Fast $\mathcal{C}^2$ Interpolating Subdivision Surfaces using Iterative Inversion of Stationary Subdivision Rules

Technical Report TR02-001, UNC-Chapel Hill

Andrew Thall
Medical Image Display and Analysis Group (MIDAG)
University of North Carolina at Chapel Hill

Abstract. This paper presents a simple iterative algorithm for computing an initial mesh which interpolates the vertices of a target base-mesh in the limit under Catmull-Clark subdivision rules. It uses only local 1-neighborhood information at each vertex of the target mesh, and runs in $\mathcal{O}(nm)$ time where n is the number of vertices in the base mesh and m is the number of iterations. Since m typically small gives interpolation to within acceptable tolerances, the algorithm is effectively $\mathcal{O}(n)$, requiring only constant work for each base vertex to achieve an almost-everywhere $\mathcal{C}^2$ interpolating surface. This algorithm is effective for interpolation of closed, two sided meshes and assumes non-singularity of the underlying global splitting matrix. While the argument presented is for Catmull-Clark subdivision, it may be trivially extended to include Loop subdivision of tri-meshes as well. This method may be modified in a quick-and-dirty fashion to interpolate boundary normals as well as positions at the desired mesh points; unlike more sophisticated methods, however, no smoothing is done—the surface will thus have "ripples", fine-scale surface perturbations not implied by a coarse medial sampling.

Keywords: surface fitting, mesh interpolation, subdivision surfaces, m-reps, medial modeling, skeletal modeling

1. M-rep Boundary Description by Interpolating Subdivision Surfaces

[This paper is an excerpt from Chapter 4 of *Deformable Solid Modeling via Medial Sampling and Displacement Subdivision*, the Ph.D. thesis (in preparation) of Andrew Thall at UNC-Chapel Hill. References to m-reps, sampled skeletal modeling primitives, refer to this work.]

A sampled medial skeleton must be fleshed out by a surface which can carry the fine-scale geometric information. The difficulties of fitting a boundary based on medial interpolation and exact Blum medial correspondence were discussed in Chapter 2. One alternative to continuous cm-reps is to use an implicit surface representation, implicitizing the medial radius function; this has some similarity to methods for rendering convolution surfaces and was explored by Fletcher [7]. (See Fig. 1.) Implicit representations simplify figure-subfigure blending, but have several drawbacks—they still require medial resampling, as above, and present difficulties in parameterizing the surface for medial correspondence and boundary displacement. The alternative to a medial approach is to use the coarse medial sampling alone to derive the boundary, fitting a surface to the medially-implied boundary involutes. Subdivision surfaces are ideal for these medially-implied boundaries for a number of reasons:

– they allow surfaces of varying mesh connectivity and topology, requiring less attention to special cases and continuity constraints than would spline-based surface patches;

– they can interpolate boundary positions and normals for the known involute positions of the medial atoms;

– they are a multiresolution surface representation, which fits in well with the multiscale modeling paradigm of m-reps;

– they are a subject of much current research—including research on CSG-style approximate Boolean operations—and are being implemented in graphics APIs and rendering hardware.

This chapter discusses the creation and use of interpolating subdivision surfaces for m-rep boundaries. Section 2 describes a new, iterative algorithm for interpolative subdivision and shows that it is equivalent to solving a linear system for an interpolating Catmull-Clark subdivision surface. This method is equally applicable to interpolating Loop subdivision on triangle meshes. The section also discusses methods to interpolate normals and gives an error metric (from the implied medial atoms) when normals are not interpolated. Section 2.3 presents a method for directly computing limit positions for irregular mesh vertices (i.e., with non-quadrangular face-neighbors), thus allowing the new interpolation technique to be used on general closed meshes. An appendix is included which discusses the subdivision algorithm and its implementation.

2. Iteratively interpolating subdivision surfaces

The classic uniform, stationary subdivision surfaces—Doo-Sabin and Catmull-Clark—are approximating subdivision techniques, as is Loop subdivision for tri-meshes.[1] Vertices at the coarsest level are linearly transformed at each iteration to new locations, approaching their corresponding points on the limit surface. They can be tranformed immediately to these limit points using a modified subdivision matrix, and this is generally done after the surface mesh

[1] A reminder on nomenclature: *stationary* subdivision means that the same subdivision rules are used at each subdivision level; *uniform* subdivision means that the same set of rules are used everywhere on the mesh. Non-stationary methods may be used to adjust surface normals, as below; non-uniform methods may be used for forming cusps, edges and corners, as per DeRose et al.

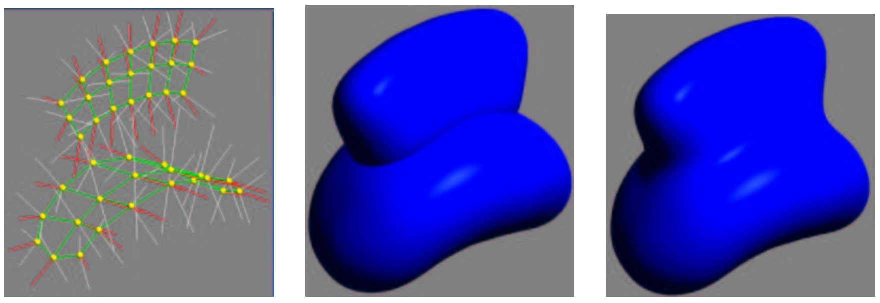


Figure 1. A figure-subfigure blend based on implicitizing the medial radius function. This uses interpolated medial positions as centers of Gaussian density fields, and thus differs from the implicit blending in Pablo (Fig. **??**) which is based on implicitizing the *boundary*, not the medial radius function. (Tom Fletcher)

has been subdivided to adequate fineness, often adaptively based on boundary curvature approximations.

For fairing of polyhedral objects—attempting to exactly fit a smooth surface to known vertices and vertex normals—other methods have also been developed. Jorg Peters[12], for example, did $\mathcal{C}^1$ interpolation using piecewise-bicubic patches for mesh-fitting, with linear normal-interpolation along patch boundaries. Implicit techniques such as those of Bajaj and Ihm[1] create algebraic patches of $\mathcal{C}^1$ continuity for closed polyhedra. Moreton and Séquin[11] employed a functional optimization approach to flesh surfaces based on point, normal, and curvature constraint sets.

Subdivision methods have risen to prominence, however, due to their conceptual simplicity, their equivalence to spline-based surfaces away from extraordinary points, and their requiring only minimal constraints on object-topology and mesh connectivity. Surface interpolation is attainable by several means. The Butterfly interpolation scheme of Dyn et al.[5] or the techniques of Zorin[18] can give $\mathcal{C}^1$ continuity on trimeshes subject to tension constraints or other parameters. Halstead, Kass, and DeRose[9] employed a modified Catmull-Clark technique to get interpolating subdivision of quadmesh structures, using thin-plate and membrane energies to constrain the subdivision and interpolate both positions and normals at the desired boundary points. The advantage of Catmull-Clark surfaces and others like them is that they are almost everywhere $\mathcal{C}^2$ and have closed-form limit positions and limit normals for vertices at any subdivision level. (See App. 3.)

2.1. Interpolating Catmull-Clark boundaries

For m-rep surface-fitting, a technique like that of Halstead et al. would be satisfactory but is in fact more exact than necessary—and pays for that exactness

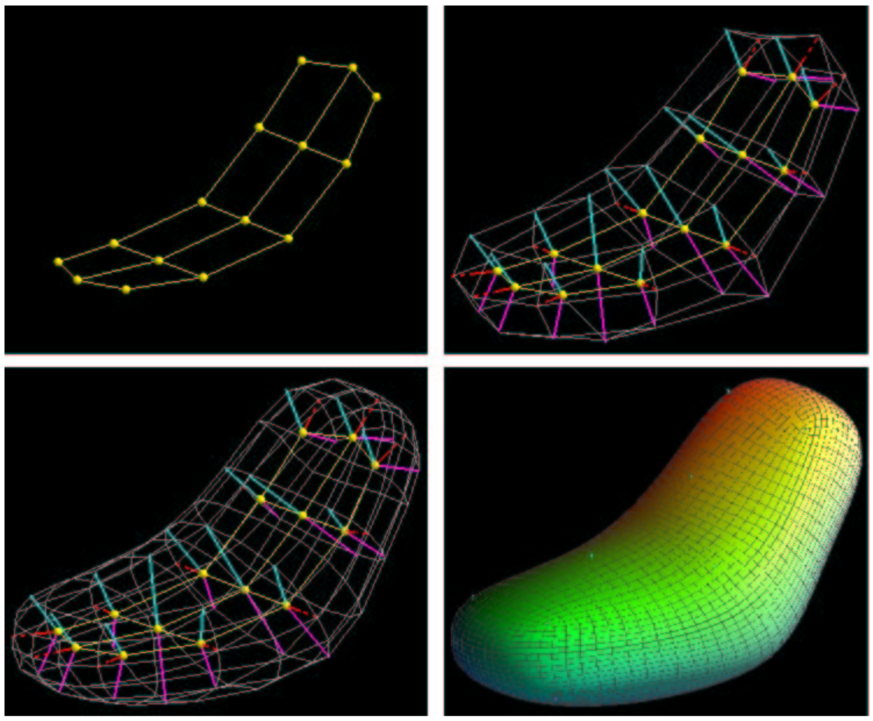


Figure 2. Medial mesh, boundary involutes and interpolating subdivision boundaries.

with its complexity and implementation details. What is needed instead is a subdivision surface which creates a limit surface with boundary positions and normals lying within the tolerance regions of the medial primitives. Such a surface should be $\mathcal{C}^2$ as well, allowing displacements in the normal direction to be limited by the radius of curvature in concave or saddle-shaped regions. Such a technique has been developed for m-reps boundaries by means of an iterative technique applied to the initial, coarse mesh of involute positions, which gives approximately interpolating subdivision boundaries within the desired tolerance. While the method has been applied to Catmull-Clark subdivision, it would be effective for any subdivision method where the limit masks can be simply computed for a local 1-neighborhood. Figures 2 and 3 show examples of this boundary-fitting.

