RICE UNIVERSITY

# A Factored, Interpolatory Subdivision Scheme for Surfaces of Revolution

by

**Scott Schaefer**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**Master of Science**

Houston, Texas

May, 2003

# A Factored, Interpolatory Subdivision Scheme for Surfaces of Revolution

Scott Schaefer

## Abstract

We present a new non-stationary, interpolatory subdivision scheme capable of producing circles and surfaces of revolution and in the limit is $C^1$. First, we factor the classical four point interpolatory scheme of Dyn et al. into linear subdivision plus differencing. We then extend this method onto surfaces by performing bilinear subdivision and a generalized differencing pass. This extension also provides the ability to interpolate curve networks. On open nets this simple, yet efficient, scheme reproduces the curve rule, which allows $C^0$ creases by joining two patches together that share the same boundary. Our subdivision scheme also contains a tension parameter that changes with the level of subdivision and gives the scheme its non-stationary property. This tension is updated using a simple recurrence and, chosen correctly, can produce exact surfaces of revolution.

## Acknowledgements

I'd like to thank Joe Warren, my advisor, who first introduced this problem to me. Also I'd like to thank him for taking the time to discuss my ideas and read the many revisions of this thesis that I produced. Finally I'd like to thank my committee members for their helpful comments and questions.

# Contents

# Illustrations

# Chapter 1

# Introduction

Subdivision is an excellent tool for modeling due to its ability to construct smooth surfaces with minimal effort. Typically, artists manipulate a coarse description of a model representing a smooth surface. The computer then subdivides the model to obtain a new, smooth version that somehow follows the original shape. Moving the vertices on the coarse mesh, also referred to as "control points," affects the shape of the smooth surface and allows for a very intuitive modeling paradigm.

Interpolatory subdivision schemes are a special subset of all subdivision schemes where there is the added requirement that the limit surface interpolates the original control points. These methods can give artists an even more intuitive feel for the shape of the final, smooth surface because the vertices that they position actually lie on the limit surface. One example of an interpolatory scheme for curves is the classic four-point scheme described in [3]. In that paper, Dyn et al. describe a stationary tension parameter that is used to control the shape of the curve. Dyn et al. further extended this interpolatory scheme to triangular surfaces using a butterfly subdivision scheme in [6]. Kobbelt [9] took the same curve scheme and extended the method to arbitrary quadrilateral surfaces. Zorin [14] later proved that Kobbelt's method produces $C^1$ surfaces for all valences of vertices. We propose a method similar to the curve scheme [3] except our tension parameter changes with the level of subdivision, which results in a non-stationary scheme. Our extension to surfaces also differs from Kobbelt's and gives our method the ability to interpolate curve networks.

Recently, there has also been an interest in subdivision schemes that are combinations of simple steps or factored schemes that break apart a single subdivision scheme into simpler passes. These factored methods can be simpler to implement because large, complex subdivision masks can be calculated by performing several small, locally supported passes over the mesh. Lounsbery et al. [10] present a subdivision scheme factored into separate passes over the mesh. The authors describe an approximating subdivision method in two passes: linear subdivision and averaging. First, a temporary mesh is generated by performing linear subdivision on the input mesh and then an averaging pass where vertices are replaced by the average of the centroids of the adjacent faces.

In [12], Stam generalized B-spline surfaces to meshes of arbitrary topology. The even/odd schemes described in the paper are performed by linearly subdividing the mesh and either applying a special mask for the odd scheme or taking the dual and then averaging repeatedly for the even scheme. The surfaces produced by this method are $C^{d-1}$ everywhere (where $d$ is the degree of the surface) except at extraordinary vertices where the surface is only $C^1$.

Zorin and Schröder [15] extended the ideas of Lounsbery et al. and developed an approximating scheme similar to Stam's. In that paper the authors factor B-spline subdivision into linear subdivision plus repeated dualing of the mesh where vertices are positioned at the barycenter of polygons. In this way the authors generalize B-splines of bidegree up to nine and prove the scheme to be $C^1$ at extraordinary vertices. The subdivision scheme that we propose will also be presented as factoring an existing scheme into two simpler steps: linear subdivision and differencing.

However, these subdivision schemes have a disadvantage as well. These methods are unable to model many widely used shapes such as spheres and tori because these surfaces possess no polynomial parameterization. A notable exception is the method of Morin et al. [11]. The authors show how to produce an approximating method that generalizes B-splines to surfaces of revolution by adding a non-stationary tension parameter. This generalization was accomplished by factoring the subdivision scheme into linear subdivision plus a generalized averaging pass that includes the tension parameter. By choosing this tension parameter correctly, exact surfaces of revolution can be produced. Bajaj et al. [1] took this work further and extended the scheme topologically to volume meshes and d-dimensional hypercubes. Factoring the subdivision mask is important in both of these schemes and this strategy allows for an intuitive way to extend the subdivision masks to arbitrary valence vertices.

We develop a new subdivision scheme for surfaces similar to [11] in that we factor our subdivision scheme into two different steps: linear subdivision and differencing. Our method differs in that it is an interpolatory subdivision scheme as opposed to an approximating method. First, we derive a stationary, interpolatory scheme for curves that appears in [3] and show how our method factors into linear subdivision followed by differencing. Next we extend this scheme to quadrilateral surfaces and generalize the different passes to vertices of arbitrary valence. This surface subdivision scheme is then shown to be a subdivision scheme for curve networks. Finally, we derive a non-stationary, interpolatory scheme for curves capable of generating circles and extend that scheme to arbitrary quadrilateral surfaces. Using this non-stationary scheme, we show that by choosing the tension parameter correctly exact surfaces of revolution can be produced.

# Chapter 2

# Stationary Subdivision

We will first construct a stationary, interpolatory subdivision scheme for curves and surfaces. Our ultimate goal is to develop a non-stationary scheme capable of modeling a much wider range of shapes; however, the stationary scheme is clearer both in terms of its explanation and notation. Later, we show how the non-stationary case is a simple extension of the stationary method described next.

## 2.1 Curve Subdivision

We begin by constructing an interpolatory scheme for curves. A subdivision scheme for curves is interpolatory if the method is of the form $p_{2i}^{k+1} = p_i^k$ and $p_{2i+1}^{k+1} = \sum s_{2(j-i)+1} p_j^k$ where $p_i^k$ is the $i^{th}$ vertex at the $k^{th}$ level of subdivision and $s_j$ is the $j^{th}$ coefficient of the subdivision mask. When considering an interpolatory subdivision scheme for curves, we will assume that all of these control points are spaced uniformly in a parametric space (see figure 2.1, left). Our subdivision scheme will then take this set of vertices and produce a new set of vertices on an interval that is twice as fine as the original. Then all that is required is to develop a rule to insert new vertices, $p_{2i+1}^{k+1}$, between each of the given control points because the old vertices are interpolated and, therefore, not modified.

