A Factored Approach to Subdivision Surfaces

Joe Warren, Scott Schaefer
Rice University[*]

1 Introduction

Polygons are a ubiquitous modeling primitive in computer graphics. Their popularity is such that special purpose graphics hardware designed to render polygons is commonplace. However, modeling with polygons is problematic for highly faceted approximations to smooth surfaces. Since these approximations can consist of hundreds of thousands of polygons, designers cannot be expected to manipulate these approximations directly due to their sheer size.

Subdivision is a technique that solves this problem by representing a smooth shape in terms of a coarse polygonal model. This coarse model can be refined to produce increasingly faceted approximations to the associated smooth shape. The subdivision rules used during this refinement process depend only on the topological connectivity of the initial polygonal model and yield surfaces with guaranteed smoothness.

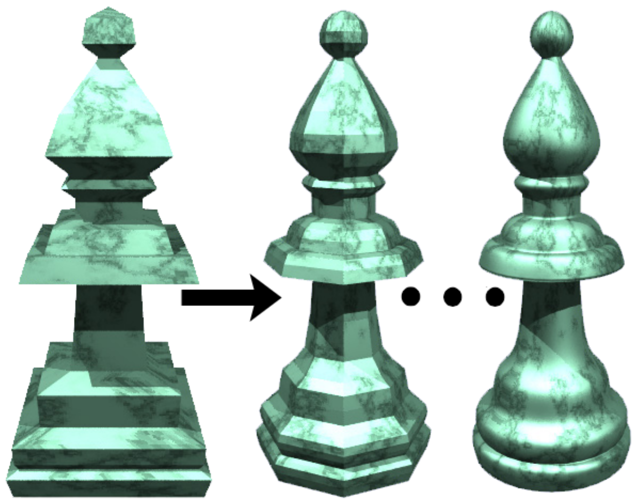

Figure 1: Subdivision of initial coarse model of a bishop (left), subdivided once (middle), smooth limit surface (right).

During refinement, the rules associated with a subdivision scheme are applied recursively to construct a sequence of polygonal models. If these rules are represented by the operator S, this process has the form

$$p^k = Sp^{k-1}.$$

Applying S to an initial model p^0 yields a sequence of polygonal models $p^1, p^2, \ldots$. The rules comprising S specify how the polygonal faces of p^{k-1} are split as well as how the vertices of p^k are positioned in terms of the vertices of p^{k-1}. If these rules are chosen carefully, the limit of this process is a smooth surface p^∞ that approximates the coarse model p^0.

Figure 1 illustrates this process. The left of the figure shows the original, coarse model of a bishop. After one round of subdivision, the bishop contains more faces and begins to resemble the final, smooth shape. Continuing this process yields a smooth model that follows the initial shape. Due to its flexibility and ease of use, subdivision has made its way into several computer generated films and shorts by Pixar as well as being included in many standard modeling packages such as Maya.

This tutorial explains how to implement several different subdivision schemes under a single, unified framework. Our discussion will illustrate Catmull-Clark subdivision [Catmull and Clark 1978] for quadrilateral meshes, Loop subdivision [Loop 1987] for triangular meshes, and a newer combined subdivision scheme called Quad/Triangle subdivision [Stam and Loop 2003] that allows the inclusion of meshes with both quadrilateral and triangular faces. The algorithms that we explain do not require complicated data-structures or mesh traversal algorithms. Instead we focus on methods that illustrate the simplicity of the implementation. Finally, we end with a discussion on generating surfaces that are not smooth everywhere, but instead contain sharp crease curves.

2 Curve Subdivision

We begin with a simple example of a subdivision scheme for curves. Curve subdivision is simpler than surface subdivision while embodying most of the relevant concepts (though in a modified form). Cubic B-splines are a popular class of curves that are smooth and possess a simple subdivision scheme. We study this subdivision scheme first since many surface subdivision schemes (Catmull-Clark, Loop) are based on generalizations of the subdivision rules for cubic B-splines.

Given a polygonal curve p^k, we denote the ith vertex of p^k by p_i^k. The edges of the polygonal curve are implicit in this representation since consecutive vertices (p_i^k and p_{i+1}^k) form an edge. The subdivision rules for cubic B-splines then have the form

$$\begin{aligned} p_{2i}^k &= \tfrac{1}{8}p_{i-1}^{k-1} + \tfrac{3}{4}p_i^{k-1} + \tfrac{1}{8}p_{i+1}^{k-1}, \\ p_{2i+1}^k &= \tfrac{1}{2}p_i^{k-1} + \tfrac{1}{2}p_{i+1}^{k-1} \end{aligned}. \tag{1}$$

Given that there are two rules, the number of vertices in the polygonal curve doubles after each round of subdivision. In particular, the vertex p_i^{k-1} is repositioned to p_{2i}^k while p_{2i+1}^k is inserted at the midpoint of the edge from p_i^{k-1} to p_{i+1}^{k-1}.