The technique I developed for m-rep boundaries, like that of Halstead, involves solving the linear system for an initial subdivision grid which will produce limit positions interpolating the required boundary positions. Unlike Halstead, however, it uses an iterative solution method, requiring only 1-neighborhoods of mesh vertices and producing successively closer approximations to the involute positions by an algorithm which is $\mathcal{O}(m \cdot n)$, where n is the number of vertices being interpolated and m is the number of iterations. In practice, with m small, this effectively adds a constant cost per coarse-level

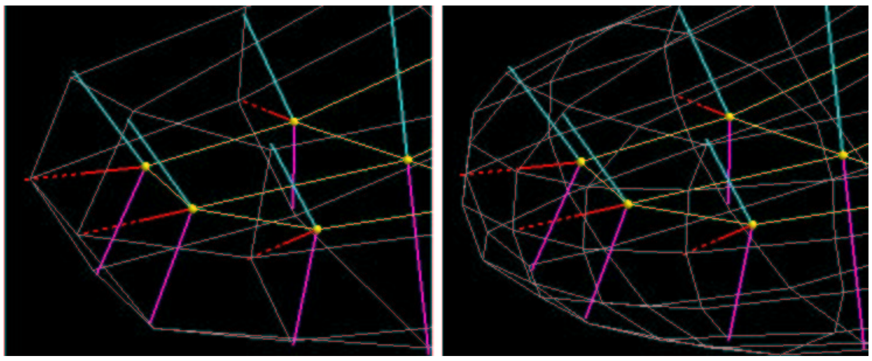


Figure 3. *Near*-interpolation of medial involutes by subdivision boundary.

mesh-vertex over the cost of non-interpolated subdivision using the same mesh. The rest of this section will detail the theory and practice of *iterative interpolating subdivision surfaces* (*IIS-surfaces*).

2.2. IIS-SURFACES—REGULAR VERTICES

Consider first an initial *regular* vertex mesh—one with only quadrangular faces. The iteratively interpolating subdivision surface (*IIS-surface*) is produced by creating a Catmull-Clark surface based on a modified initializing vertex grid; this modified grid is created by the following algorithm:

```
1. Initialize a boundary mesh $v_n^0$ with vertices $v_n^{\text{limit}}$ of positions to be interpolated
2. For iteration $i = 1$ to $m$
3.        For each vertex $v_j^i$ and its immediate edge $e_{j\,k}^i$ and face $f_{j\,k}^i$ neighbors
                [$k$ indexing into neighborhood (6- or 8-, typically) of $v_j^i$]
4.            Compute perturbation $\delta_j \ni$: $v_j^i + \delta_j$ gives $v_j^{\text{limit}}$ as its limit position,
                based on current $e_{j\,k}^i$ and $f_{j\,k}^i$
5.            Let $v_j^{i+1} = v_j^i + \frac{1}{2}\delta_j$
```

Step (4) is computed directly by solving the formula for the limit point given a vertex and its neighbors in an intermediate-level mesh, as established in Appendix 3. For a regular vertex of valence-n and its $2n$-neighborhood

$$[v, f_1, f_2, \ldots, f_n, e_1, e_2, \ldots, e_n]$$

the limit point of Catmull-Clark subdivision is computed as

$$v_{\text{limit}} = \frac{1}{n(n+5)}\left[n^2 v + \sum_{k=1}^{n}[4e_k + f_k]\right]. \tag{1}$$

Solving this for a perturbed $v + \delta$ to produce a desired v_{limit} gives

$$v + \delta = \frac{n+5}{n}\left[v_{\text{limit}} - \frac{1}{n(n+5)}\sum_{k=1}^{n}[4e_k + f_k]\right]. \tag{2}$$

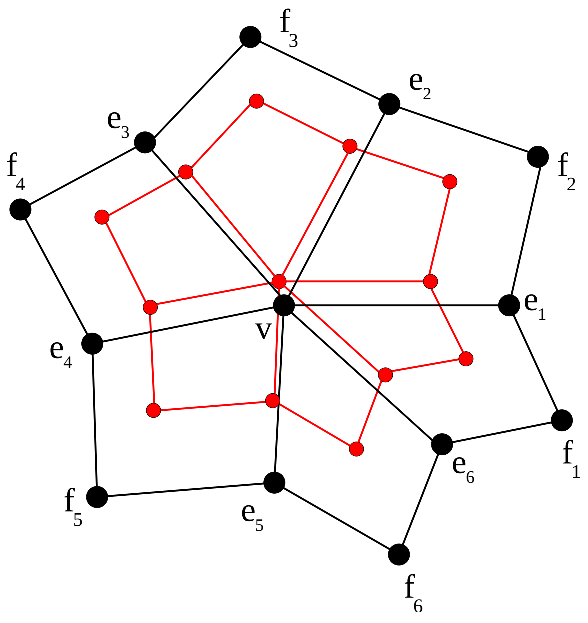


Figure 4. Vertex labeling in a regular 1-neighborhood. The e_i share an edge with the central vertex v and the f_i share a face.

(Fig. 4 shows the vertex labeling for a normal neighborhood.)

Thus, given a mesh of limit positions, a perturbed mesh could be produced by substituting the perturbed v^i values. One would expect that such a perturbed mesh would over-correct for the expected vertex shifts under Catmull-Clark subdivision, since it ignores changes to a vertex's neighbors which would effect the actual subdivision. Therein the reasoning behind step (5), where the perturbation is averaged between v_j^i and the perturbed $v_j^i + \delta_j$. The above equation thus yields the iteration

$$\begin{aligned} v_j^{i+1} &= v_j^i + \frac{1}{2}\delta_j^i \\ &= v_j^i + \frac{1}{2}\left[\frac{n+5}{n}\left[v_{\text{limit}} - \frac{1}{n(n+5)}\sum_{k=1}^{n}[4e_k + f_k]\right] - v_j^i\right] \end{aligned} \qquad (3)$$

for a vertex v_j of valence n. While this looks like the proverbial hairy spider, it actually amounts to a simple computation with a precomputed mask in the local 1-neighborhood of each vertex, thus requiring only local connectivity information on the initial involute mesh. Only one or two iterations produce a good approximation to the desired boundary; the iterative process, in fact, is computing a solution to the linear system defining the global problem by performing a successive overrelaxation (*SOR*) on a Jacobi iteration. To see that this is so, for the global subdivision, express the problem of solving

$$\begin{aligned}\vec{v}_{\text{limit}} &= \mathbf{A}\vec{v} \\ &= (\mathbf{D}+\mathbf{U}+\mathbf{L})\vec{v}\end{aligned}$$

and thus, elementwise,

$$\begin{aligned}v_{\text{limit}_j} &= A^j\vec{v} \\ &= (D^j+U^j+L^j)\vec{v} \\ &= D^j\vec{v}+(U^j+L^j)\vec{v} \\ &= \frac{n_j}{n_j+5}v_j+\frac{1}{n_j+5}\sum_{k=1}^{n}[4e_{kj}+f_{kj}]. \end{aligned} \qquad (4)$$

for $\vec{v}$, where **A** is the (sparse) subdivision limit-surface matrix for the entire mesh and $\vec{v}_{\text{limit}}$ is the vector of mesh vertices to be interpolated. As defined, each row j of the matrix **A** computes the weighted sums in equation 2.2 for the $v_{j_{\text{limit}}}$ element of $\vec{v}_{\text{limit}}$. From this $\mathbf{A} = (\mathbf{U}+\mathbf{L}+\mathbf{D})$ decomposes **A** into its upper and lower triangular and its diagonal components, where the D^j component is the vertex weight for the v_j vertex, and the $[U^j+V^j]\vec{v}$ gives the weighted sum of the edge and face neighbors of v_j. Inverting this to solve for $v_j+\delta_j$ gives

$$\begin{aligned}v_j+\delta_j &= \frac{n_j+5}{n_j}\left[v_{\text{limit}_j}-\frac{1}{n_j+5}\sum_{k=1}^{n}[4e_{kj}+f_{kj}]\right] \\ &= \frac{1}{D^j}\left[v_{\text{limit}_j}-(U^j+L^j)\vec{v}\right]\end{aligned}$$

and so, for the full linear system,

$$\vec{v}+\vec{\delta} = \mathbf{D}^{-1}\left[\vec{v}_{\text{limit}}-(\mathbf{U}+\mathbf{L})\vec{v}\right] \qquad (5)$$

where $\vec{\delta}$ is the vector of perturbations δ_j for each respective v_j.

