, we average the rules for the overlapping portion of the two-rings for each corner. Outside the two-ring of each corner in q^0, our scheme uses the subdivision rules for cubic splines from equation 2.

Note that if we assume an ordering on these incident edges, this symmetry is broken and the process of determining a common tangent plane is made much more robust. For curve networks used in lofting, this ordering of edges can be inferred from the patch structure placed on top of the curve network. Specifically, if the surface is a topological manifold, the patch structure defines an ordering of the edges incident on v. We use this ordering to construct the subdivision rules at the corners of the curve network.

4.1. Overview

Due to the presence of cross curves and crease curves, the subdivision rules at a corner of q^0 include a large number of cases that, if enumerated, would result in a large amount of tedious, special coding. Thus, instead of taking the more traditional route of explicitly specifying masks for various cases, we construct all necessary rules in a concise and uniform manner using a matrix approach. Using this method, we consider only two cases separately: the case of a corner vertex with no incident crease curves and the case of a sector at a corner vertex bounded by two crease curves (cf. [BLZ00]). The rules for vertices outside two-rings of corner vertices are the standard uniform cubic B-spline rules.

In both cases, our construction is based on the generalization of the ideas that were used in subdivision literature since [DS78]: to achieve desired behavior, the subdivision matrix is decomposed as $C = Z\Lambda Z^{-1}$ (where Z is the matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues), eigenvalues are changed, and the matrix is assembled back with modified eigenvalues $C' = Z\Lambda' Z^{-1}$. This is usually done analytically, and explicit masks are derived for implementation. While this approach works well in simple cases, we go a step further for the more general cases that arise during practical modeling. In this situation, we *prescribe* both the eigenvalues and eigenvectors in such a way that the resulting subdivision rules (encoded in the subdivision matrix) have the properties described above. Furthermore, the strategy we use for constructing the eigenvectors Z is to generalize the eigenvectors for the matrix C in equation 4 to higher valences and creased configurations.

Our matrix C, defined over the two-ring of a corner vertex of valence n, has size $(2n+1)\times(2n+1)$ with $2n+1$ eigenvectors. For $n > 5$, we specify fewer eigenvectors (specifically, $6+n$) with the remaining eigenvectors having an associated eigenvalue of zero. Instead of constructing the eigenvectors corresponding to these zero eigenvalues explicitly, we use the pseudo-inverse $Z^+ = (Z^T Z)^{-1} Z^T$ to reconstruct C from a smaller set of nonzero eigenvalues and corresponding eigenvectors:

$$C = Z\Lambda Z^+ \tag{5}$$

4.2. Rules for corners with no incident crease curves

To construct Z in this case, we first define a set of n angles $\psi_0, \ldots \psi_{n-1}$ which control the directions in which incident curves approach the corner vertex in its tangent plane. If the corner has no cross curves, we force equi-angular spacing via $\psi_i = \frac{2\pi i}{n}$. If the ith and jth incident curves form a single cross curve, we constrain the angles ψ_i and ψ_j to satisfy $\psi_i + \pi = \psi_j$. Due to this constraint, the angles can no longer be equally spaced, which is the reason why traditional techniques based on the Fourier transform can not always be used to pre-compute masks. We now compute the ψ_i by minimizing $\sum_i (\psi_{i+1} - \psi_i)^2$ subject to the previous angle constraints.

Given the ψ_i, we define $6+n$ eigenvectors of Z in four groups $Z_0\ Z_1\ Z_2\ Z_3$ with each group corresponding to eigenvalues 1, $1/2$, $1/4$ and $1/8$. The matrix Z_0 has a single column corresponding to the vector of ones. The matrices Z_1 and Z_2 have two and three columns, respectively, and depend on the angles ψ_i. In particular, the row of Z_1 and Z_2 corresponding to the ith vertex on the jth-ring of v has the form:

$$\begin{aligned} Z_1 &= \begin{pmatrix} j\cos(\psi_i) & j\sin(\psi_i) \end{pmatrix}, \\ Z_2 &= \begin{pmatrix} \beta(j) & \beta(j)\cos(2\psi_i) & \beta(j)\sin(2\psi_i) \end{pmatrix} \end{aligned} \tag{6}$$

where $\beta(j) = 0$ if $j = 0$ and $j^2 - \frac{1}{3}$ if $j > 0$. The matrix Z_3 has n columns. At the ith vertex of the j-ring of v, the kth column of Z_3 is $j^3 - j$ if $k = i$ and 0 otherwise. The coefficients $\beta(j)$ and $j^3 - j$ reproduce the quadratic and cubic eigenvectors (corresponding to the eigenvalues $\frac{1}{4}$ and $\frac{1}{8}$) of the subdivision matrix for the interpolating scheme of equation 3.

For valences $n \geq 5$, we now compute C using equations 5 and 6. The leftmost two examples of figure 4 shows close-ups of two valence five corners. However, for the $n < 5$ case, equation 6 defines *more* eigenvectors than we need. In this case, we select a linear subspace of the eigenvectors spanned by Z_2. Specifically, we compute the null space of Z_0, Z_1 and Z_3 as $Z_{013}^{\perp}$ and then project $Z_{013}^{\perp}$ onto Z_2. If Z_2^+ is the pseudo-inverse of Z_2, this resulting projection is $Z_2(Z_2^+ Z_{013}^{\perp})$. Finally, we compute C using equation 5 with this subspace in place of Z_2.

For $n = 4$, this construction reproduces the subdivision matrix C of equation 4. In the symmetric case where $\psi_i = \frac{2\pi i}{n}$, we can pre-compute C explicitly as a block circulant matrix. The attached appendix states these rules and uses them in the smoothness analysis for the resulting lofted surface.