Lane and Riesenfeld [Lane and Riesenfeld 1980] observed that the subdivision rules for cubic B-splines could be decomposed into two separate sets of rules: one set for linear subdivision and another for averaging (smoothing) the resulting curve. (This separation of a subdivision scheme into multiple passes is known as *factorization*.) In particular, the rules for linear subdivision have the form

$$\begin{aligned} \hat{p}_{2i}^k &= p_i^{k-1}, \\ \hat{p}_{2i+1}^k &= \tfrac{1}{2}p_i^{k-1} + \tfrac{1}{2}p_{i+1}^{k-1}, \end{aligned}$$

where $\hat{p}^k$ denotes the polygonal curve produced by inserting a new vertex at the midpoint of each edge of p^{k-1}. If we apply an averaging pass of the form

$$p_i^k = \frac{1}{4}\hat{p}_{i-1}^k + \frac{1}{2}\hat{p}_i^k + \frac{1}{4}\hat{p}_{i+1}^k \tag{2}$$

*e-mail: {jwarren,sschaefe}@rice.edu

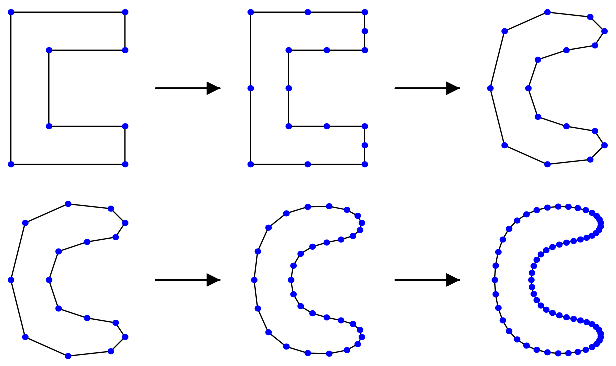

Figure 2: Subdivision of initial curve (top left), after linear subdivision (top middle), after averaging (top right). Further subdivision (bottom).

to $\hat{p}^k$, the resulting polygonal curve is exactly the same polygonal curve p^k produced by equation 1. Figure 2 (top) shows an example of this process where the curve is first linearly subdivided and then averaged.

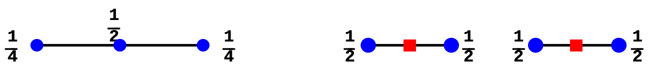

Figure 3: Averaging rule for curve subdivision (left). Geometric interpretation as placing vertex at average of adjacent midpoints (right).

The left portion of figure 3 shows a diagrammatic depiction of the averaging rule of equation 2. The use of these diagrams is common in defining subdivision rules. These diagrams show a local portion of a polygonal mesh with weights attached to each vertex in the diagram. The central vertex in the diagram is typically repositioned by applying the weights in the diagram to their corresponding vertices. The right portion of the figure shows a geometric interpretation of this averaging rule as sequence of two simpler averaging operations. First, the midpoint of each edge in the diagram is computed (depicted as square vertices). Next, the midpoint of these vertices is computed yielding the final position of the central vertex.

Building a geometric interpretation for the averaging rule on the left allows this rule to be generalized to curve networks in which more than two edges meet at a common vertex. The left portion of figure 4 shows several edges meeting at common vertex of valence n (the *valence* of a vertex is the number of edges containing it). Again, the averaging operation consists of computing the midpoints of these edges (square vertices) and then calculating the centroid of these square vertices. The right portion of the figure depicts these two averaging steps written as a single averaging operation.

These general networks of curves naturally partition two-dimensional space. As an application, figure 5 shows an example of using curve networks to separate a mouse brain into its different anatomical regions. The figure shows the curve network created by an anatomist (left) and curve subdivided once (middle). Continuing this process produces the final, smooth network partitioning the brain (right).

While this subdivision scheme for curve networks is useful, its true value lies in the methodology used to generate the scheme. First, we factored a smooth subdivision scheme for curves into linear subdivision followed by an averaging pass. Next, we found a geometric interpretation for this averaging pass that applied to curve networks. In the next section, we apply a similar methodology to surfaces.

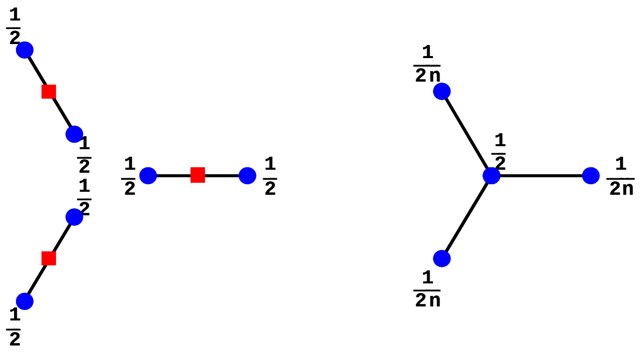

Figure 4: Using the geometric interpretation for curve networks to place a vertex at the average of the midpoints for edges containing that vertex (left). The generalized averaging rule for curve subdivision (right).

3 Surface Subdivision

In the previous section we represented polygonal curves by a list of vertices with the topology of the curve implicit in the indexing of the list. For surfaces, we adopt an explicit topology/geometry representation. A surface is represented as a list of vertices $\{x,y,z\}$ and a list of faces where each face is a list of indices into the vertex list. This indexed data structure is common in graphics as it facilitates the rendering of polygons, and explicitly separates the topology from the geometric position of the vertices.

Quad Subdivision

We begin with quadrilateral subdivision, as this method is the most similar to the curve method shown earlier. As in curve subdivision, we perform subdivision in two steps: linear subdivision and averaging.

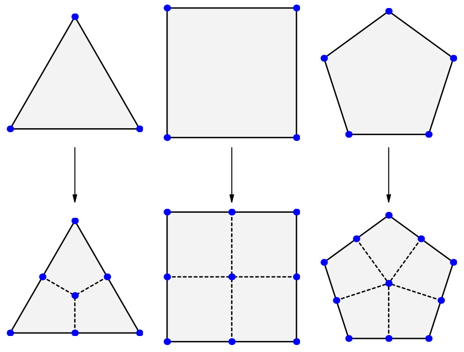

Figure 6: Linear subdivision of polygonal faces for quad subdivision schemes. After one round, all faces are quads.

To perform linear subdivision on a polygonal face, we carry out what is commonly called a Catmull-Clark split [Catmull and Clark 1978] on each face in the mesh. First, we insert new vertices at the midpoints of each edge of the face and one new vertex at the centroid of the face. Next, the vertices are connected to form m quads from the m-sided polygon as shown in figure 6.