Then, each iteration of equation 3 above can be combined in the matrix expression

$$\begin{aligned}
\vec{v}^{\,i+1} &= \vec{v}^{\,i} + \frac{1}{2}\vec{\delta}^{\,i} \\
&= \vec{v}^{\,i} + \frac{1}{2}\left[\mathbf{D}^{-1}\left[\vec{v}_{\text{limit}} - (\mathbf{U}+\mathbf{L})\vec{v}^{\,i}\right] - \vec{v}^{\,i}\right] \\
&= \vec{v}^{\,i} + \frac{1}{2}\mathbf{D}^{-1}\left[\vec{v}_{\text{limit}} - (\mathbf{U}+\mathbf{L})\vec{v}^{\,i} - \mathbf{D}\vec{v}^{\,i}\right] \\
&= \vec{v}^{\,i} - \frac{1}{2}\mathbf{D}^{-1}\left[(\mathbf{U}+\mathbf{L}+\mathbf{D})\vec{v}^{\,i} - \vec{v}_{\text{limit}}\right] \\
&= \vec{v}^{\,i} - \frac{1}{2}\mathbf{D}^{-1}\left[\mathbf{A}\vec{v}^{\,i} - \vec{v}_{\text{limit}}\right] \\
&= \vec{v}^{\,i} - \frac{1}{2}\mathbf{D}^{-1}\vec{\xi}^{\,i} \\
&= \vec{v}^{\,i} - \omega\mathbf{D}^{-1}\vec{\xi}^{\,i}
\end{aligned}$$

where $\vec{\xi}^{\,i}$ is the error residual vector for $\vec{v}^{\,i}$. This is the canonical form for an SOR on a Jacobi iteration, as discussed in Press et. al. [15], Strang [17], and Golub and Van Loan [8]. (Actually, it is an underrelaxation, since ω in the $\omega\delta_j$ term is less than 1.) For diagonally-dominant matrices—such as the (implied) global subdivision matrix for a closed, two-sided m-rep boundary—one expects good convergence for this method, and while $\omega = \frac{1}{2}$ was a pure guess, the iteration converges so rapidly in test cases that no fine-tuning was deemed necessary. (This matrix is not, however, diagonally dominant; a full proof of convergence is still pending. For a start, the spectral radius is 1, and 2nd and 3rd eigenvalues are 0.5.)

To further illustrate this method, consider the cases of regular vertices of valence 3 and 4. For an ordinary even vertex and its 8-neighborhood

$$[v, f_1, f_2, f_3, f_4, e_1, e_2, e_3, e_4]$$

the limit point of Catmull-Clark subdivision can be computed as

$$v_{\text{limit}} = \frac{4}{9}v + \frac{1}{9}\sum_k e_k + \frac{1}{36}\sum_k f_k \tag{6}$$

and solving for a perturbed $v+\delta$ producing a given v_{limit} gives

$$v + \delta = \frac{9}{4}\left[v_{\text{limit}} - \frac{1}{9}\sum_k e_k - \frac{1}{36}\sum_k f_k\right]. \tag{7}$$

Similarly, for a valence-3 vertex with its 6-neighborhood, one has

$$v_{\text{limit}} = \frac{3}{8}v + \frac{1}{6}\sum_k e_k + \frac{1}{24}\sum_k f_k \tag{8}$$

and thus

$$v + \delta = \frac{8}{3}\left[v_{\text{limit}} - \frac{4}{9}\sum_k e_k - \frac{1}{9}\sum_k f_k\right]. \tag{9}$$

The formalae in equations **??** and **??** yield the iterations

$$\begin{aligned} v_j^{i+1} &= v_j^i + \frac{1}{2}\delta_j^i \\ &= v_j^i - \frac{1}{2}\left[v_j^i + \frac{9}{4}\left[\frac{1}{9}\sum_k e_k^i + \frac{1}{36}\sum_k f_k^i - v_{j\,\text{limit}}\right]\right] \end{aligned} \tag{10}$$

and

$$\begin{aligned} v_j^{i+1} &= v_j^i + \frac{1}{2}\delta_j^i \\ &= v_j^i - \frac{1}{2}\left[v_j^i + \frac{8}{3}\left[\frac{1}{6}\sum_k e_k^i + \frac{1}{24}\sum_k f_k^i - v_{j\,\text{limit}}\right]\right] \end{aligned} \tag{11}$$

for a vertex v_j of valence 3 or 4 respectively.

Special issues are raised by the presence of non-quad faces in the initializing mesh. Given a vertex with adjacent faces with greater or fewer than 4 sides, one would like to avoid special cases in the formulae for inverting the limit point equations, which themselves would have to be based on an altered eigenstructure for the subdivision. Traditionally, limit positions at irregular vertices are computed by first doing a single Catmull-Clark splitting and averaging, which results in a regular mesh, and then applying the limit masks to the new vertices. Instead, as will be shown below, it is possible to *regularize* the neighborhood about a vertex to create a regular 1-neighborhood with the same limit-structure. Given such a regularized neighborhood, the standard limit-position and limit-tangent masks can be applied, and the above inversions and iterative interpolation method can be applied without modification. This technique will be discussed below.

2.3. IIS-SURFACES—IRREGULAR VERTEX LIMIT POSITIONS AND INVERSES

Recall that an *irregular* mesh is one with non-quadrangular polygons, and an irregular vertex is one at the corner of such a polygon. Because an initial Catmull-Clark mesh (such as one produced by blending a figure and subfigure) may contain irregular vertices, it is necessary to compute their limit points as well. This is frequently done in two stages:

1. a single Catmull-Clark split-and-average, after which all vertices are regular, followed by

2. the application of the limit mask for a regular vertex at the level-2 vertices.

This is inadequate for the needs of interpolating subdivision as developed above, where the limit equations must be inverted for use in the iterative, SOR interpolation scheme. Instead, going back to first principles for Catmull-Clark subdivision, working again from Halstead et al.[9], one can directly compute the limit point and tangent vectors for an irregular vertex (either ordinary or extraordinary). There is no standard matrix form for the subdivision at a vertex with an irregular one neighborhood. Instead, Catmull-Clark subdivision proceeds equivalently in the following way.

1. New face vertices $f_1^{i+1}, \ldots, f_n^{i+1}$ are created at the centroid (arithmetic mean) of the bounding polygon vertices for each bounding face $F_1, \ldots, F_n$.

2. New edge vertices are created

$$e_j^{i+1} = \frac{v^i + e_j^i + f_{j-1}^{i+1} + f_j^{i+1}}{4}. \tag{12}$$

3. The new vertex v^{i+1} can now be computed as

$$v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e_j^i + \frac{1}{n^2} \sum_j f_j^{i+1} \tag{13}$$

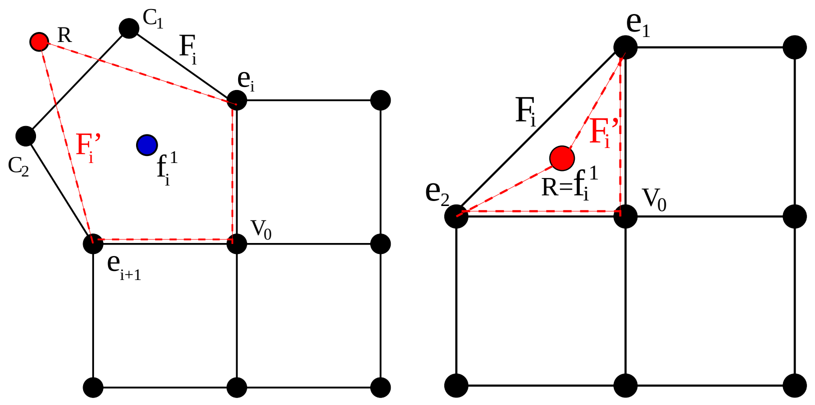
After the initial splitting, the new mesh is and remains regular. In regards vertex valences: extraordinary vertices will be created as new face-vertices of non-quadrilateral polygons, but all subsequent splittings leave vertex valences unchanged and create no new extraordinary vertices.

The thing to note from the above is that for a vertex v^0 having an irregular 1-neighborhood, the new 1-neighborhood $\{v^1, f_1^1, f_2^1, \ldots, e_1^1, e_2^1, \ldots\}$ depends only on $\{v^0, f_1^1, f_2^1, \ldots, e_1^0, e_2^0, \ldots\}$—i.e., the only difference an irregular mesh element makes is in the computation of the f_i^1 face vertices for the respective F_i polygons. Let $o(F_i)$ be the number of vertices in polygon F_i.

Theorem: Given F_i in a 1-neighborhood of v^0 and containing edges to vertices e_i^0 and e_{i+1}^0. If $o(F_i) \neq 4$, a new polygon F_i' can be constructed $\ni$: (a) $o(F_i') = 4$, (b) F_i' contains v^0 and the edge-vertices e_i^0 and e_{i+1}^0, and (c) F_i' has the same centroid as F_i.

The proof is fairly trivial.[2] If $o(F_i) = n > 4$, the vertices of F_i can be ordered as $\{v^0, e_i, c_1, c_2, \ldots, c_{n-3}, e_{i+1}\}$. Then it is necessary to find an

[2] Feynman's Observation: anything that can be proven is trivial.