To understand the behavior of C in the neighborhood of a

corner v, we parameterize the tangent plane at v defined by the two eigenvectors in Z_1 via $(x_1, x_2) = r(\cos(\psi), \sin(\psi))$. Under this parameterization, the first column of Z_1 defines a curve network lying on the linear function x_1 while the second column of Z_1 defines a curve network lying on the linear function x_2. Viewed in this informal manner, the limit networks associated with the three columns of Z_2 lie on the quadratic functions,

$$x_1^2 + x_2^2, \quad x_1^2 - x_2^2, \quad \tfrac{1}{2} x_1 x_2, \tag{7}$$

respectively. Similarly, the limit curves associated with the columns of Z_3 lie on n piecewise cubic functions that are C^2 at v.

4.3. Rules for corners with incident crease curves

At a corner, the incident crease curves divide the network into sectors bounded by consecutive crease curves. Our goal is to construct subdivision rules for the crease curves as well as rules for other curves that lie on the interior of a sector. Boundary curves are simply treated as crease curves.

For crease curves terminating at a corner v, we use the uniform rules of equation 2 on the interior of the curve and interpolate v. These rules converge to natural cubic splines (i.e; cubic curves whose second derivative at v is zero). For two crease curves crossing v, we use the interpolatory rules of equation 3. Note that the subdivision rules for crease curves depend only the vertices on the crease curves (as in standard creased constructions for subdivision surfaces).

The construction of the subdivision matrix C for each sector also starts with the eigenvector matrix Z constructed as in section 4.2. On the interior of a sector, the subdivision rules depend on vertices of the bounding crease curves as well as vertices of curves interior to the sector. The subdivision rules $\bar{C}$ for these interior (smooth) curves have the form

$$\bar{C} = \bar{Z} \Lambda Z^{+} \tag{8}$$

where $\bar{Z}$ consists of rows in Z corresponding to the interior vertices of these smooth curves. As in the case of no creases, these subdivision rules force the limit curves in this sector to lie on a common C^2 surface; however, these surfaces can be different for different sectors.

The second example from the right in figure 4 shows a valence five corner in which two non-adjacent curves form a single crease curve crossing v. These two crease curves form two sectors of width π. For the upper sector, the rules for $\bar{C}$ reduce to the uniform rules of equation 2. For the lower sector, the matrix $\bar{C}$ has the form

$$\begin{pmatrix} \frac{9}{16} & \frac{1}{32} & \frac{3}{8} & \frac{1}{8} & \frac{-3}{32} & 0 & 0 & 0 & 0 \\ \frac{9}{16} & \frac{-3}{32} & \frac{1}{8} & \frac{3}{8} & \frac{1}{32} & 0 & 0 & 0 & 0 \\ \frac{9}{64} & \frac{1}{128} & \frac{23}{32} & \frac{1}{32} & \frac{-3}{128} & 0 & \frac{1}{8} & 0 & 0 \\ \frac{9}{64} & \frac{-3}{128} & \frac{1}{32} & \frac{23}{32} & \frac{1}{128} & 0 & 0 & \frac{1}{8} & 0 \end{pmatrix}$$

on the two-ring of v. The four rows of $\bar{C}$ define subdivision rules for two interior vertices on each of the smooth curves incident on v.

If the width of the sector is less than π, a simpler alternative to equation 8 is to use uniform cubic rules on each smooth curve incident on v. These rules are more flexible at the corner v, but are not guaranteed to converge to a common tangent plane. The rightmost example of figure 4 shows an example in which three crease curves meet at v.

5. Lofted subdivision surfaces

Given the subdivision scheme **C** for curve networks as defined in the previous section, we next construct a modified version of Catmull-Clark subdivision **S** that commutes with **C** via equation 1 where **M** generalizes the tensor product case to extraordinary vertices. Given this generalized **M**, we then explicitly solve for this modified Catmull-Clark scheme **S** using a block decomposition of local subdivision matrices corresponding to **C**, **M** and **S**.

5.1. Generalizing M

In the tensor product cases of section 3, the operator **M** had the property that the limit curve q^∞ was interpolated by the surface p^∞ if $q^0 = \mathbf{M} p^0$. If the vertex v was not a corner vertex of q^0, **M** computed the position of the control points of the curve network by applying the mask $(\frac{1}{6}\ \frac{2}{3}\ \frac{1}{6})$ to p^0 with the mask centered at v and oriented across the curves (upper left of figure 2). If v was a corner of q^0, **M** applied the 3×3 limit mask for bicubic subdivision at v (upper right of figure 2).

For general quad meshes p^k, we next define the behavior of **M** at a vertex whose surface valence is other than four. At an interior vertex v of q^k, **M** applies the mask shown in the lower left of figure 2 to p^k. This mask is based on the number of unlofted edges n_1 and n_2 on each side of q^k. If $n_1 = n_2 = 1$, M degenerates to the mask for the regular case.

At a corner of q^0 with surface valence n, we define **M** to be the limit mask for repeated averaging Catmull-Clark subdivision, shown in lower right of figure 2. When $n = 4$, this mask degenerates to that of the regular case. In the presence of crease curves, we modify **M** to be interpolatory on all vertices lying on creased polylines. The effect of this choice on the resulting lofted surface will be to decouple the subdivision rules along creased curves from the rest of the surface.