Due to the use of a topology/geometry representation for the polygonal mesh, we must ensure that every face sharing a common

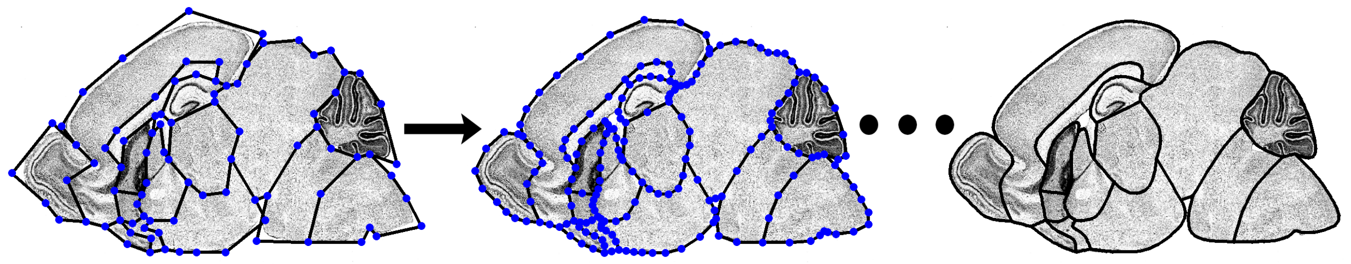
Figure 5: Subdivision of a curve network partitioning a mouse brain into anatomical regions. Initial curve network (left). Subdivided once (middle). Final, smooth curve network (right).

vertex uses the same topological index for that vertex. In particular, if two faces sharing a common edge are subdivided, the index of the new midpoint for that edge must be the same for all four new faces that contain that vertex. We solve this problem by storing the index for each new midpoint on an edge in a hash table keyed by the indices of the two vertices lying at the endpoints of that edge.

The algorithm for linear subdivision is then, for each polygon composed of indices $\{v_1, v_2, \ldots, v_m\}$, check to see if the vertex on edge $\{v_i, v_{i+1}\}$ is already in the hash table. If so, use the vertex index stored in the hash table. If the vertex is not in the hash table, insert a new vertex into the mesh and add the index of that vertex into the hash table with the key $\{v_i, v_{i+1}\}$ (call the new vertex e_i). Add a vertex, c, at the centroid of the polygon into the vertex array. Finally, form the new polygons $\{e_i, v_i, e_{i+1}, c\}$.

Notice that after one round of linear subdivision all of the polygons are quads in the mesh. Also, all new vertices inserted into the mesh after the first round of subdivision will have valence four (valence is the number of polygons containing the vertex). As we continue to subdivide the mesh, any new vertex will be valence four and vertices not of valence four become increasingly isolated. Since almost all vertices of the final surface will be valence four, we call these vertices *ordinary* vertices. Conversely, we call vertices not of valence four *extraordinary* vertices.

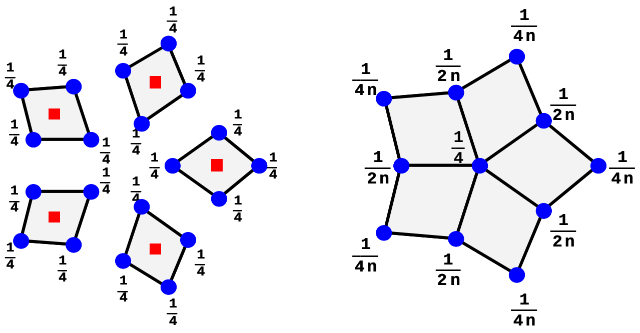

Figure 7: Averaging pass for quad subdivision. Computation of centroids with squares denoting the position of the centroid (left). Averaging the centroids together generates the composite averaging rule at an arbitrary valence vertex (right).

Once linear subdivision is complete, we perform one round of averaging on the mesh. The averaging operation on quadrilateral meshes is analogous to the averaging operation for curves. For each vertex, we place that vertex at the average of the centroids of all quads containing that vertex. Figure 7 (left) shows the centroid calculation for each quad and the composite rule formed by averaging the centroids together (right).

This averaging pass can be implemented as single pass over the list of faces. Before the pass, we initialize each entry of a table of new vertex positions to have value $\{0,0,0\}$. Next, for each quad q, we compute the centroid of q and add this centroid's position to the four entries in this table indexed by the vertices of q. After processing all of the faces in the mesh, we divide each entry in the table by the valence of the vertex associated with the entry. (This valence information can also be computed during the centroid calculations.) Note that dividing by the valence forces the coefficients of the associated averaging rule (shown in figure 7 right) to sum to one and makes the resulting subdivision scheme affinely invariant.

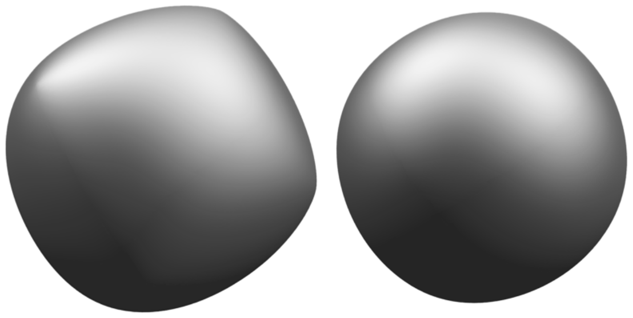
Figure 8: Subdivision of a cube. Uncorrected averaging (left). Catmull-Clark subdivision (right).