$o(F_i) > 4$ $\quad$ $o(F_i) < 4$

Figure 5. Regularizing the neighborhood of an irregular vertex.

$F_i' = \{v^0, e_i, R, e_{i+1}\}$ having the same f_i^1 as its centroid, i.e.,

$$
\begin{aligned}
f_i^1 &= \frac{1}{n}\left[v^0 + e_i + e_{i+1} + \sum_{j=1}^{n-3} c_j\right] \\
&= \frac{1}{4}\left[v^0 + e_i + e_{i+1} + R\right] \\
\Rightarrow & \\
R &= \frac{4}{n}\left[\sum_{j=1}^{n-3} c_j\right] - \frac{n-4}{n}\left[v^0 + e_i + e_{i+1}\right]. \qquad (14)
\end{aligned}
$$

For the case $o(F_i) = 3$,

$$
\begin{aligned}
f_i^1 &= \frac{1}{3}\left[v^0 + e_i + e_{i+1}\right] \\
&= \frac{1}{4}\left[v^0 + e_i + e_{i+1} + R\right] \\
\Rightarrow & \\
R &= \frac{1}{3}\left[v^0 + e_i + e_{i+1}\right],
\end{aligned}
$$

and the new R is simply the centroid of the three vertices in F_i. (This is simply Eqn. 14 for case $n = 3$, with $\sum c_j$ nil, and not really a special case.) Fig. 5 gives an illustration of the process.

Corollary: An irregular 1-neighborhood of a vertex v^0 can be replaced by a regular neighborhood producing the same $\{v^1, f_1^1, f_2^1, \ldots, e_1^1, e_2^1, \ldots\}$ and therefore having the same limit structure (position and tangent space) in the neighborhood of v^∞.

This follows directly by applying the above method to all irregular polygons adjoining a vertex, given the known facts:

1. the level-$(n+1)$ 1-neighborhood for an ordinary or extraordinary vertex v^{n+1} is determined completely by v^n, by the level-n edge vertices e_i^n, and by the level-$(n+1)$ face vertices f_i^{n+1};

2. the limit structure can be derived explicitly by eigenanalysis, given a regular 1-neighborhood about an ordinary or extraordinary vertex.

This, then, allows limit positions to be determined directly from the initial mesh positions, in turn allowing the SOR iterative technique to be used to interpolate original mesh positions for irregular vertices just as for regular ones.

2.4. Error metrics for IIS-surfaces vs. medially-implied boundaries

IIS-surfaces do not interpolate mesh normals; thus, the $\vec{v}_1$, $\vec{v}_2$, and $\vec{b}$ vectors of the medial atoms defining the mesh of boundary involutes will not match the surface normals of the interpolating subdivision meshes. The degree of mismatch can, in fact, be arbitrarily bad. In most cases, the mesh topology for the closed surface at the coarsest scale produces surface normals *acceptably near* to the values implied by the medial atoms. Fully interpolating surface subdivision, as typified by Halstead, require extra conditions to prevent rippling effects on interpolating surfaces. Because the interpolation mesh for an m-rep is computed only for the coarsely sampled, medially-implied boundary, and because the normals are not specified explicitly, the smoothing inherent in stationary subdivision is sufficient to avoid such rippling in typical cases. A method of as-needed normal interpolation will be explored below; this is fast and simple, but lacks the fine control of methods which also include bending-energy minimization, and tends to exacerbate the problem of ripples, which must then be eliminated using small-scale boundary-texturing displacements.

To produce an as-needed normal-adjusting method, a method is first required to quantify the deviation of surface normals from medially-implied normals, and a non-stationary modification to the subdivision algorithm will be proposed to handle such cases. There is perhaps no reason not to use more general interpolating subdivision schemes, if one is willing to trade off the speed and simplicity of the IIS-surfaces for more control over boundary curvature. This may amount to spending a lot to achieve that *overprecision* that m-reps are intended to avoid.

There needs to be a measure for the goodness of fit of the iterative subdivision boundary to the surface implied by the medial atoms. While the boundary should only need to be specified within tolerances, it is important to be able to state that this is in fact the case. Taking as *truth* the known boundary locations and their normals as implied by the involute vectors of the sampled atoms, there are several relevant metrics:

1. measurement of the distance from the original involute endpoints of each of their initializing positions in the coarse-level mesh, scaled by r;
2. measurement of the rotation θ from an original involute normal to the subdivision boundary normal;
3. measurement of the shift from a known medial location $\mathbf{p}$ to a new one based on creating a new medial atom from associated boundary locations and normals.

Metric (1) can be used to determine how many iterations are needed to approximate the medially-implied boundary. Metric (2) can be used to determine if and where a non-stationary normal-interpolation method should be applied to the subdivision mesh. Metric (3) is based on the involute-to-average-medial-atom technique presented in the last chapter. (Recall Fig. **??**.) The distance of displacement of the old medial location $\mathbf{p}$ from the new (in r-proportional terms) gives a measure of how poorly the coarse medial sampling fits the interpolating surface in a Blum-sense. An alternative to adjusting the normals of the subdivision mesh (as described below) might be to modify instead the medial atom, using the approximately Blum atom generated for the test.

2.5. ACCURATE NORMAL INTERPOLATION USING MODIFIED IIS-SURFACES

In many cases, the basic topology and structure of the coarse involute mesh guides the subdivision to approximately correct normal values. Nonetheless, without a boundary fit that incorporates the surface normals implied by the medial atoms, the m-rep medial atom to boundary-implied medial atom error can be made arbitrarily bad. Using the above metric on normal deviation, it is simple to identify if and where this occurs. For such situation, I have created a quick and not-too-dirty method to interpolate the normal at a given location, using a non-stationary, non-uniform modification to the interpolating subdivision algorithm. **Given**: an initial mesh of medially-implied boundary involutes v_n with implied normals $\vec{n}_n$.

```
1. Construct a level-0 IIS-surface mesh of vertices $v_n^0$ that give $v_n^{\text{limit}} \approx v_n$
      under Catmull-Clark subdivision.
2. Subdivide twice to get vertices $v_n^2$ and their level-2 1-neighborhoods.
3. Compute $v_n^{\text{limit}}$ and $\vec{n}_n^{\text{limit}}$.
4. For $i = 1$ to $n$ do
5.          If $\vec{n}_i^{\text{limit}}$ is acceptable close to $\vec{n}_i$, **Done**.
6.          Else, compute the rotation $\mathcal{R}_i \in SO(3)$ taking the vector $\vec{n}_i^{\text{limit}}$ to $\vec{n}_i$.
7.          Rotate $v_i^2$ and its entire 1-neighborhood by $\mathcal{R}_i$ about $v_i^{\text{limit}}$.
8.          Substitute these rotated vertices into the $v_n^2$ mesh.
```

The resulting modified v_n^2 mesh will interpolate the normals for the limit positions of the modified vertices. The proof of this is trivial, resting on three facts:

– local 1-neighborhoods of initial vertices are disjoint after 2 subdivisions;

– a vertex's limit-surface position and normal depend only on the 1-neighborhood of that vertex;

– subdivision is rotationally invariant, allowing the rotation of the limit normal to be achieved by rotating the subdivision neighborhood by the same amount about the limit point while keeping that limit point unchanged.

It is a strength of this method that it is applied on an *as needed* basis, and that it requires only an $\mathcal{O}(1)$ operation for those of the n initial vertices requiring modification, adding at worst $\mathcal{O}(n)$ operations overall (for n typically small). Further, the interpolation can be applied *in-place* on a subdivision mesh, modifying only the independent 1-neighborhoods about the selected mesh positions.

Fig. 6 shows an m-rep boundary generated by this algorithm; the polygonal outlines show where the underlying subdivision mesh was perturbed to match normals in the neighborhoods of the sampled medial involutes. Fig. 7 compares a bone (in a pelvic-region model) with boundary generated by both normal-interpolating and non-normal-interpolating IIS-surfaces. While normal interpolation produces a more detailed boundary, the detail seen could alternatively be provided by boundary displacements on the less detailed surface. The advantage of the normal interpolating method is that the sampled medial atoms now form an accurate Blum-medial location for the generated boundary and thus provide the correct ∇r information for a sampled medial representation. The interpolation provided by the non-normal-interpolating surface is effectively for a chordal axis, since the 1st-order information provided by the medial atom is ignored.