Now we derive the well-known four point scheme of [3] as an example of such a subdivision scheme. Consider four consecutive control points that have uniformly spaced parametric components associated with them. These points define a unique cubic polynomial that interpolates them. The new point inserted into the curve should be positioned in the middle of the four points such that the new point lies

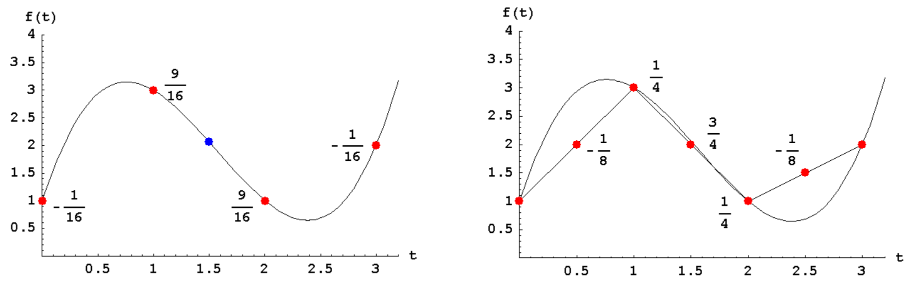

Figure 2.1 : The coefficients used to generate the point in the middle on the cubic polynomial interpolating those four points (left). The coefficients to generate the same point after performing one round of linear subdivision (right).

on the cubic function defined by those points. Positioning the point on this cubic function will give the subdivision scheme the property that if all of the control points are uniformly sampled off a cubic function, then this scheme will reproduce that cubic function. The new point's position can be solved for in terms of the four control points (see figure 2.1, left). Since the points are spaced uniformly in the parametric space, the coefficients will not depend on the spacing of the control points and yields the subdivision scheme in equation 2.1. This is exactly the well-known four point scheme of [2] and [3] with the tension parameter set to 1.

$$\begin{aligned} p_{2i}^{k+1} &= p_i^k \\ p_{2i+1}^{k+1} &= \tfrac{-1}{16}p_{i-1}^k + \tfrac{9}{16}p_i^k + \tfrac{9}{16}p_{i+1}^k - \tfrac{1}{16}p_{i+2}^k \end{aligned} \tag{2.1}$$

We can introduce a tension parameter $\omega$ that blends between this subdivision mask and linear subdivision. The resulting subdivision scheme is then equation 2.2. When $\omega$ is zero, the subdivision scheme is linear subdivision; when $\omega$ is one, this scheme is exactly the four-point scheme in equation 2.1. Dyn et al. [3] show that this new, tensioned subdivision scheme produces a curve that is $C^0$ for $|\omega| < 4$ and $C^1$ for $0 < \omega < 2$.

$$\begin{aligned} p_{2i}^{k+1} &= p_i^k \\ p_{2i+1}^{k+1} &= \tfrac{-\omega}{16}p_{i-1}^k + \tfrac{8+\omega}{16}p_i^k + \tfrac{8+\omega}{16}p_{i+1}^k - \tfrac{\omega}{16}p_{i+2}^k \end{aligned} \tag{2.2}$$

Representing a subdivision mask as a generating function can often be useful when analyzing subdivision schemes [13]. We define a generating function $s[x]$ for curves to be $s[x] = \sum_{i \epsilon Z} s_i x^i$ where $s_i$ is the $i^{th}$ coefficient of the subdivision mask or zero when that coefficient does not exist. We will also represent the control points as a generating function where $p^k[x] = \sum p_i^k x^i$. Then one step of subdivision is represented as $p^{k+1}[x] = s[x] * p^k[x^2]$. One advantage that generating functions provide is that breaking apart a subdivision scheme into separate passes over the mesh is simple (such as linear subdivision and repeated averaging as in [15]); just factoring the generating function provides the necessary tool to split a subdivision step into separate passes over the mesh. We call this subdivision method with multiple passes a *factored subdivision scheme.* These factored subdivision schemes can be easier to implement because each pass can be typically represented on the one-ring of a vertex, which does not require complicated mesh traversal algorithms and data-structures to be built. Accumulating these local passes together can allow very large, complex passes to be calculated in a simple manner.

Representing our subdivision mask as a generating function, $s[x]$ for equation 2.1 is $s[x] = \frac{-1}{16}x^{-3} + \frac{9}{16}x^{-1} + 1 + \frac{9}{16}x - \frac{1}{16}x^3$. If we divide this mask with the generating function for linear subdivision $\frac{(1+x)^2}{2}$, we obtain the subdivision mask $\frac{-x^{-2}+2x^{-1}-1}{8} + 1 + \frac{-1+2x-x^2}{8}$. The right-hand side of figure 2.1 shows the set of control points after linear subdivision and the weights of the points that result in the same subdivision schemes as on the left-hand side of figure 2.1.

One round of subdivision is now: apply linear subdivision and then apply the mask $\frac{-x^{-2}+2x^{-1}-1}{8} + 1 + \frac{-1+2x-x^2}{8}$ to the resulting curve. We can interpret the second mask as taking each point and adding to that point the average of the second differences of the vertices on the adjacent edges. Figure 2.2 shows an example of this process. First, we take the initial curve representing the cross-section of a pawn and perform linear subdivision to produce a new curve. The old vertices from the previous curve are highlighted for clarity. Next, we apply the differencing pass to this new curve, and

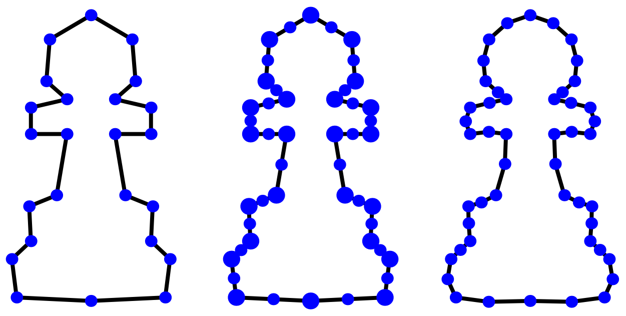

Figure 2.2 : Subdividing a curve by performing linear subdivision and then differencing.

the resulting curve corresponds to one round of subdivision. We can then continue this process to subdivide the curve further if desired.

In this framework linear subdivision isolates the change in topology (adding new vertices) and the differencing mask isolates the positioning of that geometry. Generalization of linear subdivision to surfaces is trivial so we need to concentrate only on generalizing the difference mask. Also notice that before we factored the subdivision scheme, the original vertices were left alone but new vertices had the four-point mask applied to them (see equation 2.1). Now the rule for all of the vertices after linear subdivision has become uniform; the differencing pass is applied to all of the vertices regardless of whether the vertices were originally on the curve or inserted during linear subdivision. The original vertices, $p_i^k$, are interpolated because the second difference mask, $-x^{-1}+2-x$, is zero for the adjacent vertices since these vertices are the result of linear subdivision (i.e. $-p_i^k + 2(\frac{p_i^k+p_{i+1}^k}{2}) - p_{i+1}^k = 0$). While the fact that this rule is uniform is trivial for curve schemes, this factorization will become more important with surfaces as there will not only be original vertices and edge vertices, but face vertices as well.