Figure 8 (left) illustrates an example surface produced by subdividing a cube using this subdivision scheme. The surfaces produced by this method are C^2 everywhere except at extraordinary vertices where the surface is only C^1. Though the surface is smooth everywhere, the shading of the surface varies rapidly near the valence three vertices of the cube. These discontinuities are due to the fact that the surface normals do not vary smoothly in these regions. (Technically, the surface is strictly C^1.)

To lessen the appearance of these discontinuities, we use the concept of a correction factor introduced by Maillot and Stam [Maillot and Stam 2001]. Let $\hat{p}_i^k$ be a vertex after linear subdivision and p_i^k be the position of the vertex after the averaging pass. We reposition the vertices of the mesh according to the update equation

$$\hat{p}_i^k + w(n)(p_i^k - \hat{p}_i^k)$$

where n is the valence of the vertex and $w(n)$ a function describing the correction factor. The correction factor that we choose for quadrilateral subdivision is $w(n) = \frac{4}{n}$. Notice that this rule does not change ordinary vertices, as their positions remain the same.

Using the correction factor, we generate the surface shown in figure 8 (right). This surface does not contain the discontinuities shown previously and has a much more visually appealing shape. In fact, this correction factor reproduces the popular Catmull-Clark subdivision scheme [Catmull and Clark 1978].

We end the quadrilateral subdivision section with a warning. Though we have developed a subdivision method capable of subdividing arbitrary n-gons, this method should not be used on triangulated surfaces. The reason is that the polygon split in figure 6 introduces an extraordinary vertex of valence three for each triangle where the surface will be only C^1. While use of a correction factor helps smooth the surface in these regions, the resulting surface will contain noticeable visual artifacts. Instead, a triangular subdivision method should be used on these surfaces.

Triangle Subdivision

Though we have developed a subdivision scheme for quadrilateral surfaces, many surfaces encountered in practice are not composed of quads. Instead, the vast majority of surfaces contain triangles as the base modeling primitive. These surfaces are inappropriate for quadrilateral subdivision so we must develop a triangular subdivision scheme. Loop subdivision [Loop 1987] is a very popular subdivision scheme for triangular meshes. As we shall show, Loop's method can also be expressed in terms of linear subdivision and an averaging scheme similar to that for quad meshes.

Unlike the quadrilateral subdivision method described above, our triangular subdivision scheme will only process surfaces composed entirely of triangles. However, this requirement is simple to fulfill as all faces in the mesh can be triangulated.

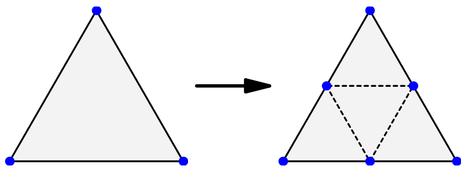

Figure 10: Linear subdivision of triangles for triangular subdivision schemes.

To perform linear subdivision on triangles, we insert new vertices on the edge of each polygon using the hash table technique described for quadrilateral subdivision. Each triangle is then split into four triangles as shown in figure 10. Notice that all new vertices will have valence six in the mesh. Since triangular subdivision produces surfaces with valence six vertices almost everywhere, valence six vertices are ordinary while other valence vertices are extraordinary vertices.

Averaging for triangular surfaces is similar to quadrilateral surfaces. For each vertex in the mesh, we place the vertex at the average of the centroids of all polygons containing that vertex. However, we use a weighted centroid for triangular surfaces shown in figure 11. The centroid takes $\frac{1}{4}$ of the vertex being repositioned plus $\frac{3}{8}$ of the two neighboring vertices. Notice that while the centroid calculation for quads is uniform ($\frac{1}{4}$ of all vertices), the centroid calculation for triangles is not uniform and depends upon which vertex the centroid is being accumulated into. [1]

Figure 12 (middle) shows an example of the triangular subdivision scheme applied to a stellated octahedron. The surfaces produced by this method are C^2 almost everywhere. At extraordinary vertices, the surface are C^1 except valence three vertices where the surface is only C^0. This lack of continuity can be seen clearly in the figure. Though the previous use of a correction factor has been to smooth shading artifacts at extraordinary and not to alter the smoothness of the surface, we must correct the triangular subdivision scheme in order to generate a surface that is smooth everywhere.

[1]Using the more intuitive centroid for triangles ($\frac{1}{3}$ of each vertex) results in a surface scheme that is C^1 everywhere while the weighted method produces C^2 surfaces except at extraordinary vertices.

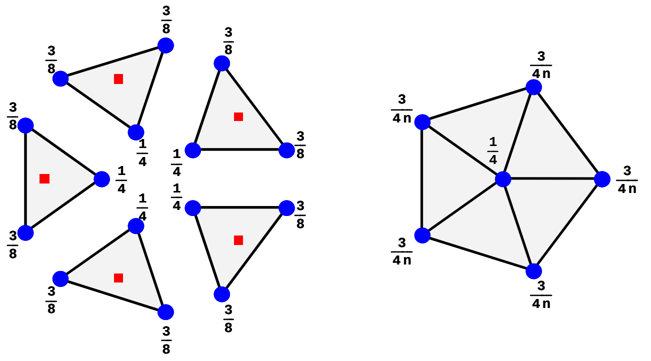

Figure 11: Centroid calculation for triangles (left). Repositioning the vertex at the average of these centroids generates the triangle averaging rule (right).

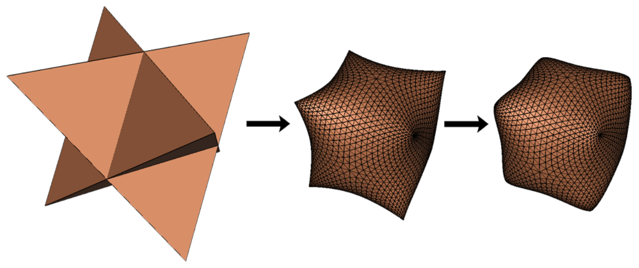

Figure 12: Subdivision of a stellated octahedron (left), uncorrected triangle averaging (middle), corrected subdivision (right).