This method is subject to the same potential difficulties with surface ripples that afflict other subdivision techniques which interpolate normals. For m-rep modeling, however, the coarseness of the medial sampling (which defines the initial subdivision mesh) alleviates much of this problem. For non-m-rep modeling, it might be useful to space the rotation out over several subdivision steps to give a smoother interpolation—this is similar to a technique developed by Biermann[2]. It is also possible that ripples can be reduced by doing a *minimal* perturbation so that the boundary normal is just within a specified tolerance without being exactly interpolating. Can also do a smoothing on the mesh after the perturbation of v^2 and its neighborhood. Since there is a ring of unperturbed vertices around each perturbed neighborhood, these

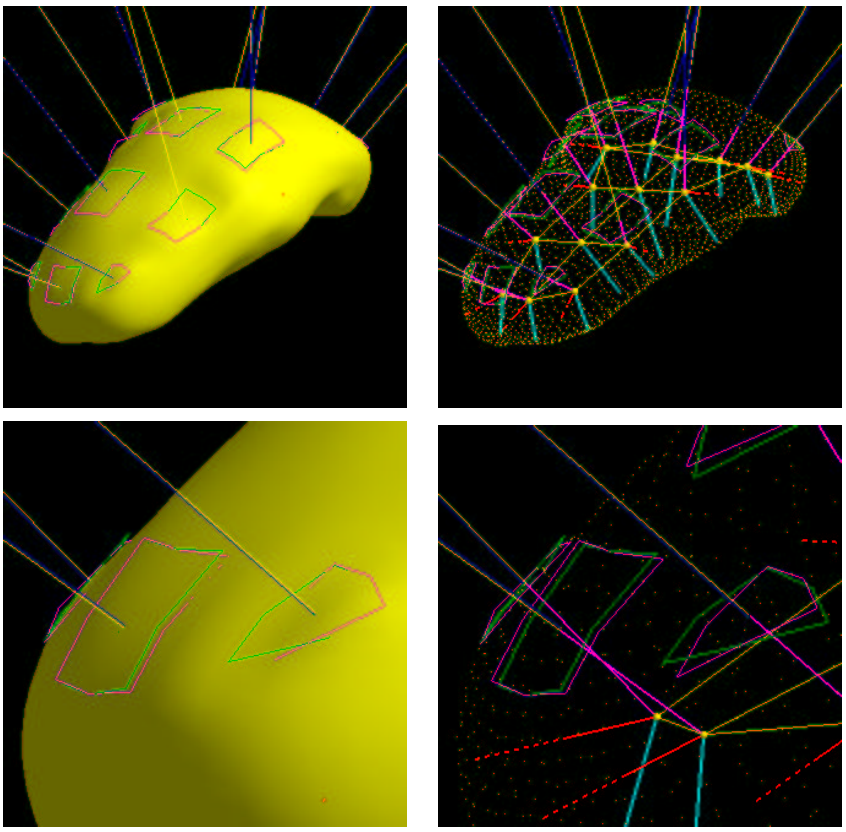


Figure 6. An m-rep boundary generated by normal-interpolating IIS-surfaces. The polygonal regions show the perturbation of 1-neighborhoods in the underlying subdivision mesh to match boundary normals at the sampled medial involutes.

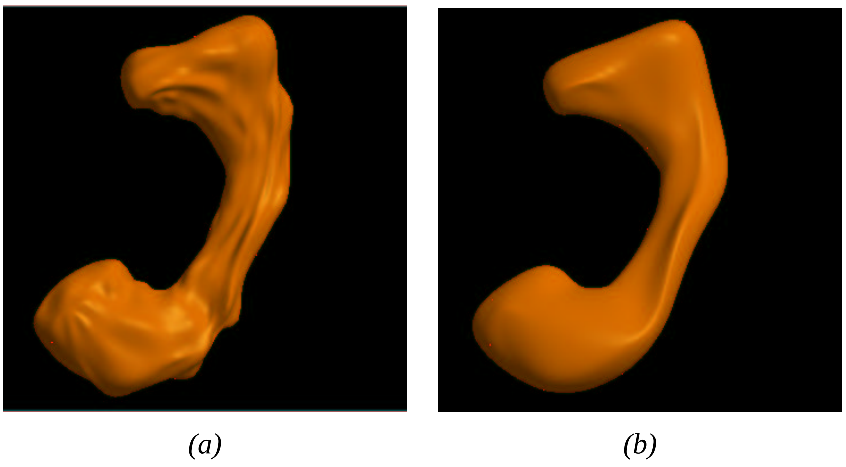


Figure 7. A bone modeled *(a)* by normal-interpolating and *(b)* by non-normal interpolating IIS-surfaces.

can be averaged by their neighbors to reduce possible rippling. Such non-stationary methods can get arbitrarily *ad hoc* and may become pointlessly rococo. As alternatives, there remain the general interpolating subdivision schemes such as Halstead's or Biermann's that give more precise surface control. For m-reps, however, the above scheme is effective and computationally cheap.

2.6. Interpolation of other boundary attributes

The usefulness of a medially-defined coordinate system was discussed in the previous chapter in Sec. **??**. It is necessary, therefore, to interpolate medial coordinates from known involute positions on the boundaries, and to interpolate approximate values of r as well. The standard practice for texture coordinate interpolation on subdivision surfaces is to split the texture coordinates at the same time the mesh itself is split. Typically, texture coordinates will be subdivided using the same splitting and perturbation scheme as for the vertices (DeRose[3]); this gives 2nd order continuity to the surface coordinates. However, for an interpolating surface such as an m-rep, this results in "coordinate-creep", with values shifting from their known values at interpolated positions.

Instead, a simple midpoint subdivision rule has been used instead for r and (u, v, t) coordinates, leaving values unchanged at even vertices. The values at odd edge-vertices are the average of the even endpoints, and the values at odd face-vertices are the average of the even face vertices. This does have the drawback of giving only 0-th order continuity of the coordinate fields on the surface; this has not proven to be a sticking point for any of the current research.

Another possibility is to use the same IIS-surface scheme on the surface coordinates; this would still produce coordinate-creep near known locations subject to the number of iterations of the approximation. If continuous coordinate fields are needed, though, this is the most straightforward way to achieve them.

It is important to note that the interpolated r-field on the surface is only an approximation to the true medial radius function on the surface. It is, however, well-defined, and it provides a good approximation for determining width-proportional tolerances at boundary locations. This will be discussed in the next chapter in Sec. **??**. Also note that, while subdivision-based coordinate interpolation allows coordinates to be assigned to all mesh vertices at any level, it does not solve the problem of finding an (x, y, z) location in space corresponding to an arbitrary (u, v, t). Methods for this will be discussed in detail in Sec. **??**, but basically involve linearly interpolating vertex parameter values across tiled boundaries at a subdivision level deemed suitably fine.

2.7. Drawbacks and limitations of IIS-surfaces

There are a number of drawbacks to the use of such subdivision surfaces for m-rep boundaries. They lack explicit parameterizations, and thus,likewise, closed-form surface curvatures, principal directions, and fundamental forms, which would be useful for doing differential geometry on them. One can fit splines to regular regions or do Stam-style parameterization[16]; one might use methods of Jörg Peters and Georg Umlauf for finding Gaussian and mean curvature of subdivision surfaces[13, 14], or tricks on the surface mesh vertices as per Desbrun et al. at Caltech Multi-Res Modeling Group[4]. It remains a fact that analysis on subdivision boundaries is non-trivial.

These are drawbacks for *all* subdivision surfaces. There is a drawback particular to iteratively interpolating subdivision surfaces. A standard trick for putting edges and creases in Catmull-Clark or other subdivision surfaces is to "freeze" certain vertices and edges in the mesh, allowing averaging only along a restricted set of mesh edged or not at all. This method cannot be applied directly to the modified meshes produced by IIS-surfaces; it might be possible to freeze vertices and edges in the initial mesh and not do inverse interation there, but I suspect that this will produce undesirable artifacts. Boundary displacement, as discussed in the next chapter, makes vertex/edge freezing less necessary; whether it eliminates the need entirely remains to be seen.

Another disadvantage of IIS-surfaces—as opposed to a global least-squares approach to subdivision surface interpolation such as that of Halstead—is that the method requires that the (implicit) subdivision matrix be non-singular. While this might limit IIS-surface use in general modeling tasks, m-rep-generated boundary meshes always have closed, two-sided topology and the subdivision equations are therefore invertible. The simplicity and adjustable tolerance of IIS-surface fitting makes it preferable in this and similar cases.

Another drawback common to all surface-fitting by stationary subdivision is that the surface-fit is essentially a bicubic-spline interpolation, restricting the boundary curvature behavior accordingly. Thus, as Fig. 8 illustrates, a surface fit to an endcap (the mostly highly curved region, typically) may have undesirable wiggles. The use of the edge-atom η-elongation can resolve this, or a boundary perturbation might do likewise; as a last recourse are the subdivision-surface-fitting algorithms with boundary curvature constraints, such as Halstead's.

A final note on IIS-surfaces: a careful reading of Farin [6](Pg. 125) reveals a discussion of just such an iterative approach for b-spline interpolation, though without the above modifications for normal interpolation. Since a Catmull-Clark surface is almost everywhere a bicubic b-spline, it is natural that such an approach should be applicable to the subdivision surfaces as well.

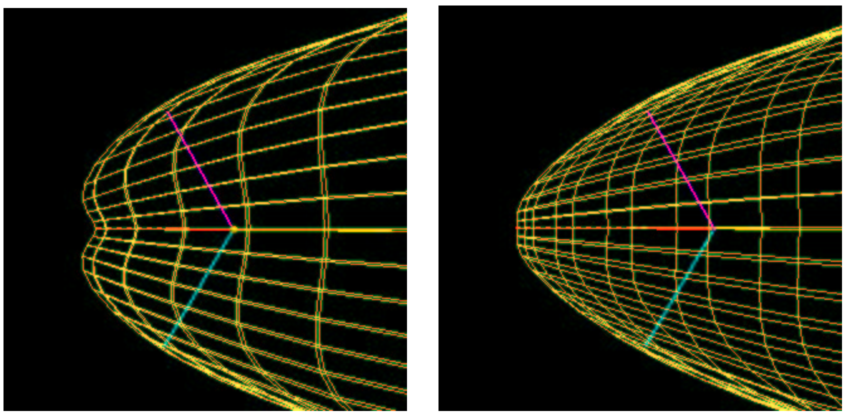


Figure 8. An endcap interpolated using interpolating IIS-surfaces. On the left, with $\eta = 1$, the surface has typical cubic-spline wiggling. On the right, η has been adjusted and the medial atom moved to eliminate the effect.