5.2. Lofted subdivision via block decomposition

Given this choice for **M**, we can now solve for **S** using equation 1. If we constrain the rules for **S** to reproduce those of bicubic subdivision off of the curve networks, the surface rules for vertices on the curve network are uniquely determined. To make the construction of the rules more explicit, we introduce the following notation. Let p^k be the vector of control points of the mesh after k subdivision steps. p^k can be partitioned into two subvectors: curve network control points p_c^k and all other control points p_n^k. The subdivision schemes and change of basis operators can be expressed as global matrices acting on these vectors. Let S^k be the matrix for the surface subdivision scheme, $p^{k+1} = S^k p^k$, let C^k

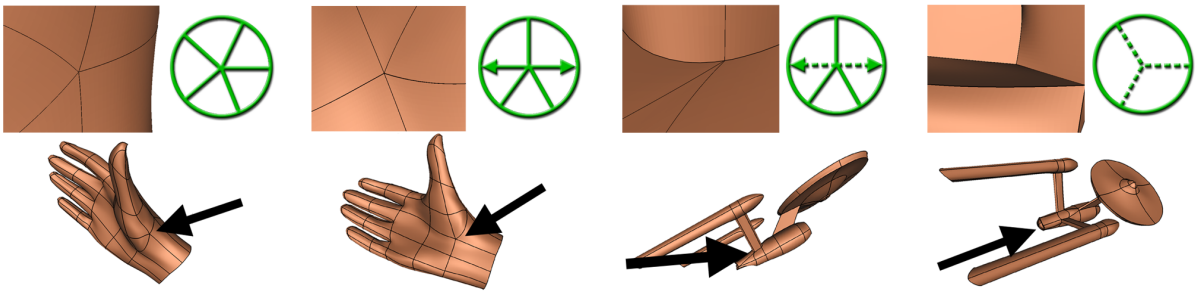

Figure 4: *Four corner configurations arising during realistic modeling. The upper left pictures are close-ups of both the network and the lofted surface. The upper right diagram depicts the layout of the ψ_i. Dotted edges are creases. Pairs of arrows denote cross curves.*

be the matrix for the curve subdivision scheme, and M^k the change of basis matrix. The subdivision scheme $\mathbf{S}$ can be expressed in block form as $p_c^{k+1} = S_c^k p^k$ and $p_n^{k+1} = S_n^k p^k$. If the change of basis matrix M^k is also expressed in block form as $(M_c^k M_n^k)$, so that $q_c^k = M_c^k p_c^k + M_n^k p_n^k$ the commutative relation of equation 1 has the block form

$$C^k M^k = \begin{pmatrix} M_c^{k+1} & M_n^{k+1} \end{pmatrix} \begin{pmatrix} S_c^k \\ S_n^k \end{pmatrix}.$$

We note that M_c^k is a square matrix, as it acts on mesh control points p_c^k corresponding to the vertices of the polyline network and produces new values for the same vertices. Furthermore, as explained below, M_c^k has an inverse. Solving for the modified surface subdivision operator S_c^k in terms of C^k, M^k and S_n^k yields

$$S_c^k = (M_c^k)^{-1}(C^k M^k - M_n^{k+1} S_n^k). \tag{9}$$

This equation not only defines our surface scheme, but can be used to apply this scheme to a mesh. The local masks for M^k, M_n^{k+1}, S_n^k and C^k are already defined: M^k and M_n^{k+1} explicitly in this section, S_n^k by the standard Catmull-Clark rules and C^k in Section 4. The only remaining transformation $(M_c^k)^{-1}$ repositions vertices of the polyline network. Fortunately, as it can be easily verified directly, $(M_c^k)^{-1}$ has the same support as M_c^k. At a corner vertex v, $(M_c^k)^{-1}$ is defined by a simple mask: it repositions v to lie at $\frac{9}{4}$ of its current position minus $\frac{3}{2}$ of the centroid of its edge neighbors. At an interior vertex v, $(M_c^k)^{-1}$ simply scales the vertex position by $\frac{3}{2}$.

Analysis. At interior vertices v of q^k, the resulting rules produce by our surface scheme reduce to those of uniform bicubic B-splines when $n_1 = n_2 = 1$. In this case, the scheme is C^2 at v. At a corner vertex v of q, the subdivision rules reduce to those of bicubic B-splines when $n = 4$. Again, the scheme is C^2 at v. More generally, the resulting subdivision rules differ from those of standard Catmull-Clark in the two-ring of corner vertices. If $\psi_i = \frac{2\pi i}{n}$, our modified Catmull-Clark scheme has a spectrum of the form $1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \ldots$ and converges to surfaces that are C^1 with bounded curvature at v. The attached appendix contains an explicit construction for these rules as function of the valence n and outlines our smoothness proof.

Unfortunately, proving any type of smoothness result in more general cases is very difficult. One reassuring fact is that the commutative relation of equation 1 ensures that any eigenvalues of C are also eigenvalues of S. Thus, at corner vertex of q^0, the spectrum of S includes the eigenvalues $1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \ldots$. However, this condition is not sufficient to ensure the correct spectrum for the resulting surface scheme since lofting inserts extra eigenvalues into the spectrum of S. However, our experience has been that the lofted scheme produces visually smooth surfaces in all configurations where corner vertices are fully lofted (i.e; all surfaces edges incident on the corner are lofted). Figure 4 shows four close-ups of lofted subdivision surfaces near a corner vertex.

6. Automated lofting of curve networks

In the previous sections, we have constructed subdivision schemes for curve networks and surfaces that loft these networks. Now, given a network of polylines q^0, we first describe a skinning method for constructing the topology of a base mesh p^0 that interpolates q^0. We then fair the positions of vertices of p^0 subject to the constraint that $q^0 = Mp^0$.

6.1. Skinning

Our skinning algorithm constructs mesh connectivity for the interior of each patch specified by the designer. If we merge (in cyclic order) the open polylines bounding the patch, the result is a closed polyline q. Our task is to form a quadrangulation of q, i.e; a quad mesh whose boundary is exactly q.

While many quadrangulation and even more triangulation algorithms have been proposed in mesh generation literature, the task of generating suitable meshes for subdivision surfaces is quite different. For standard mesh generation, the main goal is to maintain good quad or triangle aspect ratios and/or approximate a given shape well. For subdivision surfaces, methods such as Nasri [NAH03] seek to generate

meshes whose topologies minimize the number and valence of extraordinary vertices.