The correction term that we choose for triangular surfaces reproduces Loop subdivision [Loop 1987]. This subdivision scheme has the property that the surfaces generated are C^2 everywhere except at extraordinary vertices where the surface is C^1. Furthermore, at vertices of valence four and five, the surfaces will have discontinuous, but bounded curvature. While the correction term for quadrilateral subdivision is a simple polynomial ($\frac{4}{n}$), the correction term for triangular subdivision is more complicated and is

$$w(n) = \frac{5}{3} - \frac{8}{3}\left(\frac{3}{8} + \frac{1}{4}Cos\left(\frac{2\pi}{n}\right)\right)^2.$$

Warren [Warren and Weimer 2001] proposed a simpler correction factor of $\frac{8}{n+2}$ that also generates smooth surfaces everywhere but does not have bounded curvature at vertices of valence four and five.

Figure 12 (right) shows the same stellated octahedron subdivided using the corrected triangle averaging method that produces Loop subdivision surfaces. Notice how the corners are now rounded and smooth. Figure 9 contains a more complicated example of a bunny modeled using triangles and subdivided using the correction factor that reproduces Loop subdivision as well.

Combined Quad/Triangle subdivision

So far we have developed methods that subdivide surfaces composed of nearly all quadrilateral polygons or completely of triangles. However, this separation of subdivision schemes between

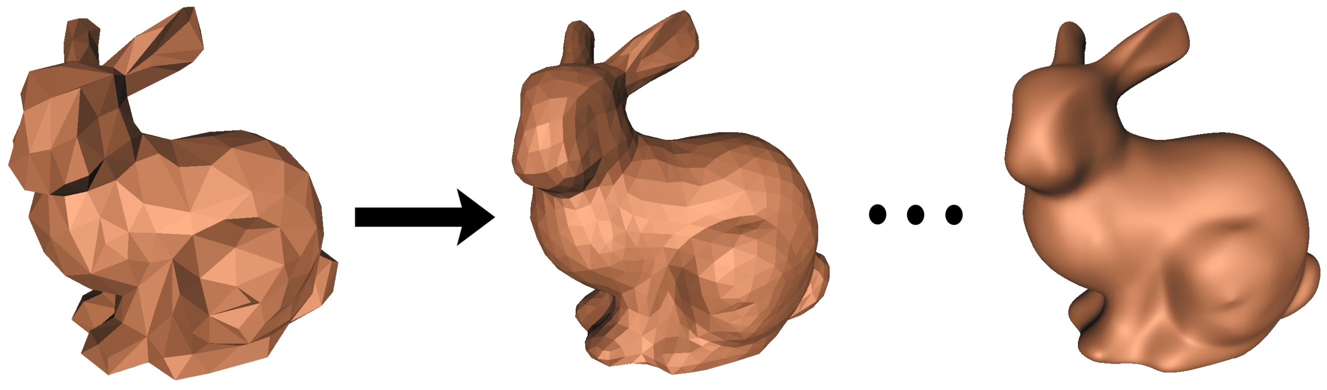
Figure 9: Subdivision of a bunny composed entirely of triangles using the factored Loop subdivision method. Initial mesh (left). Subdivided once (middle). Final smooth surface (right).

the two mostly commonly used modeling primitives is unnecessary. Some surfaces, such as cylinders/tori, are naturally parameterized by quads while other surfaces are more conveniently parameterized by triangles. To remedy this problem, Stam and Loop [Stam and Loop 2003] presented a subdivision scheme that unified these two methods (quads and triangles) into one subdivision scheme that produces Catmull-Clark subdivision for all quadrilateral surfaces, Loop subdivision for all triangular surfaces and generates smooth surfaces when both quads and triangles are present in the surface. We present a variant of Stam and Loop's method, but recast the scheme in terms of a generalized averaging pass. Our quad/triangle subdivision scheme produces Catmull-Clark subdivision on all quadrilateral surfaces, a variant of Loop subdivision on all triangular surfaces and smooth surfaces when the model contains both quads and triangles.

Once again we formulate this subdivision method as linear subdivision and averaging. During linear subdivision, we split all quadrilaterals as done for Catmull-Clark subdivision (shown in figure 6) and all triangles as in Loop subdivision (shown in figure 10). Averaging precedes as before with centroids for quads computed as the average of the four vertices and for triangles as in figure 11 (left). However, we weight each centroid by the angular contribution of that polygon in the ordinary case. For instance, the ordinary case for quad subdivision is four quads containing a vertex so the weight is $\frac{2\pi}{4} = \frac{\pi}{2}$. Likewise, for triangular subdivision there are six triangles containing a vertex in the ordinary case so the weight for triangles is $\frac{2\pi}{6} = \frac{\pi}{3}$. Finally, we normalize by the sum of the weights of the polygons containing each vertex.

In the case of vertices contained by only quads or only triangles, this method produces the same results as the uncorrected quad and triangle averaging methods respectively. Notice that at the boundary where a triangle and a quad meet, linear subdivision will generate the polygonal structure shown in figure 13 (right) all along the edge. In Stam and Loop's paper on quad/triangle subdivision, the authors define the polygonal configuration in figure 13 to be an ordinary boundary between the two surfaces since that structure is replicated along the entire interface between quads and triangles (see figure 14). The averaging rules (applied after linear subdivision) chosen by Stam and Loop are also shown on the right of figure 13. The authors analyzed the smoothness of the surface at this edge and showed that the surface is C^1 across the edge. From this ordinary boundary, the authors then generalized their subdivision scheme to vertices containing an arbitrary number of quads and triangles. The subdivision scheme that we present for quads and triangles differs in the rules used at extraordinary vertices; however, our rules reproduce the subdivision rules of Stam and Loop's method along the ordinary boundary and, therefore, share the same smoothness results along that edge.