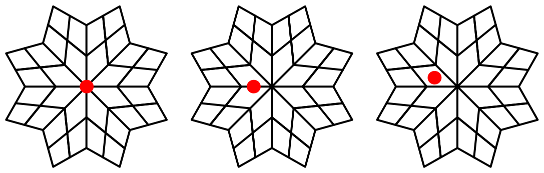

Figure 2.3 : Different types of points that need to be considered when subdividing the mesh. They are (from left to right) vertex, edge, and face points.

## 2.2 Subdivision for Closed Surfaces

Next we generalize the curve subdivision scheme to surfaces by first taking the tensor product of the derived curve scheme's masks. This operation allows us to produce a method for quadrilateral surfaces that works in the regular case where the valence of a vertex is four (we define the valence of a vertex to be the number of quads incident to the vertex). The challenge is to derive a generalization of the masks to extraordinary vertices (valence $\neq 4$) that reproduces the regular case and is smooth in the limit.

Kobbelt [9] took a different approach when he generalized this same mask for curves to quadrilateral surfaces. He considered the three different types of points generated in one round of subdivision: vertex points, edge points, and face points (see figure 2.3). Vertex points were left alone because they were the original control points and don't change in this interpolatory scheme. Edge points were generated by applying the curve rule to the edges of the mesh to produce a new vertex in the middle of an edge. Face points were then an application of the curve scheme to four edge points. Therefore, the only generalization that Kobbelt needed to extend this scheme to quadrilateral meshes was to generalize the curve rule to the case of an edge vertex that is adjacent to an extraordinary vertex. The rule that he derives in the extraordinary case is shown in figure 2.4.

We define a *curve network* to be a set of vertices and edges that connect those vertices. We can define a curve network on a surface by choosing a subset of vertices

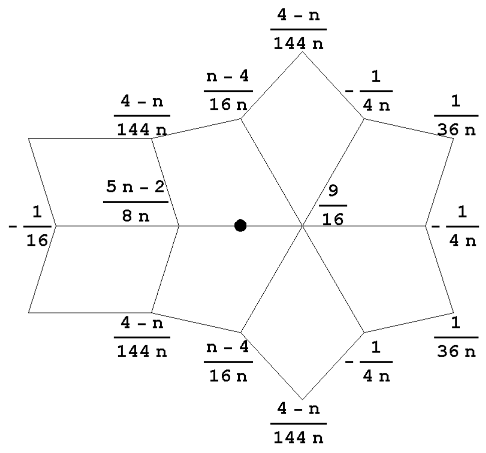

Figure 2.4 : Kobbelt's rule for positioning an edge point near an extraordinary vertex.

and edges on the coarse control polygon to be part of that curve. If subdivision rules are defined for the curve network, then the curve can be subdivided as well. As long as the surface subdivision scheme produces a surface that interpolates the subdivided curve network, then the shape of the surface can be controlled by manipulating the control points of the original curve network. Specifying the position of the curve network gives a very intuitive feel for the shape of the final subdivided surface.

Note that Kobbelt's subdivision scheme cannot be viewed as a subdivision scheme for curve networks because the rule for positioning a new vertex on an edge relies not only on edge adjacent vertices, but face adjacent vertices as well (see figure 2.4). The subdivision method that we present will have the property that the position of new edge points is determined only by edge adjacent vertices and, hence, can be viewed as a subdivision scheme for curve networks.

In contrast to the way Kobbelt extended his subdivision scheme to surfaces, we have factored the subdivision mask into linear subdivision and differencing. We only need to generalize those two masks to surfaces. Generalizing linear subdivision is trivial and just becomes bilinear subdivision. Now all that is left is to generalize the difference mask. Note that this rule is uniform for all vertices: vertex, edge, and face.

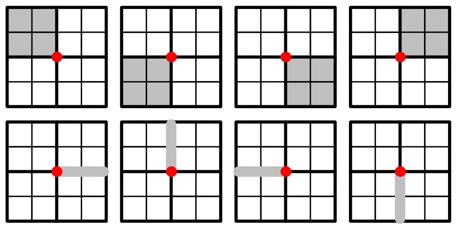

Figure 2.5 : Positioning of a vertex by adding in face differences and edge differences.

This property makes the method easier to implement due to the two uniform masks that are applied to the surface.

To compute the tensor product of the difference mask, we again appeal to generating functions. We can represent this operation as the product of two generating functions by multiplying $s[x] * s[y]$. Therefore, the subdivision mask for surfaces in the regular case is then $s[x, y] = s[x] * s[y] = \sum s_i s_j x^i y^j$. These coefficients can be easily viewed as a 2D matrix.

$$\frac{1}{64}\begin{pmatrix} 1 & -2 & -6 & -2 & 1 \\ -2 & 4 & 12 & 4 & -2 \\ -6 & 12 & 36 & 12 & -6 \\ -2 & 4 & 12 & 4 & -2 \\ 1 & -2 & -6 & -2 & 1 \end{pmatrix} = \frac{1}{64}\left(\begin{pmatrix} 1 & -2 & 1 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \right.$$

$$\left. \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 1 & -2 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ -8 & 16 & -32 & 16 & -8 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right)$$

This scheme is then interpreted as: linearly subdivide the mesh and then for each point add in the average of the face differences (the tensor of the second difference mask shown above) from each of the face-adjacent vertices and the edge differences from the edge-adjacent vertices. Figure 2.5 illustrates this process. After linear subdivision we add in the face differences (represented as gray regions) to reposition the point as shown in the top of the figure. Next we add in the edge differences (highlighted in the bottom of the figure), which produces the final position of the point after one round of subdivision.

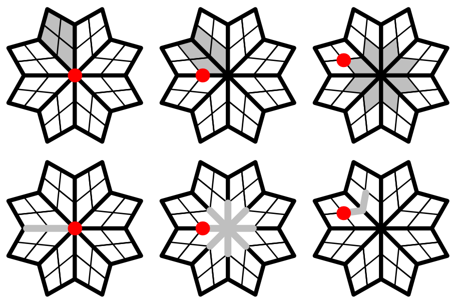

Figure 2.6 : Face differences and edge differences that need to be computed in the extraordinary vertex case.

In the curve case, the original vertices are interpolated because the edge differences are necessarily zero, since the adjacent vertices are the result of linear subdivision. The same is true here; the edge differences are still zero and the face differences are as well. Vertices that are added on the edges of quads during linear subdivision are only affected by the edge differences because the face differences will be zero. Face points are the only points that are affected by both the face differences and the edge differences.

### 2.2.1 Extraordinary Vertices

Unfortunately, this scheme in its current incarnation does not handle extraordinary vertices, that is, vertices with valence other than 4. Figure 2.6 shows the masks that need to be computed in the extraordinary case. The top row shows the face masks that need to be calculated for the different types of points (vertex, edge, and face) and the bottom row shows the corresponding edge masks. As seen on the far right of the figure, we need to generalize the difference operator for faces to extraordinary vertices. The edge difference mask also needs work since it is unclear how the difference mask should behave when centered on an extraordinary vertex (see middle of bottom row).