Theoretical bounds. We have designed a simple algorithm for generating a quadrangulation p of q that optimizes the valence of added extraordinary vertices and their number. If we let $val(p_i)$ denote the edge valence of a vertex p_i of p, the following proposition shows the fundamental limitation on what can be achieved.

Proposition 1 Let q be a closed polyline formed as the union of k open polylines. If the length of q is even, any quadrangulation p of q satisfies

$$\sum_i |val(p_i) - v_i| \geq |k-4| \tag{10}$$

where $v_i = 2$ if p_i is a corner vertex of q, $v_i = 3$ if p_i is an interior vertex of q and $v_i = 4$ otherwise.

This proposition has several important consequences. Ideally, we desire a quadrangulation p with one quad incident on each corner of q, two quads incident on an interior vertex of q and four quads meeting at every interior vertex of p. This situation would make the left-hand side of equation 10 zero. Unfortunately, this proposition forces p to have extraordinary vertices when $k \neq 4$. Second, as the valence of these extraordinary vertices decreases, the number of extraordinary vertices must increase. In particular, if only vertices of valence five or less are allowed, the resulting mesh must have at least $|k-4|$ extraordinary vertices.

For $k = 4$, generating a quadrangulation with no extraordinary vertices is possible in many cases. However, in some cases, creating extraordinary vertices is necessary. A simple example is shown in Figure 5. In this case, it is easy to show that there can be no regular quadrangulation, and therefore at least one vertex of valence greater than four and one vertex of valence less than four needs to be introduced.

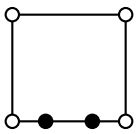

Figure 5: *A patch with four corners which cannot be quadrangulated without adding extraordinary vertices. Empty circles mark the corner vertices, i.e. vertices where multiple curves meet, filled circles mark interior boundary vertices.*

Our algorithm takes the extreme approach and generates only extraordinary vertices of valence three and five since the quality of the resulting surface is the closest to that of the regular case. One can argue that in some cases better results can be achieved by using higher valence vertices (e.g. up to 7); such extensions are easy to add to our basic algorithm. Moreover, our algorithm always stays close to the optimal number of extraordinary vertices. Specifically, for $k \geq 4$, our method generates a quadrangulation with no more than $k-2$ vertices of valence five, no more than 2 vertices of valence three (plus an extra triangle or pentagon for odd length q).

Overview. Given a closed polyline $q = (q_1, ..., q_n)$, we define a *chain* of length m to be a subsequence $(q_i, \ldots, q_{i+m})$ of q corresponding to a single open polyline with corners at q_i and q_{i+m}. (Note that index arithmetic on q is performed mod n.) Our quadrangulation method constructs p by performing a sequence of *chain advances* on q. Given a chain $(q_i, \ldots, q_{i+m})$, this chain advance adds a layer of m quads to p bounded below by $(q_i, \ldots, q_{i+m})$ and above by $(q_{i-1}, p_j, \ldots, p_{j+m-1}, q_{i+m+1})$ where $p_j, \ldots p_{j+m-1}$ are new vertices that lie on the interior of the final patch p. The polyline q is then replaced by the polyline

$$(q_1, \ldots, q_{i-1}, p_j, \ldots, p_{j+m-1}, q_{i+m+1}, \ldots, q_n).$$

Our quadrangulation method performs a sequence of chain advances that can be partitioned into two phases.

- Phase one performs two types of chain advances and terminates when all but one of the chains in q have length one.
- Phase two performs two different types of chain advances and terminate when the length of q is five or less. At this point, the method forms a single polygon from q and adds it to p.

Phase one. In the first phase, our method repeatedly applies two types of chain advances. Type one advances involve a single chain $(q_i, \ldots q_{i+m})$ where q_{i-1} and q_{i+m+1} are interior vertices (see figure 6 left). These advances introduce no extraordinary vertices and leaves the number of chains in q unchanged. Type two advances involve a sequence of chains of length one $(q_i, \ldots, q_{i+m})$ where q_{i-1} and q_{i+m+1} are interior vertices (see figure 6 right). These advances introduce $m-1$ valence five vertices while decreasing the number of chains in q by $m-1$.

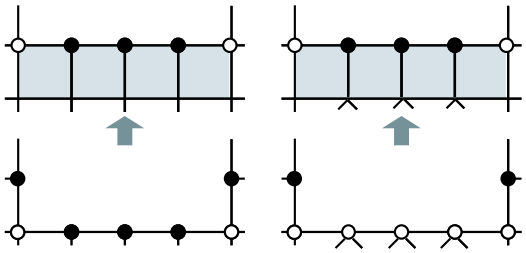

Figure 6: *Phase one chain advances. Left: type one. Right: type two. Empty circles indicate corner vertices before the advance, filled circles indicate interior chain vertices.*

Phase one applies these chain advances until all but one of the chains in q have length one. During this phase, there are many different type one and type two advances possible. Type one advances have priority over type two advances. In most cases, several type one advances are possible. While any choice of type one advance is permissible, we use the following heuristic to select among various type one advances.

If q consists of k chains, we let the vector $l = (l_1, \ldots l_k)$ denote the length of the chains of q. Applying $d = (d_1, \ldots d_k)$

type one advances to each chain of q yields a new polygon whose chains have length $l - Hd$ where H is a matrix whose ith row has ones in position $i-1$ and $i+1$ and zero otherwise. Since our ultimate goal is to choose d such that $l - Hd$ is zero, we compute $d = H^+ l$ and then advance on the maximal entry of d.

If no type one advances are possible, q must contain a chain of length one. In this case, we perform a type two advance of the shortest continuous sequence of chains of length one.