Acknowledgements

Thanks to Steve Pizer for his proposal of an iteratively refined approximating approach, and to Sarang Joshi for his advice on the applicability of successive overrelaxation methods to the solution of linear systems. This work received support from National Cancer Institute grant PO1-CA47982, and by NSF SGER grant CCR-9910419 for 1999-2000. I also received support from a Link Foundation fellowship for 1998-1999.

3. Appendix: Limit Positions on Subdivision Surfaces

Limit positions and tangent vectors (and thus normals) can be computed for initial even vertices of a subdivision mesh by weighted averages of their local n-neighborhood with appropriately computed masks. For stationary subdivision methods (such as Loop and Catmull-Clark), these masks can be computed by eigenanalysis of the single splitting matrix for vertices of the given valence. A good discussion of this for Catmull-Clark subdivision can be found in Halstead et al.[9] The computation in this chapter is for the specific cases related to the subdivision code in Rakshasa and in Pablo/Seurat. The extended discussion and numerical tests in Sec. 3.1 and 4 were done for my own edification to ensure correct understanding of the mathematics.

3.1. Limit positions for regular, valence-4 vertices

For this discuss, a *regular* vertex is one which is has only quad neighbors, rather than the arbitrary polygonal neighbors allowed at the first stage of

Catmull-Clark subdivision.[3] (Computing limit positions of irregular vertices is more complicated but not different in essence. One first regularizes the local neighborhood; this is discussed in Thall[2003].) Consider a regular, level-i vertex v_0^i and its 8-neighborhood as a 9-vector

$$\vec{V}^i = [v_0^i, f_1^i, f_2^i, f_3^i, f_4^i, e_1^i, e_2^i, e_3^i, e_4^i]^T$$

along with the associated subdivided vertex v_0^{i+1} and its neighbors

$$\vec{V}^{i+1} = [v_0^{i+1}, f_1^{i+1}, f_2^{i+1}, f_3^{i+1}, f_4^{i+1}, e_1^{i+1}, e_2^{i+1}, e_3^{i+1}, e_4^{i+1}]^T$$

arranged as in Figure 9. A single Catmull-Clark splitting operation gives the relationship $\vec{V}^{i+1} = \mathbf{S}\vec{V}^i$, where the splitting matrix

$$\mathbf{S} = \frac{1}{16}\begin{pmatrix} 9 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} \\ 4 & 4 & 0 & 0 & 0 & 4 & 0 & 0 & 4 \\ 4 & 0 & 4 & 0 & 0 & 4 & 4 & 0 & 0 \\ 4 & 0 & 0 & 4 & 0 & 0 & 4 & 4 & 0 \\ 4 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 \\ 6 & 1 & 1 & 0 & 0 & 6 & 1 & 0 & 1 \\ 6 & 0 & 1 & 1 & 0 & 1 & 6 & 1 & 0 \\ 6 & 0 & 0 & 1 & 1 & 0 & 1 & 6 & 1 \\ 6 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 6 \end{pmatrix}$$

is based on the masks for even and odd vertices with standard weights. (See Fig. 10.)

The masks for computing the limit position V_0^∞ and vectors in the tangent plane of the limit surface at this point can be found by computing the left-eigenvectors and eigenvalues of the splitting matrix. (For a proof of this, see Halstead[9]. From the theory, $\lambda_1 = 1$ and its corresponding eigenvector is

$$l_1 = \frac{1}{n(n+5)}[n^2, 1, \ldots, 1, 4, \ldots, 4]^T.$$

Also from the theory, eigenvalues $\lambda_2 = \lambda_3 = \frac{4+A_n}{16}$, where A_n is defined below in Eq. 15, for vertices of valence n.

[3] Don't confuse this with *ordinary* vs. *extraordinary* vertex, which refers only to the valence, the number of connected mesh-neighbors.

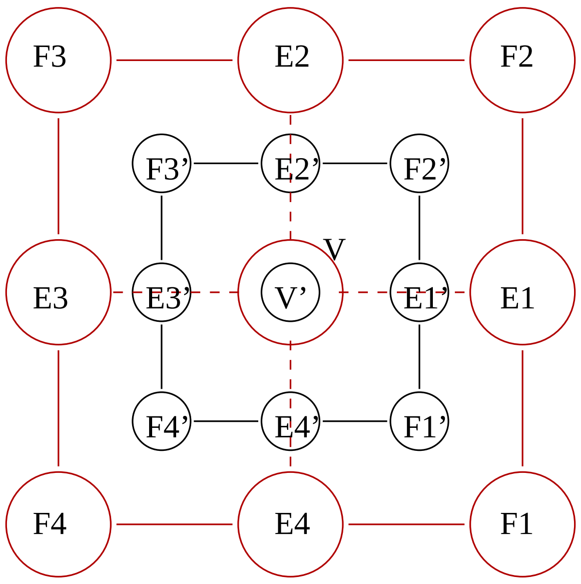


Figure 9. Vertex and its 8-neighborhood (in red) and related subdivided vertices (in black).

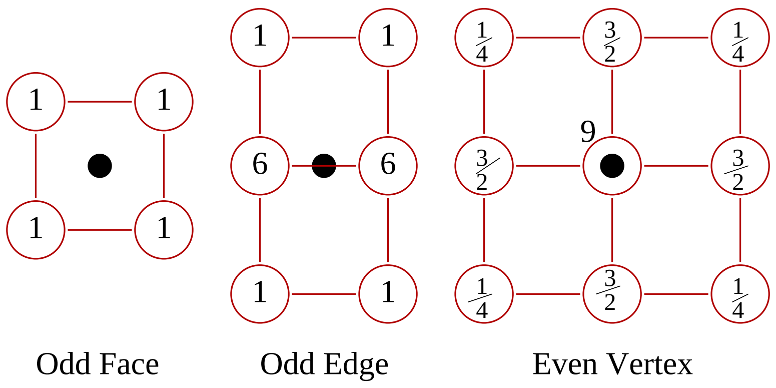


Figure 10. Subdivision masks for odd and even vertices (of valence 4).

Using MATLAB, numerical eigenanalysis produced eigenvalues $\lambda_1 = 1.0$, and $\lambda_{\{2,3\}} = 0.5$ as expected. For left eigenvectors, I found

$$l_1 = \frac{1}{36}\begin{pmatrix}16\\1\\1\\1\\1\\4\\4\\4\\4\end{pmatrix}, \; l_2 = \begin{pmatrix}0.0\\-0.02063816984618\\-0.47936183015382\\0.02063816984618\\0.47936183015382\\-1.0\\-0.91744732061529\\1.0\\0.91744732061529\end{pmatrix}, \; l_3 = \begin{pmatrix}0.0\\0.47385759580580\\-0.02614240419420\\-0.47385759580580\\0.02614240419420\\0.89543038322319\\-1.0\\-0.89543038322319\\1.0\end{pmatrix}.$$

The l_1 eigenvector is in accord with the limit weights for the limit vertex as given in Halstead et al. Because $\lambda_2 = \lambda_3$, the l_2 and l_3 vectors are non-unique; using Fourier methods, Halstead's folk analytically derived formulae for limit tangent vectors as

$$\begin{aligned} c_2 &= \sum_j A_n \cos(\frac{2\pi j}{n}) e^1_j + (\cos(\frac{2\pi j}{n}) + \cos(\frac{2\pi(j+1)}{n})) f^1_j \\ c_3 &= \sum_j A_n \cos(\frac{2\pi j}{n}) e^1_{j+1} + (\cos(\frac{2\pi j}{n}) + \cos(\frac{2\pi(j+1)}{n})) f^1_{j+1} \end{aligned}$$

where

$$A_n = 1 + \cos(\frac{2\pi}{n}) + \cos(\frac{\pi}{n}) \sqrt{2(9 + \cos(\frac{2\pi}{n}))} \tag{15}$$

where n is the valence, and j is the edge/face index modulo $n+1$. From these, for valence-4 vertices one can compute neighborhood masks of

$$m_2 = \begin{pmatrix} 0 \\ -1 \\ -1 \\ 1 \\ 1 \\ -4 \\ 0 \\ 4 \\ 0 \end{pmatrix}, \quad m_3 = \begin{pmatrix} 0 \\ 1 \\ -1 \\ -1 \\ 1 \\ 0 \\ -4 \\ 0 \\ 4 \end{pmatrix}.$$

These lie in the span of the l_2 and l_3 vectors derived from the numerical eigenanalysis of **S** and are sweet and gracious, entirely more pleasant for computational purposes. (Another source, Havemann[10], gives masks $m_2 = [0, 1, 1, -1, -1, 1, 0, -1, 0]$ and $m_3 = [0, 0, 2, 0, -2, 1, 1, -1, -1]$. These appear to be incorrect in not lying in the subspace spanned by the eigenvectors corresponding to the second and third eigenvalues. It is possible that they are using different weights in their splitting matrix.)