To calculate the face difference at an arbitrary valence vertex, we process each

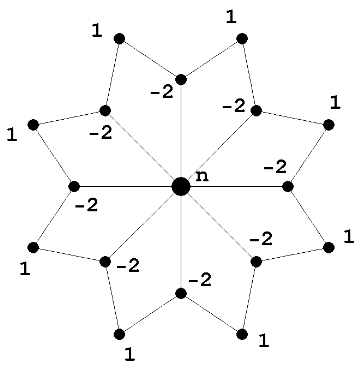

Figure 2.7 : Face difference mask for an arbitrary valence vertex.

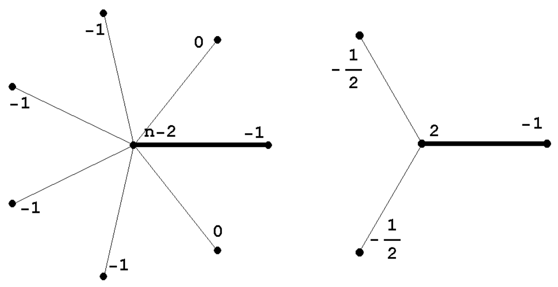

Figure 2.8 : Edge difference rule at an extraordinary vertex across the highlighted edge (left). Edge difference rule at a valence three vertex (right).

polygon containing the specified vertex and accumulate the mask $1-x-y+xy$. This process will result in the mask shown in figure 2.7. This mask is then divided by $4n$ where $n$ is the valence of the vertex.

Extending the edge rule $-x^{-1}+2-x$ to extraordinary vertices requires more work. This rule needs to have the property that, at a vertex of valence four, the rule is $-x^{-1}+2-x$ to maintain the tensor product structure. Figure 2.8 (left) shows the edge rule at an extraordinary vertex. The rule is essentially symmetric except that the differences on edges adjacent to the highlighted edge are zero so that the tensor product case is reproduced. Note that this ordering of the edges is implicitly stored

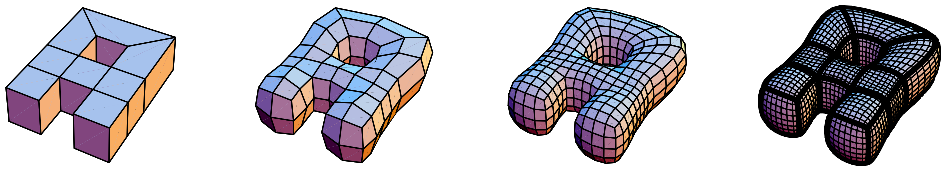

Figure 2.9 : Subdivision of an “A” containing vertices of valence three to five. Far right also shows the curve network interpolated by the surface.

in the polygons containing these vertices. Using this resulting difference, we divide the difference by $2n$ and add the quantity to the vertex to be repositioned.

However, there is a problem with the mask shown in figure 2.8 (left). The mask fails to produce a surface that is $C^1$ in the case of a vertex of valence 3. To overcome this continuity issue, we provide a different rule to use in the case where the valence is 3. Instead of using the difference mask of figure 2.8 (left), we use the mask in figure 2.8 (right). Essentially only half of the difference along the other edges is subtracted out.

Notice that the edge rules that we have described only require other edge vertices. The face differences for edge vertices are necessarily zero and none of the edge differences require face-adjacent vertices. Therefore, the subdivision rules that we have described so far can be viewed as a subdivision scheme for curve networks as well. Specifically, the edges of the original control surface define a curve network that, when subdivided, the surface made up of that network will pass through.

Figure 2.9 shows an example mesh subdivided using these rules for extraordinary vertices. The “A” depicted in the figure contains vertices of valence three on the “feet” and also of valence five where the “feet” connect to the rest of the “A.” The mesh on the far right also shows the curve network defined by the edges of the original mesh that the surface interpolates.

## 2.3 Implementation

This subdivision scheme is surprisingly simple to implement. Previously, we factored a single subdivision step into two passes: linear subdivision and differencing. Therefore, we will implement a single round of subdivision as two separate passes. Although several different masks need to be computed for each pass, each mask will require only the one-ring around each vertex. This local structure of the masks allows us to define a simple mask for each quad that is accumulated for the vertices in the mesh. We will also assume that the geometry being subdivided will be in a topology/geometry format represented as $\{T, G\}$ where $T$ is a list of quads, each of which is a list of four indices, and $G$ is a list of vertices.

The first step in the subdivision process is to perform linear subdivision. This phase can be computed using a linear pass over all of the quads in the mesh. For each quad in the given mesh we divide the quad into four quads. Vertices at the corners of the quad stay the same, but new vertices on the midpoints of the edges and at the centroid of the quad need to be added to the mesh. To avoid adding the same vertex twice to the list of vertices, we use a hash table. Before the four new vertices at the midpoints of the edges are added into the new mesh, each vertex is checked to see if that vertex has already been added to the hash table with its key formed by the indices of the two vertices at either end of that line segment. If the key is already in the hash table, the index associated with that key is used instead of adding a new vertex to the mesh. If the vertex is not found, then a new vertex is inserted both into the mesh and into the hash table. Once all of the quads have been processed, the mesh is complete and is passed on to the next pass in the subdivision process.

The next part of the subdivision process is the differencing pass. Because linear subdivision isolates the change in topology, no alterations are necessary in the topology list during this pass. During the differencing pass we also add various differences into each of the vertices. Therefore, we will initialize our output mesh to be the same as the input mesh and accumulate the various differences into each of the vertices.

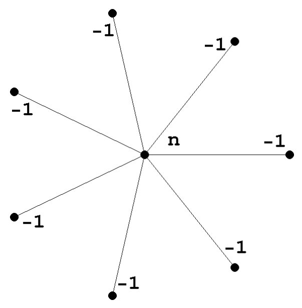

Figure 2.10 : The symmetric mask that is accumulated in edgediff[$i$].

This pass requires that several different quantities be computed for each vertex. First, the valence of each vertex needs to be calculated. This quantity, val[$i$] is found by performing a pass over all of the quads and incrementing val[$i$] for each index, $i$, in the quad's list of indices. Next, the face difference, facediff[$i$], and edge difference, edgediff[$i$], need to be calculated for each vertex.

The face difference (figure 2.7) is relatively simple to compute using a linear pass over all of the quads. Let **face** be the list of vertex indices for the quad that we're currently processing. For each quad in the mesh and for all indices in that quad, we'll compute the mask $\frac{G[\text{face}[j]]-G[\text{face}[j-1]]-G[\text{face}[j+1]]+G[\text{face}[j+2]]}{4\text{val}[\text{face}[j]]}$ and add that quantity into facediff[face[$j$]]. After all of the quads have been processed, facediff[$i$] will contain the face difference mask for the $i^{th}$ vertex as shown in figure 2.7.