Phase two. At the start of phase two, all but one of the chains in q have length one. Without loss of generality, we assume that q has length n with its first chain having length $m \geq 1$. In the second phase, we perform two different types of chain advances. Type three advances involves the chain $(q_1, \ldots q_{m+1})$ (see figure 7 left). This advance creates valence five vertices at q_n and q_{m+2} while decreasing the number of chains by two. Type four advances involve the chain (q_n, q_1) (see figure 7 right) . If $m < n-1$, this advance creates a valence five vertex at q_{n-1} and decreases the number of chains by one. If $m = n-1$, this advance creates no valence five vertices since q_{n-1} is an interior vertex of the chain $(q_1, \ldots q_n)$.

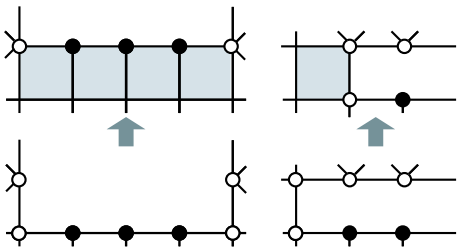

Figure 7: *Phase two chain advances. Left: type three. Right: type four. Empty circles indicate corner vertices before the advance, filled circles indicate interior chain vertices.*

Phase two applies type three advances to q until $n \leq 2m+3$. Next, phase two applies type four advances to q until q has five or fewer vertices. Phase two concludes by generating a single triangle, quad or pentagon from this final q. Figure 8 shows several examples of quadrangulations created by our method.

To summarize our valence bounds, for $k = 3$, the method generate a quad mesh with either a single valence three vertex (q has even length) or an extra triangle (q has odd length). For $k \geq 4$, the four types of chain advances generate one valence five vertex for each chain eliminated. Since up to $k-2$ chains may be eliminated, the final mesh has up to $k-2$ valence five vertices. The final polygon constructed at the end of phase two may introduce up to two vertices of valence three in the mesh. Thus, our method generates a quadrangulation with no more than $k-2$ vertices of valence five, no more than 2 vertices of valence three (plus an extra triangle or pentagon for odd length q).

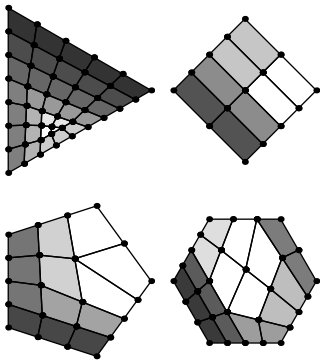

Figure 8: *Various quadrangulations produced by our method. Successive phase one chain advances are shaded from dark to light.*

In some situations, our quadrangulation method introduces extraordinary vertices on the boundary of p. If the length of all chains in q is three or more, we can add a preliminary phase zero to our method that moves all extraordinary vertices to the interior of p. This phase consists of k type one advances, one per chain in q. These advances add a single ring of quads to p with no extraordinary vertices on the boundary of the final p. For meshes of this type, our lofting method produces surfaces that are provably C^1 at the corners of q and C^2 elsewhere on q.

6.2. Fairing

Having computed the topological structure for each patch in the base mesh p^0, we next compute positions for the vertices of p^0 such that the limit surface p^∞ is fair and interpolates the curve network. In this framework, our task is to optimize the shape of the surface p^0 by minimizing a fairness functional $E(p)$ subject to the constraint that $q^0 = Mp^0$, which guarantees that the curves are interpolated.

There is a substantial amount of work on fairing meshes; for subdivision surfaces fairing was first considered in [HKD93], where the goal is to construct Catmull-Clark subdivision surfaces interpolating the control points. In that paper the functional is evaluated on the surface, which is more reliable but also quite computationally expensive and requires considerable effort to implement. We have found that a much simpler approach based on fairing the base mesh p^0 itself yields good results.

Our implementation uses a thin plate functional defined as follows:

$$E(p) = \sum_{p_i} \left(\sum_{p_j \in N_i} \alpha_j^i p_j \right)^2 + \left(\sum_{p_j \in N_i} \beta_j^i p_j \right)^2 + \left(\sum_{p_j \in N_i} \gamma_j^i p_j \right)^2$$