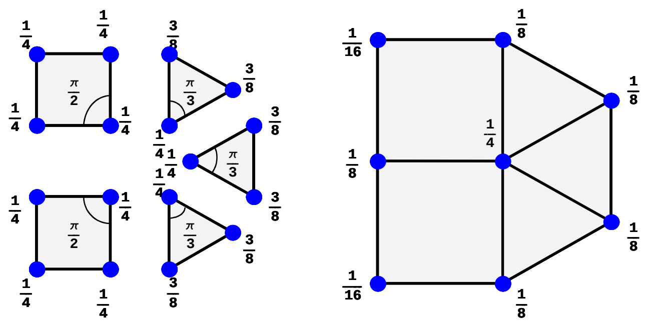

Figure 13: Quad/Triangle subdivision is performed using the centroids from quad and triangle subdivision weighted by their angular contribution: $\frac{\pi}{2}$ for quads and $\frac{\pi}{3}$ for triangles (left). The resulting subdivision rule along a ordinary quad/triangle boundary.

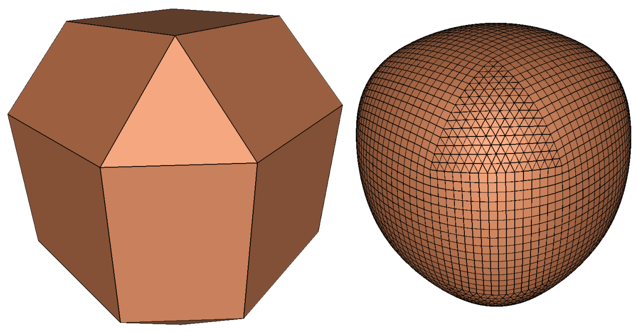
Figure 14: Subdivision of a mixed quad/triangle surface.

Since our quad/triangle method generates triangle averaging on all triangular surfaces, the surfaces will not be smooth at vertices containing only three triangles (as shown in figure 12). To generate smooth surfaces everywhere, we present the correction term

$$w(n_t, n_q) = \begin{cases} 1.5 & n_q = 0, n_t = 3 \\ \frac{12}{3n_q+2n_t} & otherwise \end{cases}$$

where n_q/n_t is the number of quads/triangles containing the vertex. This correction generates Catmull-Clark surfaces with all quadrilateral models and a variant of Loop surfaces with models composed completely of triangles. However, the combined correction term ($\frac{12}{3n_q+2n_t}$) is not smooth at vertices contained by only three triangles. Hence, we use a piecewise function for $w(n_t, n_q)$ that uses the correction value for Loop subdivision at this valence to generate a smooth surface. Stam and Loop also provided a correction term table in their paper that was generated by an optimization method in an attempt to produce surfaces of bounded curvature at low valence vertices. Interestingly, our polynomial correction term is a surprisingly good approximation of that correction table even though our rules differ slightly at extraordinary vertices. Instead of performing our own optimization, we use the provided correction term for simplicity. Figure 14 illustrates an example surface composed of quads and triangles subdivided several times using our hybrid quad/triangle method.

4 Creased Surfaces

So far, all of the surfaces that we have generated with subdivision have been smooth (at least C^1 everywhere). However, many common surfaces are not smooth everywhere, but contain normal discontinuities such as creases and sharp corners. Fortunately, we can augment our existing subdivision schemes to generate subdivision rules capable of creating C^0 discontinuities in an otherwise smooth surface.

To add this ability to subdivision, we use a tagged mesh structure similar to that presented in Hoppe et. al [Hoppe et al. 1994]. We extend the topological elements in a mesh to include not only triangles and quads, but edges and vertices as well. Faces with more than two indices are considered to be polygons, while those with exactly two are edges and one are vertices. To identify sharp creases in the mesh we annotate those edges where the creases should appear by inserting an edge into the topology structure for the mesh. Similarly, inserting a vertex into the mesh identifies a sharp corner.

To alter our subdivision scheme, we introduce the concept of dimension to vertices. We define the *dimension* of a vertex to be the lowest dimensional cell touching that vertex (triangles/quads have dimension two, edges have dimension one, and vertices have dimension zero). To compute the dimension of all of the vertices in a mesh, we first initialize the dimension of all vertices to be two. Next, we perform a single pass over the cells (faces, edges, and vertices stored in the topology list) where we set the dimension of all vertices in each polygon to be the minimum of their current dimension and the dimension of that cell.