The edge difference mask (figure 2.8) is slightly more difficult because the difference mask is not symmetric. However, the mask that we'll compute during this step will be symmetric, but when the difference is added into the vertex to be positioned, edges will be subtracted out to give the difference its final form. For each quad in the mesh and for all indices in that quad, we'll accumulate the mask $\frac{2*G[\text{face}[j]]-G[\text{face}[j-1]]-G[\text{face}[j+1]]}{2}$ into edgediff[face[$j$]]. This process will form the mask shown in figure 2.10 for all of the vertices.

The final step in this pass is to accumulate all of these passes for the various differences together to reposition the vertices resulting from linear subdivision. Again, this step can be computed using a linear pass over all of the quads in the mesh. For each quad and for each index, we'll add into the vertex the face difference associated with the face-adjacent point on the quad divided by the valence of this vertex. The edge differences associated with the two edge adjacent points on the quad will also be added in except that the orthogonal edge on the quad will be subtracted out to produce the mask shown in figure 2.8 (left). The resulting edge differences will then be divided by four times the valence of the edge-adjacent vertex and added into the vertex to be positioned. The extra factor of two in the denominator for the edge-adjacent vertices is because each edge-adjacent vertex will be processed exactly twice since the surface is a manifold. This process can be summarized in the following update equation:

$$\begin{aligned}
G[\text{face}[j]] &\mathrel{+}= \frac{\text{facediff}[\text{face}[j+2]]}{\text{val}[\text{face}[j]]} \\
G[\text{face}[j]] &\mathrel{+}= \frac{\text{edgediff}[\text{face}[j-1]]-2G[\text{face}[j-1]]+2G[\text{face}[j-2]]}{4\text{val}[\text{face}[j-1]]} \\
G[\text{face}[j]] &\mathrel{+}= \frac{\text{edgediff}[\text{face}[j+1]]-2G[\text{face}[j+1]]+2G[\text{face}[j+2]]}{4\text{val}[\text{face}[j+1]]}
\end{aligned} \tag{2.3}$$

The only special case that needs to be considered is in the case of a valence three vertex. In the case where an edge-adjacent vertex is of valence three, only half of the orthogonal edge is subtracted out removing the coefficient of two in the update equation.

# Chapter 3

# Non-Stationary Subdivision

Previously, we developed a stationary subdivision scheme for curves and surfaces, which means that the subdivision rules did not change as a function of the level of subdivision. We now turn to developing a non-stationary algorithm capable of modeling a much wider variety of shapes. This non-stationary algorithm will turn out to be a simple extension of the stationary method presented earlier. We will also see that, by using this new method, we will be able to generate circles and, therefore, surfaces of revolution as well.

## 3.1 Curves

In the stationary case for curves we developed a subdivision scheme by considering the unique cubic polynomial that interpolates four consecutive points spaced uniformly in a parametric space. We will proceed in a similar fashion for our non-stationary subdivision scheme. However, now we use a different set of functions to determine the interpolating curve. Instead of developing an interpolating curve with the functions $\{1, t, t^2, t^3\}$, we use the functions $\{1, t, \cos(t), \sin(t)\}$. This will give the subdivision scheme the property that if the points are sampled uniformly off a function that can be represented as a linear combination of $\{1, t, \cos(t), \sin(t)\}$, then this subdivision scheme will reproduce that function. Unlike the cubic case, these coefficients will be dependent on the spacing between the points due to the use of $\cos(t)$ and $\sin(t)$. Solving for the weights gives the subdivision scheme

$$\begin{aligned} p_{2i}^{k+1} &= p_i^k \\ p_{2i+1}^{k+1} &= \tfrac{-w_k}{16} p_{i-1}^k + \tfrac{8+w_k}{16} p_i^k + \tfrac{8+w_k}{16} p_{i+1}^k - \tfrac{w_k}{16} p_{i+2}^k \end{aligned} \tag{3.1}$$

$$\begin{aligned} w_k &= \frac{1}{\sigma_{k+1}\sigma_{k+2}{}^2} \\ \sigma_k &= \sqrt{\frac{1+\sigma_{k-1}}{2}}, \qquad -1 < \sigma_0 < 1 \end{aligned} \tag{3.2}$$

At first glance, equation 3.1 looks surprisingly similar to equation 2.2 from the stationary case. The difference here is that $w_k$ changes as a function of the subdivision level $k$, which gives this method its non-stationary property. Also note that the recursive definition of $\sigma_k$ is exactly that of the half-angle identity for $\sin(t)$ and $\cos(t)$. Dyn derives this same subdivision method in [5]. However, the resulting scheme is more complicated as the method is in terms of $\cos(t)$ and the half-angle identity is not used.

Ivrissimtzis et al. [8] also develop this same subdivision scheme by assuming that four consecutive control points lie on a circle with uniformly spaced *theta* values associated with them. The authors then solve for the new point on the circle, whose angle is defined by the centroid of the angular components associated with the other four vertices, in terms of the position of the four control points on the circle. Their representation is simpler than that of Dyn's [5]; however, their method is still expressed in terms of $\cos(t)$.

The parameter, $\sigma_0$, can be viewed as a tension that controls the shape of the curve. For $\sigma_0 > -1$ the sequence $w_k$ converges and has a limit of 1. If we set $\sigma_0 = 1$, then $w_k = 1$ for all $k$ and the subdivision scheme becomes stationary and is exactly that of equation 2.1. Developing an interpolating curve with the functions $\{1, t, e^t, e^{-t}\}$ yields the same subdivision scheme again but with $\sigma_0 > 1$. Therefore, the range of the tension has been relaxed to be $\sigma_0 > -1$. Using the techniques described by Dyn and Levin in [4], we can analyze the magnitude of the difference between the coefficients of equation 3.1 and equation 2.1 to show that this subdivision scheme will produce a curve that is $C^1$ in the limit.

We can also analyze the effect that different tension values have on the shape of the curve. Figure 3.1 shows the subdivision of the same control polygon with different tension values. As the tension becomes more positive, the limit curve tends

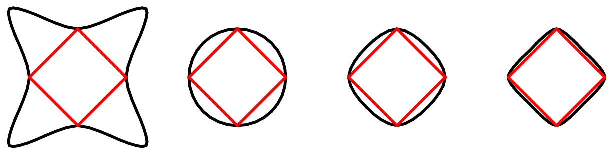

Figure 3.1 : Subdivision of the same square with tensions $-0.9$, 0.0, 1.0, and 5.0 from left to right.

to pull closer to the edges of the original control polygon. Conversely, as the tension becomes more negative the limit curve pulls farther away from the edges of the original polygon. In the case where the tension is set to 1, the curve produced is the same as the stationary method shown earlier. However, when the tension is 0 the curve in this example is exactly a circle. The fact that a circle can be produced is not surprising though because circles can be represented using $\sin(t)$ and $\cos(t)$, which are functions this subdivision scheme is designed to reproduce.