where N_i is the one-ring of the vertex p_i, and for noncrease

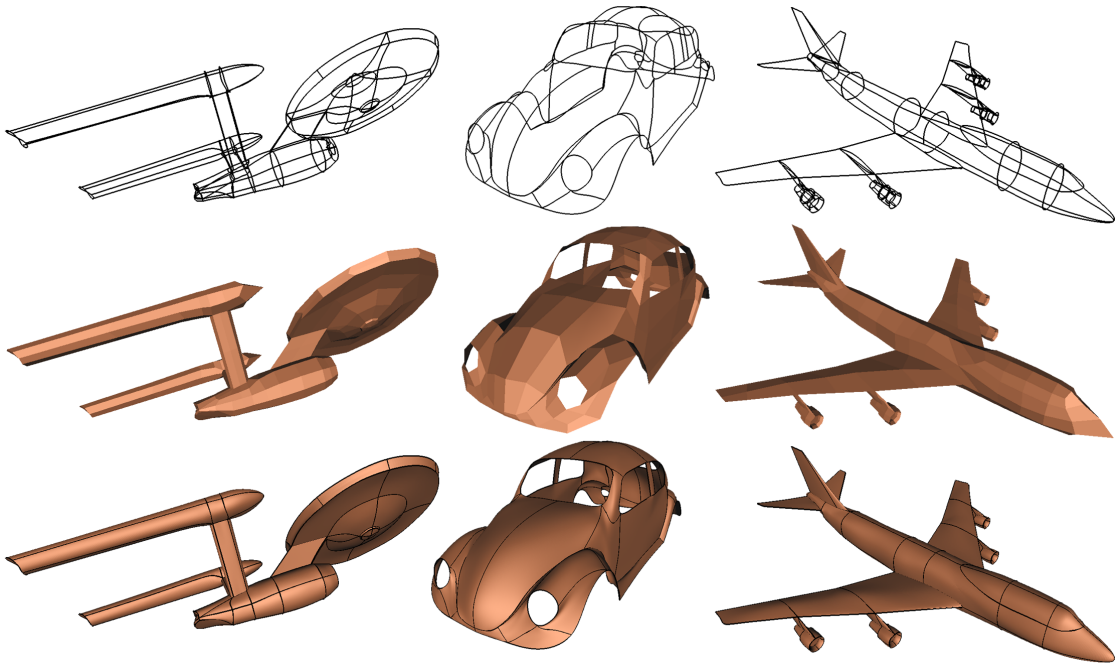

Figure 9: *Examples of surfaces lofting curve networks. Each surface was computed automatically using skinning and fairing.*

vertices p_i, of valence $k_i \neq 4$, and $j \neq i$,

$$\alpha_j^i = \frac{2}{k_i}\cos(\frac{4\pi j}{k_i}), \quad \beta_j^i = \frac{2}{k_i}\sin(\frac{4\pi j}{k_i}), \quad \gamma_j^i = \frac{1}{k_i},$$

and $\alpha_i^i = \beta_i^i = 0$, $\gamma_i^i = -1$. For $k_i = 4$, α_j^i and β_j^i should be $\alpha_j^i = \frac{1}{4}\cos(\pi j)$ and $\beta_j^i = \frac{1}{4}\sin(\pi j)$. This functional, up to a scale factor, can be viewed as a finite difference approximation of $\int F_{uu}^2 + F_{vv}^2 + 2F_{uv}^2 dudv$ for a particular choice of local parameterizations.

This functional is known to be far from optimal for fine meshes; generally speaking, a nonlinear functional formulated in terms of curvature approximations is likely to yield somewhat better results. However, the difference on relatively coarse meshes, such as the typical control meshes of subdivision surfaces, appears to be less significant. The great advantage of this functional in our setting is that it is quadratic and can be written in the form $p^T Ap$, where A is a symmetric matrix. Therefore, vertex positions can be computed without a good initial guess. These positions can then be used as an initial guess for a nonlinear optimization procedure.

We minimize this functional with constraints using the standard Lagrange multiplier approach, i.e. solving

$$\nabla_p E(p) + M^T\lambda = Ap + M^T\lambda = 0; \quad Mp = q^0$$

where λ is the vector of Lagrange multipliers, one per curve vertex. We note that this is a symmetric but not a positive definite linear system, so one has to be careful when using an iterative method to solve it: the standard Conjugate Gradient method does not apply, Conjugate Residuals or another more general method need to be used. For small numbers of control points (up to several hundreds) a direct solver is acceptable. Figure 9 shows three examples of lofted surfaces constructed from polyline network by our automatic method.

7. Future work

We have noted some partially lofted configurations (where not all surface edges incident on a corner vertex are lofted) in which the resulting subdivision surfaces are only C^0, not C^1. While our skinning method does not generate such partially lofted configurations if all polylines in q^0 contain at least two interior vertices, we believe that it may be possible to modify the change of basis M to yield lofted surfaces that are C^1 for all configurations including partially lofted ones. The recently developed smoothness analysis techniques of [Uml03] may aid in this process.

We also note that the surface rules produced by equation 9 are very general. In particular, this technique can be used to modify the subdivision rules for other schemes such as Loop's triangular scheme to loft networks of curves. For example, triangular C^2 quartic box-splines can be used to interpolate a network of C^2 quartic splines. (The restriction of a C^2 quartic box spline to a single grid line is a C^2 quartic spline.) At extraordinary vertices of the curve network, equation 9 defines perturbed versions of the box spline rules that allow the resulting surface to interpolate arbitrary number of curves meeting at the extraordinary vertex. This topic is an important area for further research.

8. Acknowledgements

This research was supported in part by NSF awards CCR-0093390, DMS-0138445, ACI-997814, ITR-0205671 and IBM University Partnership Award.

References

[BLZ00] BIERMANN H., LEVIN A., ZORIN D.: Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 113–120. 5

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided Design 10*, 6 (1978), 350–355. 1

[CK83] CHIYOKURA H., KIMURA F.: Design of solids with free-form surfaces. *Computer Graphics 17* (1983), 289–298. 1

[DS78] DOO D., SABIN M.: Behavior of recursive division surfaces near extraordinary points. *Computer-aided Design 10*, 6 (1978), 356–360. 5

[Her96] HERMANN T.: G^2 interpolation of free form curve networks by biquintic Gregory patches. *Comput. Aided Geom. Design 13*, 9 (1996), 873–893. In memory of John Gregory. 1

[HKD93] HALSTEAD M., KASS M., DEROSE T.: Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of SIGGRAPH 93* (Aug. 1993), Computer Graphics Proceedings, Annual Conference Series, pp. 35–44. 9

[Lev99] LEVIN A.: Interpolating nets of curves by smooth subdivision surfaces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 57–64. 2

[NA02] NASRI A., ABBAS A.: Lofted catmull-clark subdivision surfaces. In *Proceedings of Geometric Modeling and Processing International* (2002), pp. 83–93. 1

[NAH03] NASRI A., ABBAS A., HASBINI I.: Skinning catmull-clark subdivision surfaces with incompatible cross-sectional curves. In *Proceedings of Pacific Graphics* (2003), pp. 102–111. 2, 8

[Nas97] NASRI A.: Curve interpolation in recursively generated b-spline surfaces over arbitrary topology. *Computer Aided Geometric Design 14*, 1 (1997), 13–30. 1

[Nas00] NASRI A. H.: Recursive subdivision of polygonal complexes and its applications in computer-aided geometric design. *Computer Aided Geometric Design 17*, 7 (Aug. 2000), 595–619. 1