4. Limit positions for regular, valence-3 vertices

Similarly to above, consider a level-i vertex v^i_0 and its 6-neighborhood as a 7-vector

$$\vec{V}^i = [v^i_0, f^i_1, f^i_2, f^i_3, e^i_1, e^i_2, e^i_3]^T$$

along with the associated subdivided vertex v^{i+1}_0 and its neighbors

$$\vec{V}^{i+1} = [v^{i+1}_0, f^{i+1}_1, f^{i+1}_2, f^{i+1}_3, e^{i+1}_1, e^{i+1}_2, e^{i+1}_3]^T$$

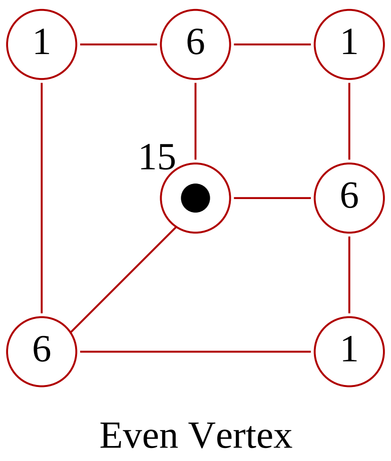


Figure 11. Subdivision mask for even vertex of valence 3.

arranged as in Figure 9. A single Catmull-Clark splitting operation gives the relationship $\mathbf{S}\vec{V}^{i} = \vec{V}^{i+1}$, where the splitting matrix

$$\mathbf{S} = \frac{1}{12}\begin{pmatrix} 5 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 2 & 2 & 2 \\ 3 & 3 & 0 & 0 & 3 & 0 & 3 \\ 3 & 0 & 3 & 0 & 3 & 3 & 0 \\ 3 & 0 & 0 & 3 & 0 & 3 & 3 \\ \frac{9}{2} & \frac{3}{4} & \frac{3}{4} & 0 & \frac{9}{2} & \frac{3}{4} & \frac{3}{4} \\ \frac{9}{2} & 0 & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} & \frac{9}{2} & \frac{3}{4} \\ \frac{9}{2} & \frac{3}{4} & 0 & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} & \frac{9}{2} \end{pmatrix}$$

is based on the masks for even mask as shown in Fig. 11, with the same odd vertex weights as in Fig. 10.

The masks for computing the limit position and tangent vectors on the limit surface are found once again by computing the left-eigenvectors and eigenvalues of the splitting matrix.

Using MATLAB, numerical eigenanalysis produced eigenvalues $\lambda_1 = 1.0$, and $\lambda_{\{2,3\}} = 0.4109705080055$, which indeed matches the analytically predicted $\lambda_{\{2,3\}} = \frac{9+\sqrt{17}}{32}$. For left eigenvectors, I found

$$l_1 = \frac{1}{24}\begin{pmatrix} 9 \\ 1 \\ 1 \\ 1 \\ 4 \\ 4 \\ 4 \end{pmatrix}, \; l_2 = \begin{pmatrix} 0.0 \\ -0.29639444898437 \\ 0.16358333275582 \\ 0.13281111622855 \\ -0.34020268834752 \\ 0.75923003449683 \\ -0.41902734614931 \end{pmatrix}, \; l_3 = \begin{pmatrix} 0.0 \\ -0.10092784247726 \\ -0.19137106863820 \\ 0.29229891111546 \\ -0.74873909794875 \\ 0.25853199878834 \\ 0.49020709916041 \end{pmatrix}.$$

The l_1 eigenvector once again is in accord with the weights for the limit vertex as given in Halstead and the l_2 and l_3 again span a subspace of acceptable vectors including those computed analytically; by the same formulae evaluated for n=3,

$$\begin{aligned} c_2 &= \sum_j A_3 \cos(\frac{2\pi j}{3}) e^1_j + (\cos(\frac{2\pi j}{3}) + \cos(\frac{2\pi(j+1)}{3})) f^1_j \\ c_3 &= \sum_j A_3 \cos(\frac{2\pi j}{3}) e^1_{j+1} + (\cos(\frac{2\pi j}{3}) + \cos(\frac{2\pi(j+1)}{3})) f^1_{j+1} \end{aligned}$$

where

$$A_3 = 1 + \cos(\frac{2\pi}{3}) + \cos(\frac{\pi}{3})\sqrt{2(9+\cos(\frac{2\pi}{3}))}$$

From these, for valence-3 vertices one can extract neighborhood masks of

$$m_2 = \begin{pmatrix} 0 \\ -4 \\ 2 \\ 2 \\ -1-\sqrt{17} \\ 2+2\sqrt{17} \\ -1-\sqrt{17} \end{pmatrix}, \quad m_3 = \begin{pmatrix} 0 \\ 2 \\ 2 \\ -4 \\ 2+2\sqrt{17} \\ -1-\sqrt{17} \\ -1-\sqrt{17} \end{pmatrix}.$$

These, once again, are computationally cleaner than the l_2 and l_3 masks derived from the numerical eigenanalysis. (Note that $m_2 \times m_3$ gives the *inward* pointing normal, so $m_3 \times m_2$ is the correct surface normal.)

4.1. Limit masks for regular vertices—general cases

Since the limit position masks are necessary (to be inverted) for interpolating subdivision as they will be computed here and tabulated for valences $n \leq 12$ using Halstead's formula. As given in Sec. 3.1, the formula for the limit point of a regular vertex of valence n is

$$l_1 = \frac{1}{n(n+5)}[n^2, 1, \ldots, 1, 4, \ldots, 4]^T.$$

and the formulae for the tangents are

$$\begin{aligned} c_2 &= \sum_j A_n \cos(\frac{2\pi j}{n}) e^1_j + (\cos(\frac{2\pi j}{n}) + \cos(\frac{2\pi(j+1)}{n})) f^1_j \\ c_3 &= \sum_j A_n \cos(\frac{2\pi j}{n}) e^1_{j+1} + (\cos(\frac{2\pi j}{n}) + \cos(\frac{2\pi(j+1)}{n})) f^1_{j+1} \end{aligned}$$

Table I. Limit vector masks for regular extraordinary points.

Valence	*Limit vector mask*
3	$\frac{1}{24}[9,1,1,1,4,4,4]^T$
4	$\frac{1}{36}[16,1,1,1,1,4,4,4,4]^T$
5	$\frac{1}{50}[25,1,1,1,1,\ldots,4,4,4]^T$
6	$\frac{1}{66}[36,1,1,1,1,\ldots,4,4,4]^T$
7	$\frac{1}{84}[49,1,1,1,1,\ldots,4,4,4]^T$
8	$\frac{1}{104}[64,1,1,1,1,\ldots,4,4,4]^T$
9	$\frac{1}{126}[81,1,1,1,1,\ldots,4,4,4]^T$
10	$\frac{1}{150}[100,1,1,1,1,\ldots,4,4,4]^T$
11	$\frac{1}{176}[121,1,1,1,1,\ldots,4,4,4]^T$
12	$\frac{1}{204}[144,1,1,1,1,\ldots,4,4,4]^T$

Table II. Valence-3 tangent mask weights.

	c_2 *limit vector mask*	c_3 *limit vector mask*
f_1	-4	2
f_2	2	2
f_3	2	-4
e_1	$-1-\sqrt{17}$	$2+2\sqrt{17}$
e_2	$2+2\sqrt{17}$	$-1-\sqrt{17}$
e_3	$-1-\sqrt{17}$	$-1-\sqrt{17}$

where

$$A_n = 1 + \cos(\frac{2\pi}{n}) + \cos(\frac{\pi}{n})\sqrt{2(9+\cos(\frac{2\pi}{n}))} \tag{16}$$

where j is the edge/face index modulo $n+1$. Using these equations, the limit point masks for regular points with valences $n \leq 12$ are tabulated in Table 4.1. The tangent masks can be computed as needed by the above formulae, as in the $n = 3$ and $n = 4$ cases in Tables 4.1 and 4.1. Note that these are in $\{f_1, e_1, f_2, e_2, \ldots\}$ counterclockwise order, as per the standard in Seurat, rather than ordered $\{e_1, f_1, e_2, f_2, \ldots\}$ as in Halstead; e.g., for the valence 3 vertex in Table 4.1, the e_1 in the table is actually the e_2 value from the formulae, e_2 is actually e_3 from the formula, and e_3 is e_1, and similar for the other valances computed by the formulae. The f_i values are as per the formula values.