Using generating functions again, $s[x]$ for equation 3.1 is $s[x] = \frac{-w_k}{16}x^{-3} + \frac{8+w_k}{16}x^{-1} + 1 + \frac{8+w_k}{16}x - \frac{w_k}{16}x^3$. This subdivision mask can then be factored by dividing $s[x]$ with the generating function for linear subdivision $\frac{(1+x)^2}{2}$. We then obtain the subdivision mask $w_k\frac{-x^{-2}+2x^{-1}-1}{8} + 1 + w_k\frac{-1+2x-x^2}{8}$. Notice that this scheme is the same subdivision scheme for curves as described in section 2.1 except that the differences at adjacent vertices are weighted by $w_k$. One step of subdivision is then to perform linear subdivision and add in the differences at adjacent vertices weighted by $w_k$.

We can further generalize this scheme to assign different tensions to the different edges as is done in [11]. Each edge in the curve will be associated with a different tension. During linear subdivision, each edge inherits the tension, updated via the recurrence in equation 3.2, from the previous edge that generated the current edge. Given a point whose adjacent segments have tensions $\sigma_0$ and $\hat{\sigma}_0$, the subdivision mask for that point is then $w_k\frac{-x^{-2}+2x^{-1}-1}{8} + 1 + \hat{w}_k\frac{-1+2x-x^2}{8}$ where $w_k$ and $\hat{w}_k$ are the weights from equation 3.2 and $\sigma_0$ and $\hat{\sigma}_0$ are the initial tensions associated with those edges.

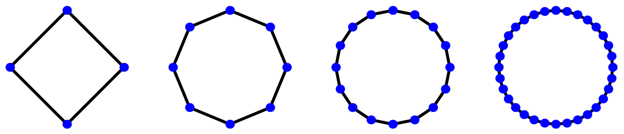

Figure 3.2 : Subdivision of a closed curve starting out as a square with tension 0.0 that converges to a circle.

In figure 3.1 when the initial tension is 0, a circle is produced. In general we can make a regular n-gon converge to a circle by choosing the initial tension correctly. By assigning the tension $\cos(\frac{2\pi}{n})$ to each edge, we can prove that the curve converges to a circle. At every round of subdivision, a regular 2n-gon is produced from the previous regular n-gon whose updated tensions are $\cos(\frac{\theta}{2})$. Taking this process of producing regular n-gons to the limit yields a circle. Figure 3.2 shows the subdivision process for a square converging to a circle.

## 3.2 Surfaces

Our next step is to extend this non-stationary curve scheme to surfaces. In section 3.1, we showed that the non-stationary curve method is a weighted generalization of the stationary curve scheme; the same will be true for our non-stationary surface scheme. Like before, we begin with the tensor product of the differencing mask to extend this subdivision scheme to regular meshes and later generalize the regular vertex case to extraordinary valence vertices.

In the non-stationary curve scheme we provided a generalization to curve segments each having different tension values. We will keep with this generalization when extending to surfaces. Suppose that our surface has tensions $\sigma_x$, $\hat{\sigma}_x$ in the x-direction and $\sigma_y$, $\hat{\sigma}_y$ in the y-direction on the edges incident to a vertex to be repositioned (see figure 3.3). The generating functions for the two curves would then be $s[x] = w_x\frac{-x^{-2}+2x^{-1}-1}{8} + 1 + \hat{w}_x\frac{-1+2x-x^2}{8}$ and $s[y] = w_y\frac{-y^{-2}+2y^{-1}-1}{8} + 1 + \hat{w}_y\frac{-1+2y-y^2}{8}$. The

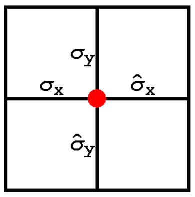

Figure 3.3 : Tensions on edges incident to a vertex to be repositioned in the non-stationary case.

subscript "k" has been dropped from the weights, but it should be understood that the weights depend on the level of subdivision. We can visualize the results of the tensor product operation, $s[x] * s[y]$, as a matrix.

$$\frac{1}{64}\left( w_x w_y \begin{pmatrix} 1 & -2 & 1 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \hat{w}_x w_y \begin{pmatrix} 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + w_x \hat{w}_y \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \end{pmatrix} + \right.$$

$$\hat{w}_x \hat{w}_y \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 1 & -2 & 1 \end{pmatrix} + w_x \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -8 & 16 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \hat{w}_x \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 16 & -8 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} +$$

$$\left. \hat{w}_y \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \end{pmatrix} + w_y \begin{pmatrix} 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right)$$

This subdivision scheme has a particularly simple interpretation in terms of the edge differences and face differences formulated for the stationary, surface method. For each vertex after linear subdivision, add into the vertex the edge differences times the weight on that edge plus the face differences times the product of the weights on the two edges of that face incident to that vertex. Tension assignment for the new edges after linear subdivision is the same as in the curve case. However, we need to define what the tensions are for the edges created interior to a quad during linear subdivision. If a quad has tensions $\sigma_x$ and $\hat{\sigma}_x$ on two of its parallel edges, then we define the tensions of the new interior edges of that quad parallel to those edges

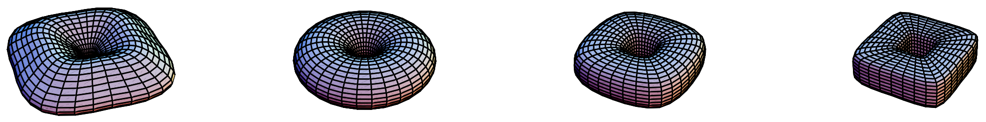

Figure 3.4 : Subdivision of a torus with tensions $-0.5$, $0.0$, $1.0$, and $5.0$.

to be the average of those two tensions. Notice that, like the non-stationary curve method before, this surface method is just a weighted version of the stationary surface scheme presented earlier. In fact, all of the mechanics developed previously to extend the stationary scheme to extraordinary valences work for the non-stationary scheme as well. Figure 3.4 shows an example of this non-stationary subdivision scheme applied to a torus with different tension values.

## 3.3 Surfaces of Revolution

In the non-stationary curve case (section 3.1) we could reproduce a circle with the correct choice of tensions. Since we constructed the surface scheme by taking the tensor product of the curve scheme, we should be able to produce exact surfaces of revolution, that is, surfaces whose cross-section is exactly a circle. In the curve case the tensions were chosen to be $\cos(\frac{2\pi}{n})$ to produce a circle from a regular n-gon. The same will be true with surfaces. If all of the weights are chosen along the cross-section to be $\cos(\frac{2\pi}{n})$ and the object's cross-section is a regular n-gon, then that cross-section will converge to a circle and will produce a surface of revolution.