[Nas03] NASRI A.: Interpolating an unlimited number of curves meeting at extraordinary points on subdivision surfaces. *Computer Graphics Forum 22*, 1 (2003), 87–97. 1

[NKL01] NASRI A., KIM T., LEE K.: Fairing recursive subdivision surfaces with curve interpolation constraints. In *Proceedings of the Internation Conference on Shape Modeling and Applications* (2001), pp. 49–61. 1

[PU01] PETERS J., UMLAUF G.: Computing curvature bounds for bounded curvature subdivision. *Computer Aided Geometric Design 18* (2001), 455–461. 12

[TS90] T. SAITOH M. H.: Interpolating curve networks with new blending patches. In *Eurographics* (1990), pp. 137–146. 1

[Uml03] UMLAUF G.: The characteristic map of non-box spline subdivision schemes. Presentation at the Eighth SIAM Conference on Geometric Design, November 2003. 10

[Vár91] VÁRADY T.: Overlap patches: a new scheme for interpolating curve networks with n-sided regions. *Comput. Aided Geom. Design 8*, 1 (1991), 7–27. 1

[WW01] WARREN J., WEIMER H.: *Subdivision Methods for Geometric Design*. Morgan Kaufmann, 2001. 11

Appendix A: Explicit curve and surface rules

In the rotationally symmetric case where $\psi_i = \frac{2\pi i}{n}$, the curve subdivision matrix C can be expressed in a block form induced by partitioning q^0 into rings around the corner v as shown in figure 10. Specifically, C can be partitioned such that the block C_{11} is a scalar, the blocks C_{i1} are columns vectors of constants of length n, the blocks C_{1j} are row vectors of constants of length n and the blocks C_{ij} where $i, j > 1$ are $n \times n$ circulant matrices. In this *modified block circulant* form, each circulant matrix C_{ij} can be encoded as a polynomial $c_{ij}[z]$ whose coefficients form the first row of C_{ij}. Collecting these polynomials $c_{ij}[z]$ yields a *matrix polynomial* $c[z]$ that compactly encodes C.

One advantage of matrix polynomials is that computing the spectral structure of their associated block circulant matrices is easy. Given a matrix polynomial $c[z]$, let $\hat{c}[z]$ be the submatrix formed by eliminating the first row and column of $c[z]$. As shown in section 8.3 of Warren and Weimer [WW01], the eigenvalues of C are the eigenvalues of $c[1]$ and the eigenvalues of $\hat{c}[\omega^j]$ where ω is the n^{th} root of unity and $1 \leq j \leq n-1$.

If $\psi_i = \frac{2\pi i}{n}$, the matrix polynomial $c[z]$ for the subdivision matrix C of equation 5 has the form

$$c[z] = \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{4} & \frac{1}{4}\eta^0[z] + \eta^1[z] + \frac{\alpha_n}{2}\eta^2[z] & 0 \\ \frac{3}{16} & \frac{11}{16}\eta^0[z] + \frac{3}{2}\eta^1[z] + \frac{11\alpha_n}{8}\eta^2[z] & \frac{1}{8} \end{pmatrix} \quad (11)$$

where the polynomials $\eta^j[z]$ have the form $\frac{1}{n}\sum_{i=0}^{n-1}\cos(\frac{2\pi ji}{n})z^i$ and the constant α_n is 0 if $n = 3$, $\frac{1}{2}$ if $n = 4$ and 1 for $n \geq 5$. Using the fact that the polynomials $\eta^j[z]$ satisfy $\eta^j[\omega^i] = 0$ when $i \neq j, n-j$ allows easy computation of the eigenvalues of C. As expected, C has the spectrum $1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \ldots$.

Another advantage of matrix polynomials is that we can compute the matrix polynomial $s[z]$ for our lofted surface scheme by applying equation 9 to the appropriate matrix polynomials. Before proceeding, we must first form $m[z]$. The operator M maps surface vertices in the two-ring of v to

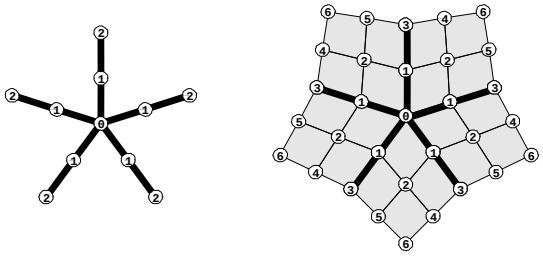

Figure 10: *Block ordering for vertices on a curve network (left) and its corresponding surface mesh (right)*

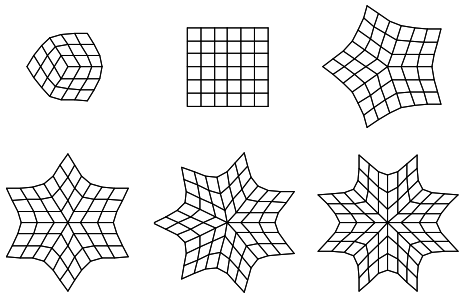

Figure 11: *Characteristic map of the surface scheme for $n = 3 \ldots 8$*

curve vertices in the two-ring of v using the ordering of figure 10. At the corner vertex v, M computes $\frac{4}{9}$ of the central vertex plus $\frac{4}{9n}$ times each edge-adjacent surface vertex plus $\frac{1}{9n}$ times each face-adjacent surface vertex. We encode this transformation as the first row of matrix polynomial $m[z]$.

$$m[z] = \begin{pmatrix} \frac{4}{9} & \frac{4}{9} & \frac{1}{9} & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & \frac{1}{6}+\frac{1}{6z} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{3} & \frac{1}{6z} & \frac{1}{6} & 0 \end{pmatrix}$$

Note the entries $m_{1j}[z]$ for $j > 1$ are implicitly multiplied by $\frac{1}{n}$ so that the rows of the matrix polynomial sum to one when evaluated at $z = 1$. Similarly, the polynomial $M_{23}[z] = \frac{1}{6} + \frac{1}{6z}$ encodes a circulant matrix whose first row is $(\frac{1}{6} 0 \ldots 0 \frac{1}{6})$.