We now perform subdivision as before using linear subdivision and averaging. Linear subdivision is unchanged except that edges in the topology list must be subdivided into two edges where vertices are inserted using the hash table technique described in quad subdivision. Next, we compute the dimension of all vertices in the mesh. Finally, we complete one round of subdivision by performing averaging on the mesh. However, we alter the averaging process with respect to the dimension of each vertex. For each cell, compute its centroid (the centroid of an edge is the midpoint and the centroid of a vertex is just the position of that vertex). For all vertices that are contained in that cell, add the centroid to each vertex only if the dimension of the vertex is equal to the dimension of the cell being processed. Figure 15 contains pseudo-code that describes the averaging pass for quad/triangle surfaces with creases.

```
// averagingPass ( Mesh m )

newVert ← array of vertices initialized to {0,0,0}    // initialization
dim ← array containing dimension of each vertex
totalWeight ← array initialized to 0
$n_t$ ← array containing number of triangles touching each vertex
$n_q$ ← array containing number of quads touching each vertex
Vert ← the vertex array in m
for each polygon t in m
    for each vertex $v_i$ in t
        if t is a vertex
            cent ← Vert[$v_i$]
            weight ← 1
        if t is an edge
            cent ← $\frac{1}{2}(\text{Vert}[v_i] + \text{Vert}[v_{i+1}])$
            weight ← 1
        if t is a triangle
            cent ← $\frac{1}{4}\text{Vert}[v_i] + \frac{3}{8}(\text{Vert}[v_{i+1}] + \text{Vert}[v_{i-1}])$
            weight ← $\frac{\pi}{3}$
        if t is a quad
            cent ← $\frac{1}{4}(\text{Vert}[v_i] + \text{Vert}[v_{i+1}] + \text{Vert}[v_{i-1}] + \text{Vert}[v_{i+2}])$
            weight ← $\frac{\pi}{2}$
        if dim[$v_i$] = dimension of t    // only add centroid if same dimension
            totalWeight[$v_i$] += weight
            newVert[$v_i$] += weight * cent
for each vertex $v_i$ in m
    newVert[$v_i$] /= totalWeight[$v_i$]    // normalize vertices by the weights
    if dim[$v_i$] = 2    // apply correction only for quads and triangles
        newVert[$v_i$] ← $\text{Vert}[v_i] + w(n_t[v_i], n_q[v_i]) * (\text{newVert}[v_i] - \text{Vert}[v_i])$

return mesh with topology of m but with newVert as the vertex list
```

Figure 15: Averaging pass for quad/triangle subdivision using dimension for creases

This method generates crease edges in the surface that follow the B-spline curve subdivision technique described at the beginning of this article. The network of crease edges subdivides independent of other vertices in the surface. Therefore, modifying the crease network induces a change in the surface where the surface interpolates the crease network. Also, notice that this description for adding creases is independent of the subdivision scheme (Catmull-Clark, Loop, or Quad/Triangle) as long as the method is expressed in terms of linear subdivision and averaging. Figure 16 (top) shows an example of an umbilic torus represented as a quadrilateral surface containing crease edges with the final, smooth surface textured. Figure 16 (bottom) illustrates an example of a ring that uses both crease edges and crease vertices to obtain its final shape.

Finally, as an example of a culmination of the techniques presented here, figure 17 shows an example of a surface composed of quads and triangles containing crease curves. This three-dimensional model of a mouse brain was built from two-dimensional cross-sections that were annotated with region information by an anatomist. The resulting surface separates the brain into different anatomical regions using a network of surfaces (more than two polygons may share an edge) similar to the two-dimensional partition by networks of curves in figure 5. The model also contains crease curves to control the interface between three or more regions in the brain. The top of the figure shows the complete brain as an initial mesh and the smooth surface resulting from subdivision. On the bottom, the cerebellum has been extracted and the crease curves highlighted on the edges. This model contains a complicated quad/triangle structure and is smooth after subdivision.

5 Conclusion

We have described several different methods for subdividing surfaces composed of quads, triangles, or a combination of both quads and triangles. Furthermore, we augmented our smooth surfaces with the ability to add normal discontinuities such as crease edges and vertices. By separating subdivision into two separate passes (linear subdivision and averaging), we achieved a simple method for applying subdivision that did not require any complicated data-structures or special cases. Also, this separation generated a common framework for different types of subdivision schemes and even led to the construction of a new variant of a subdivision scheme for quad/triangle surfaces.

We end with a note that the subdivision schemes presented here are but a small subset of different methods available. In particular, subdivision schemes are separated into two main divisions: interpolating and approximating. All the averaging methods described here are approximating in that the surfaces do not interpolate the vertices of the original surface. Interpolating methods [Kobbelt 1996; Zorin et al. 1996] (as the name implies) interpolate the vertices of the original surface giving the user a more intuitive feel of the final shape of the surface. However, these surfaces tend to be only C^1 and do not exhibit many of the desirable properties that averaging methods possess.

Acknowledgements
We'd like to thank Tao Ju for providing the figure of the mouse brain for our curve network example and for the 3D surface network example.

References

CATMULL, E., AND CLARK, J. 1978. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design 10*, 350–355.

HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. 1994. Piecewise smooth surface reconstruction. *Computer Graphics 28*, Annual Conference Series, 295–302.

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

LANE, J., AND RIESENFELD, R. 1980. A theoretical development for the computer generation and display of piecewise polynomial surfaces. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, 35–46.

LOOP, C. 1987. Smooth subdivision surfaces based on triangles. *Masters Thesis., University of Utah, Dept. of Mathematics*.

MAILLOT, J., AND STAM, J. 2001. A unified subdivision scheme for polygonal modeling. In *Computer Graphics Forum (Proc. EUROGRAPHICS '01), 20(3)*, 471–479.

STAM, J., AND LOOP, C. 2003. Quad/triangle subdivision. *Computer Graphics Forum 22*, 1, 1–7.

WARREN, J., AND WEIMER, H. 2001. *Subdivision Methods for Geometric Design*. Morgan Kaufmann.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating subdivision for meshes with arbitrary topology. *Computer Graphics 30*, Annual Conference Series, 189–192.