Figure 3.5 shows the subdivision of an object that converges to an exact sphere. At first the object may seem to consist only of triangles. That is not the case. The quadrilaterals that make up the object have been collapsed. Two vertices of each quad are identified to be the same point in the topological sense. Furthermore, we alter

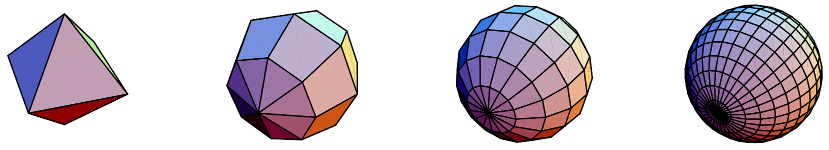

Figure 3.5 : Subdivision of a sphere. All tensions are set to zero and points at the poles are identified so that a sphere is reproduced.

linear subdivision to return the same point topologically when interpolating between two topologically identical vertices. The two poles on the sphere are located where the identified points are on the original mesh.

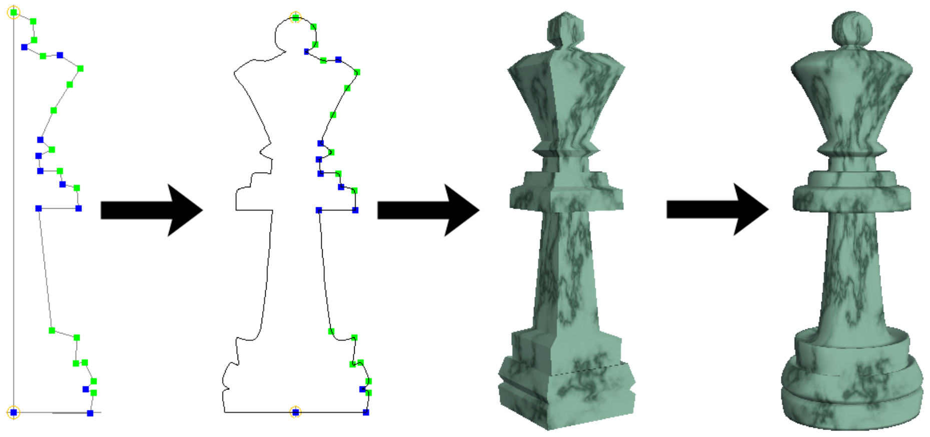

Figure 3.6 : Creating a surface of revolution using our tool. The final surface produced has the same profile as the specified curve.

We have also implemented a program using these results that allows the user to create surfaces of revolution very easily. The entire process is shown for a queen chess piece in figure 3.6 and uses the subdivision rules defined in section 4.2 to create the normal discontinuities seen. First the user specifies control points for a curve that represents the profile of the object (figure 3.6, left). The program then reflects these control points over the axis of revolution and subdivides the curve using the method described in section 2.1 to generate a complete profile of the surface. Next, the program constructs a base surface by revolving the curve around the axis of revolution to form a surface. Finally, this surface is subdivided using the techniques in section 3.2 to produce the final surface. This surface will have the same profile as the curve specified by the user and the cross section of the surface will be exactly a circle.

## 3.4 Weighted, Stationary Surfaces

When Kobbelt [9] generalized equation 2.2 to surfaces, the result was a weighted, stationary surface scheme. Previously, we presented a non-stationary, weighted subdivision scheme for surfaces. The subdivision scheme was just a simple weighted version of the non-weighted subdivision scheme in section 2.2. We can produce a weighted, stationary subdivision scheme for surfaces by weighting the differences by a stationary weight, $w$. This generates an equivalent subdivision scheme to the method in [9] except for the generalization to extraordinary valence vertices. However, our generalization has the added advantage that our subdivision scheme can be viewed as a subdivision scheme for curve networks as well.

# Chapter 4

# Normal Discontinuties

So far the surfaces that we have discussed have been surfaces that are $C^1$ everywhere. Occasionally the artist may desire to construct surfaces that have normal discontinuities. For instance, an artist may want a normal discontinuity to follow a line on the surface to construct an edge in the model. Other discontinuities include point discontinuities. These are areas where the normals diverge on the surface only at a single point.

There are a variety of techniques available that give subdivision the ability to construct such creases in the resulting limit surface. One such method is by defining what happens to the surface on open surfaces, that is, surfaces that are not closed. Then a curve discontinuity in the normals can be constructed by making two open surfaces with identical boundaries. If the subdivision rules along the open edge are chosen correctly, then the surface will appear closed except that the normals on either side of the boundary will not converge and a crease will be produced in the surface. Kobbelt took a similar approach when constructing normal discontinuities in his subdivision scheme [9].

Another method for developing normal discontinuities in subdivision surfaces is by using a tagged mesh format as was done in [7]. In this method an arbitrary surface is taken as input and is annotated with a set of curves and points that describe where the normals on the surface should diverge (i.e., where creases should appear). The subdivision rules are then altered to take these creases into account when constructing a new surface to generate these desired features.

This method of tagging discontinuities has a number of advantages over using open surfaces to represent creases. Creases are much easier to construct using the tagged mesh format instead of breaking the mesh apart to form two open surfaces. Point discontinuities are handled much easier as well, since it is not immediately clear how to even construct a point discontinuity using open surfaces.

This chapter will discuss the creation of these normal discontinuities in the surface. Although the method of constructing creases using open surfaces is less desirable than using a tagged mesh format, open surfaces do arise in practice and are not always used to construct creases. Therefore, we will show how this method extends to open quadrilateral nets. Then we will move on to building creases using a tagged mesh format. Finally, we will end with a method for subdividing non-manifold surfaces.

## 4.1 Open Quadrilateral Nets

As of yet we have not discussed what happens on the boundary of the surfaces defined by this method. If the subdivided boundary edges follow the curve subdivision scheme defined in section 2.1, then $C^0$ creases could be formed in the surface by constructing two open patches that share the same boundary. We'll generate this property by altering the edge differences that are accumulated into the vertices during the differencing pass in the surface subdivision scheme. The guiding principle of this generalization will be to try to reproduce the curve rule along the edges of the mesh. This approach will preserve the ability of this subdivision scheme to be interpreted as a subdivision scheme for curve networks even in the presence of an open mesh.

In section 2.3 we calculated a symmetric edge difference for each vertex (see figure 2.10). When that edge difference is added into the vertex to be repositioned, the edges orthogonal to the edge in the curve are subtracted out to produce the edge difference mask in figure 2.8. We'll proceed in the same way making sure that differences along orthogonal edges are zero. In order to do generate these differences, we'll introduce the concept of an *edge valence*. The valence of an edge in the mesh is

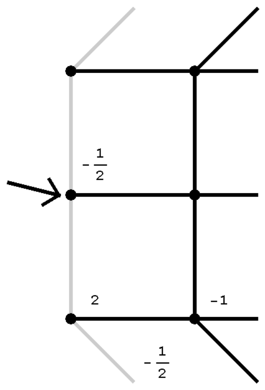

Figure 4.1 : Repositioning an edge vertex on the boundary, which is highlighted by the arrow. Gray lines are boundary edge. Weights are the accumulated edge difference, edgediff[$i$], for an adjacent vertex.

defined to be the number of quads that contain that edge. Notice that the valence of every edge is two in the case of a manifold surface and that boundary edges have a valence of one.