Table III. Valence-4 tangent mask weights.

	c_2 *limit vector mask*	c_3 *limit vector mask*
f_1	-1	1
f_2	-1	-1
f_3	1	-1
f_4	1	1
e_1	-4	0
e_2	0	-4
e_3	4	0
e_4	0	4

5. Inverse limit points for extraordinary vertices

The equations for inverse limit positions for valence-3 and valence-4 vertices were derived in Thall[2003]. Reiterating here: the interpolating surface is produced by creating a Catmull-Clark surface based on a modified initializing vertex grid; this modified grid is created by the following algorithm:

```
1. Initialize a boundary mesh $v_n^0$ with vertices $v_n^{\text{limit}}$ of positions to be interpolated
2. For iteration $i = 1$ to $m$
3.      For each vertex $v_j^i$ and its immediate edge $e_{j_k}^i$ and face $f_{j_k}^i$ neighbors
             [$k$ indexing into neighborhood of $v_j^i$]
4.      Compute perturbation $\delta_j \ni$: $v_j^i + \delta_j$ gives $v_j^{\text{limit}}$ as its limit position,
             based on current $e_{j_k}^i$ and $f_{j_k}^i$
5.      Let $v_j^{i+1} = v_j^i + \frac{1}{2}\delta_j$
```

Step [4] is computed directly by inverting the limit formulae as given above in Table 4.1. This section will derive the general form for these inverses and give the iterative equations for Step [5] for the valences 3–12.

For a regular vertex of valence-n and its $2n$-neighborhood

$$[v, f_1, f_2, \ldots, f_n, e_1, e_2, \ldots, e_n]$$

the limit point of Catmull-Clark subdivision is computed as

$$v_{\text{limit}} = \frac{1}{n(n+5)}\left[n^2 v + \sum_{k=1}^{n}[4e_k + f_k]\right] \tag{17}$$

and solving for a perturbed $v + \delta$ producing a given v_{limit} gives

$$v + \delta = \frac{n+5}{n}\left[v_{\text{limit}} - \frac{1}{n(n+5)}\sum_{k=1}^{n}[4e_k + f_k]\right]. \tag{18}$$

Thus, given a mesh of limit positions, a perturbed mesh could be produced by substituting these v^i values. One would expect that such a perturbed mesh would over-correct for the expected vertex shifts under Catmull-Clark subdivision, since it ignores changes to a vertex's neighbors which would effect the actual subdivision. Therein the reasoning behind step [5], where the perturbation is averaged between v_j^i and the perturbed $v_j^i + \delta_j$. The above equation thus yields the iteration

$$\begin{aligned} v_j^{i+1} &= v_j^i + \frac{1}{2}\delta_j^i \\ &= v_j^i + \frac{1}{2}\left[\frac{n+5}{n}\left[v_{\text{limit}} - \frac{1}{n(n+5)}\sum_{k=1}^{n}[4e_k + f_k]\right] - v_j^i\right] \end{aligned} \tag{19}$$

for a vertex v_j of valence n. Only one or two iterations often produce a good approximation to the desired boundary; the iterative process, in fact, is computing a solution to the linear system defining the global problem by performing a successive overrelaxation (*SOR*) on a Jacobi iteration. (Actually, it is an underrelaxation, since ω in the $\omega\delta_j$ term is less than 1.) To see that this is so: for the global subdivision, express the problem as solving

$$\mathbf{A}\vec{v} = \vec{v}_{\text{limit}}$$

for $\vec{v}$, where **A** is the (sparse) subdivision limit-surface matrix for the entire mesh and $\vec{v}_{\text{limit}}$ is the vector of mesh vertices to be interpolated. As defined, each row j of the matrix **A** computes the weighted sums in the above equations for the $v_{j\,\text{limit}}$ element of $\vec{v}_{\text{limit}}$. Let $\mathbf{A} = (\mathbf{U}+\mathbf{L}+\mathbf{D})$ be a decomposition of **A** into its upper and lower triangular and its diagonal components.

Then, each iteration of the iteration equations above can be combined in the matrix expression

$$\begin{aligned} \vec{v}^{\,i+1} &= \vec{v}^{\,i} + \frac{1}{2}\vec{\delta}^{\,i} \\ &= \vec{v}^{\,i} + \frac{1}{2}\left[\mathbf{D}^{-1}\left[\vec{v}_{\text{limit}} - (\mathbf{U}+\mathbf{L})\vec{v}^{\,i}\right] - \vec{v}^{\,i}\right] \\ &= \vec{v}^{\,i} + \frac{1}{2}\mathbf{D}^{-1}\left[\vec{v}_{\text{limit}} - (\mathbf{U}+\mathbf{L})\vec{v}^{\,i} - \mathbf{D}\vec{v}^{\,i}\right] \\ &= \vec{v}^{\,i} - \frac{1}{2}\mathbf{D}^{-1}\left[(\mathbf{U}+\mathbf{L}+\mathbf{D})\vec{v}^{\,i} - \vec{v}_{\text{limit}}\right] \\ &= \vec{v}^{\,i} - \frac{1}{2}\mathbf{D}^{-1}\vec{\xi}^{\,i} \\ &= \vec{v}^{\,i} - \omega\mathbf{D}^{-1}\vec{\xi}^{\,i} \end{aligned}$$

where $\vec{\xi}^{\,i}$ is the error residual vector for $\vec{v}^{\,i}$. This is the canonical form for an underrelaxation on a Jacobi iteration, as discussed in Press et. al. [15], Strang [17], and Golub and Van Loan [8]. For diagonally-dominant matrices—such as our (implied) global subdivision matrix—one expects good convergence for this method, and while $\omega = \frac{1}{2}$ was a pure guess, the iteration converges so rapidly in test cases that no fine-tuning was deemed necessary.

5.1. An implementation aid for Catmull-Clark meshes

From Halstead et al.[9], the algorithm for Catmull-Clark subdivision proceeds in the following way.

1. New face vertices $f_1^{i+1}, \ldots, f_n^{i+1}$ are created at the centroid (arithmetic mean) of the bounding polygon vertices for each bounding face. $F_1, \ldots, F_n$.

2. New edge vertices are created

$$e_j^{i+1} = \frac{v^i + e_j^i + f_{j-1}^{i+1} + f_j^{i+1}}{4}. \tag{20}$$

3. The new vertex v^{i+1} can now be computed as

$$v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e_j^i + \frac{1}{n^2} \sum_j f_j^{i+1} \tag{21}$$

One way to perform such a splitting *in-place* is create the new mesh in the following manner.

1. Create the new mesh of nodes, labeling each as one of {even, odd-face, odd-edge}, and computing local 1-neighborhoods for each vertex.

2. For each even node v^{i+1}, copy the parent v^i node information.

3. Compute each odd face f^{i+1} as the centroid of its face-neighbors, which are simply the 4 (or valence-n) v^i values as copied over from the parent mesh.

4. Compute each odd-edge e^{i+1} as the centroid of its edge-neighbors, which are simply the 2 v^i values and the 2 new f^{i+1} values.

5. Compute new values for the v^{i+1}, again using only local 1-neighborhood information, by modifying Eq. 21.

The modification is done simply by solving Eq. 20 for e_j^i and substituting this into the last equation to get

$$\begin{aligned} v^{i+1} &= \frac{n-2}{n}v^i + \frac{1}{n^2}\sum_j [4e_j^{i+1} - v^i - f_{j+1}^{i+1} - f_j^{i+1}] + \frac{1}{n^2}\sum_j f_j^{i+1} \\ &= \frac{n-2}{n}v^i - \frac{1}{n^2}v^i + \frac{1}{n^2}\sum_j [4e_j^{i+1} - f_{j-1}^{i+1}] \\ &= \frac{1}{n^2}\left[(n^2-3n)v^i + \sum_j [4e_j^{i+1} - f_j^{i+1}]\right]. \end{aligned} \tag{22}$$

References

1. Chandrajit L. Bajaj and Insung Ihm. Smoothing polyhedra using implicit algebraic splines. *Computer Graphics*, 26(2):79–88, July 1992.
2. Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 113–120. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
3. Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. *Computer Graphics*, 32(Annual Conference Series):85–94, August 1998.
4. Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators in nD. Technical report (submitted for publication), Caltech Multi-Res Modeling Group, July 2000.
5. Nira Dyn, David Levin, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.
6. Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, New York, NY, USA, fourth edition, 1997.
7. P. Thomas Fletcher. Blending and displaying multifigure m-reps using implicit surfaces. Comp236 Course project, Spring semester, Department of Computer Science, University of North Carolina - Chapel Hill, 1999.
8. Gene H. Golub and Charles F. Van Loan. *Matrix Computation*. Series in the Mathematical Sciences. Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.
9. Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 35–44, August 1993.
10. Sven Havemann. Interactive rendering of Catmull-Clark surfaces with crease edges. Technical Report TUBSCG-2001-01, Institute of Computer Graphics, TU Braunschweig, February 2001.
11. Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992.
12. Jörg Peters. Local cubic and bicubic C^1 surface interpolation with linearly varying boundary normal. *Computer Aided Geometric Design*, 7(6):499–516, November 1990.
13. Jörg Peters and Georg Umlauf. Gaussian and mean curvature of subdivision surfaces. In Roberto Cipolla and Ralph Martin, editors, *Proceedings of the 9th IMA Conference*

on the Mathematics of Surfaces (IMA-00), volume IX of *The Mathematics of Surfaces*, pages 59–69, London, Berlin, Heidelberg, September 4–7 2000. Springer.
14. Jörg Peters and Georg Umlauf. Computing curvature bounds for bounded subdivision surfaces. *CAGD: Special Issue on Subdivision*, 18(5):455–461, 2001.
15. William H. Press, Saul A. Teukolsky, William T. Vettering, and Brian P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.
16. Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 395–404. ACM SIGGRAPH, Addison Wesley, July 1998.
17. Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Cambridge, MA, 1986.
18. Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 189–192. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.