We next compute the modified subdivision rules $s_c[z]$ by applying equation 9 to the matrix polynomials $s[z]$, $m[z]$ and $s_n[z]$. The fundamental observation underlying this construction is that multiplying two block circulant matrices is equivalent to multiplying their matrix polynomials. For modified block circulant matrices, a similar construction is possible subject to the restriction that entries of the matrix polynomial corresponding to scalar, row vectors or column vectors in the modified block circulant matrix are treated appropriately. In particular, the product of a column vector times a row vector is a circulant matrix whose entries are constants. Thus, modeling multiplication of modified block circulant matrices using matrix polynomials requires introducing multiples of $\eta^0[z]$ into the resulting product to model this effect. With this caveat, we can compute $c[z]m[z] - m_n[z]s_n[z]$ where $s_n[z]$ has the form

$$s_n[z] = \begin{pmatrix} \frac{1}{4} & \frac{1}{4}+\frac{z}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{16}+\frac{3z}{8} & \frac{3}{8} & \frac{z}{16} & \frac{1}{16} & 0 & 0 \\ \frac{1}{16} & \frac{3}{8}+\frac{z}{16} & \frac{3}{8} & \frac{1}{16} & 0 & \frac{1}{16} & 0 \\ \frac{1}{64} & \frac{3}{32}+\frac{3z}{32} & \frac{9}{16} & \frac{1}{64}+\frac{z}{64} & \frac{3}{32} & \frac{3}{32} & \frac{1}{64} \end{pmatrix}.$$

Multiplying $(c[z]m[z] - m_n[z]s_n[z])$ by the inverse of $m_c[z]$ yields the matrix polynomial $s_c[z]$ for our modified Catmull-Clark scheme of the form

$$\begin{pmatrix} \frac{9}{16} & \frac{3}{8} & \frac{1}{16} & 0 & 0 & 0 & 0 \\ \frac{3}{8} & s_{22}[z] & s_{23}[z] & 0 & 0 & 0 & 0 \\ \frac{3}{32} & s_{42}[z] & s_{43}[z] & \frac{3}{32} & \frac{1}{64z} & \frac{1}{64} & 0 \end{pmatrix}$$

where the four modified entries $s_{ij}[z]$ are as given below

$$\begin{aligned} s_{22}[z] &= \tfrac{1}{16}(12\eta^0[z] + 16\eta^1[z] + 8\alpha_n\eta^2[z] - \tfrac{1}{z}(1+z)^2), \\ s_{23}[z] &= \tfrac{1+z}{16z}(2\eta^0[z] + 4\eta^1[z] + 2\alpha_n\eta^2[z] - 1), \\ s_{42}[z] &= \tfrac{1}{64}(52\eta^0[z] + 96\eta^1[z] + 88\alpha_n\eta^2[z] - \tfrac{1}{z} - 12 - z), \\ s_{43}[z] &= \tfrac{1+z}{32z}(6\eta^0[z] + 12\eta^1[z] + 11\alpha_n\eta^2[z] - 3). \end{aligned}$$

The polynomials $\eta^j[z]$ and the constant α_n are defined in the same manner as for $c[z]$.

We can analyze the smoothness of the modified surface scheme using the matrix polynomial $s[z]$ formed by $s_c[z]$ and $s_n[z]$. The modified subdivision scheme has a spectrum of the form $1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \ldots$ where the eigenvalue $\frac{1}{4}$ has multiplicity 2 if $n = 3$, 3 if $n = 4$ and 4 for $n \geq 5$ and the remaining eigenvalues lie between $\frac{1}{4}$ and zero. Similarly, we have computed the eigenvectors associated with the subdominant eigenvalues $\frac{1}{2}$ symbolically as a function of the valence n. Figure 11 shows the characteristic map induced by these eigenvectors for valence $n = 3$ to 8. Using interval arithmetic in Mathematica in conjunction with the symbolic representation of these eigenvectors, we have proven that the characteristic map is regular and injective for all valences $n \geq 3$. Thus, the lofted subdivision surface is C^1 with bounded curvature at these vertices [PU01].

Appendix B: Proof of proposition 1

Consider a quadrangulated domain homeomorphic to a disk, with k corners (i.e. vertices of valence 2) on the boundary; By assumptions of the proposition, all other vertices on the boundary are valence 3. Let the sets of interior, edge and corner vertices be V_I, V_C and V_E respectively, and the set of all vertices be V. We call these remaining boundary vertices *edge vertices.* The total number of quads can be computed as $f = (1/4)(\sum_{v_i \in V_I} k_i + 2|V_E| + |V_C|)$, where $|X|$ is the number of elements in a set X. The total number of edges is $(1/2)(\sum_{v_i \in V_I} k_i + 3|V_E| + 2|V_C|)$. By the Euler formula,

$$\begin{aligned} &1 = |V| - e + f = |V_I| + |V_E| + |V_C| + \\ &\tfrac{1}{4}(\textstyle\sum_{v_i \in V_I} k_i + 2|V_E| + |V_C|) - \tfrac{1}{2}(\sum_{v_i \in V_I} k_i + 3|V_E| + 2|V_C|) \\ &= |V_I| - \tfrac{1}{4}(\textstyle\sum_{v_i \in V_I} k_i) + \tfrac{1}{4}|V_C| \end{aligned}$$

As $|V_C| = k$, the statement of the proposition follows.