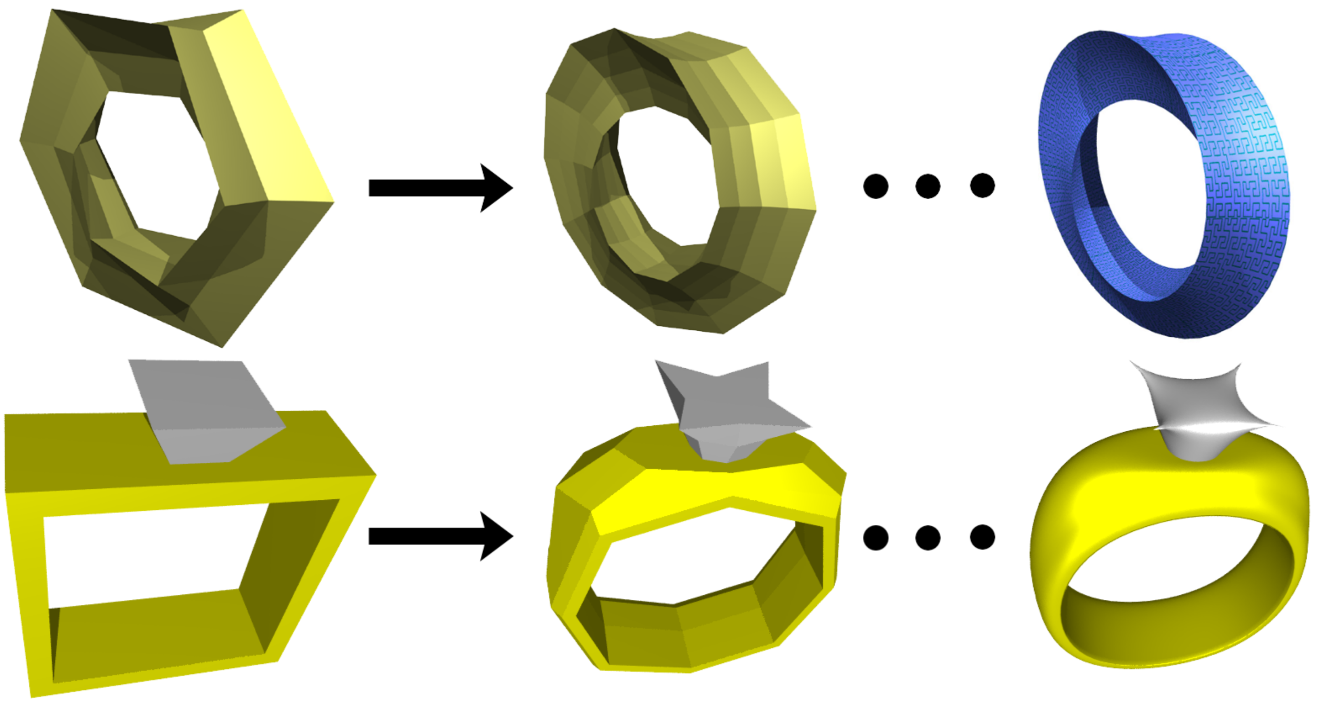

Figure 16: Umbilic torus subdivided using crease curves for the sharp edges (top). Subdivision of a ring containing both crease edges and vertices (bottom).

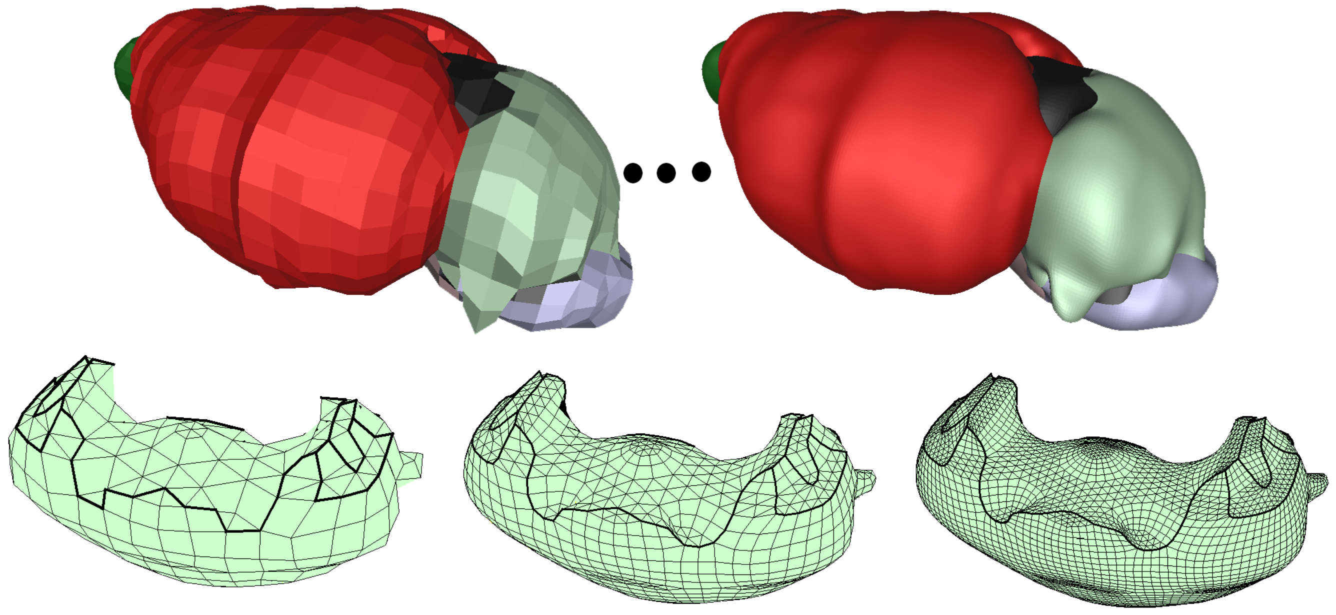

Figure 17: Model of a mouse brain separated into anatomical regions using a surface network as a subdivision surface. Initial model and final, smooth model (top). Subdivision of cerebellum from coarse to fine with crease curves highlighted (bottom).