The way that we will accumulate the edge differences, edgediff[$i$], is the same as in section 2.3 by walking over each polygon containing the vertex and accumulating the mask $1 - \frac{1}{2}x - \frac{1}{2}y$. For closed meshes each edge is counted twice and we obtain the original mask. However, on a open mesh we obtain the mask shown in figure 4.1. In this figure the gray lines depict the boundary edges of the net.

In section 2.2 we designed the edge difference mask to reproduce the curve rule along the edges of the mesh in the tensor product case. We'll do the same thing here to produce the curve rule along the boundary edges by accounting for the fact that edges in the edgediff[$i$] quantity have different weights associated with them according to the valence of that edge (see figure 4.1). In the update equation of equation 2.3, the edge difference along the orthogonal edge is subtracted out of edgediff[$i$]. To account for the differing weights on the edges, we'll subtract off the edges weighted by the valence of that edge divided by two. Because this edge difference will be

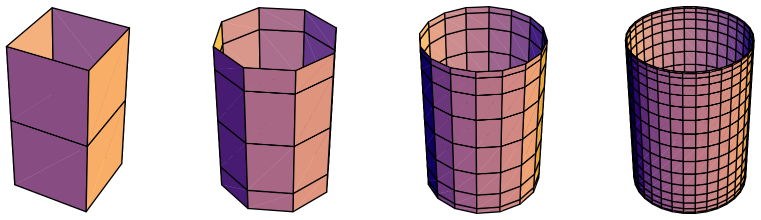

Figure 4.2 : An open, circular cylinder subdivided with all tensions set to 0.0.

added into the vertex to be repositioned once for each quad containing that edge, we will also divide the edge difference by the valence of that edge. This process can be summarized in the modified update equation:

$$\begin{aligned}
G[\text{face}[j]] \quad &+= \quad \frac{\text{facediff}[\text{face}[j+2]]}{\text{val}[\text{face}[j]]} \\
G[\text{face}[j]] \quad &+= \quad \frac{2*\text{edgediff}[\text{face}[j-1]]+\text{val}[\text{face}[j-1],\text{face}[j-2]]*(G[\text{face}[j-2]]-G[\text{face}[j-1]])}{4*\text{val}[\text{face}[j-1]]*\text{val}[\text{face}[j],\text{face}[j-1]]} \\
G[\text{face}[j]] \quad &+= \quad \frac{2*\text{edgediff}[\text{face}[j+1]]+\text{val}[\text{face}[j+1],\text{face}[j+2]]*(G[\text{face}[j+2]]-G[\text{face}[j+1]])}{4*\text{val}[\text{face}[j+1]]*\text{val}[\text{face}[j],\text{face}[j+1]]}
\end{aligned}$$

With the generalization of the edge difference mask complete, there is still the question of how the face difference mask behaves on boundaries. Remember that the face difference mask is necessarily zero for all vertices except for face vertices that are inserted into the mesh (see figure 2.3). The only thing that we'll do for the face differences is to truncate them if the mesh is absent in those areas. This truncated mask has the advantage of having a very simple implementation in terms of the method described in section 2.3 as the method does not need to be modified at all.

Figure 4.2 illustrates the subdivision of an open cylinder as an example of the behavior of this method on the boundary of the mesh. The tensions in the mesh are set uniformly to zero so that the cross-section of the surfaces approaches a circle and a surface of revolution is produced as discussed in section 3.3. Because the subdivision

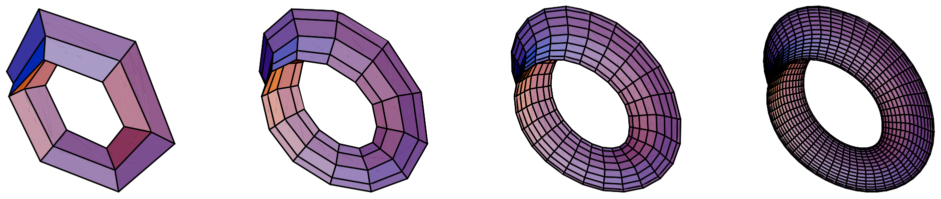

Figure 4.3 : Umbilic torus-like surface modeled as an open surface using tensions.

rules reproduce the curve case along the open edges of the mesh, the boundary of the surfaces converges to a circle as well.

Figure 4.3 illustrates subdivision of an umbilic torus-like surface. Although not a surface of revolution, the surface utilizes tensions to achieve its shape. Despite the appearance of the surface, the shape is not modeled as a closed surface. Instead, the torus is a single strip of polygons that winds around three times rotating $\frac{1}{3}$ of a rotation each revolution. The boundary edges match up perfectly to give the appearance of a closed surface. The tensions are chosen to be uniformly $\cos(\frac{2\pi}{6})$ because there are six sectors in one revolution. Since we chose the boundary rules to reproduce the curve case, no cracks appear on the edges. This figure also illustrates the use of open surfaces to form creases in the mesh as is seen by the $C^0$ line that appears where the edges of the surface meet.

## 4.2 Tagged Discontinuities

Tagged discontinuities provide a superior method for introducing creases into a mesh over using open meshes to describe creases. Usually these meshes are represented in a topology/geometry format as discussed in section 2.3. The topology here is represented as a list of quadrilaterals. To introduce a crease point or line all that is needed is to insert a point or a line into the list of quadrilaterals. These lower dimensional faces are the “tags” that identify where normal discontinuities should appear in the surface.

The rules for subdivision need to be modified in order to accommodate these tags that appear in the topology. Factoring the subdivision scheme into linear subdivision plus differencing makes these modifications quite easy as the changes can be described in terms of the edge and face differences from section 2.2. Furthermore, the crease lines introduced into the mesh will follow the same curve subdivision rules described in section 2.1.

In order to make these modifications we will need the concept of the *dimension* of a vertex. The dimension of a vertex is defined to be the dimension of the lowest dimensional face containing that vertex. Quadrilaterals are of dimension two, lines of dimension one, and points are of dimension zero. For example, if a vertex has five quadrilaterals touching it as well as two lines, then the vertex is of dimension one. The dimension of all of the vertices on a mesh can be calculated using a simple linear pass over all of the faces in the topology list.

Now all of the concepts introduced previously need to be redefined in terms of the dimension of the vertices. The valence of a vertex is now defined as the number of faces containing the vertex of the same dimension as that vertex. The valence of an edge is defined to be the number of faces containing that edge that are of the same dimension as the lowest dimensional vertex on either end of that edge. Also, when calculating the various differences (edge and face) for each vertex, the masks are restricted to faces that are of the same dimension as the vertex that the differences are being calculated at.

The rules for subdivision of a mesh are now very simple in terms of these modified definitions. Linear subdivision is not changed at all. However, the differencing pass is changed in one small way. When processing a vertex of dimension two, only add the edge and face differences into the vertex if the vertex centered at the respective difference is of the same dimension as the vertex to be repositioned. However, when

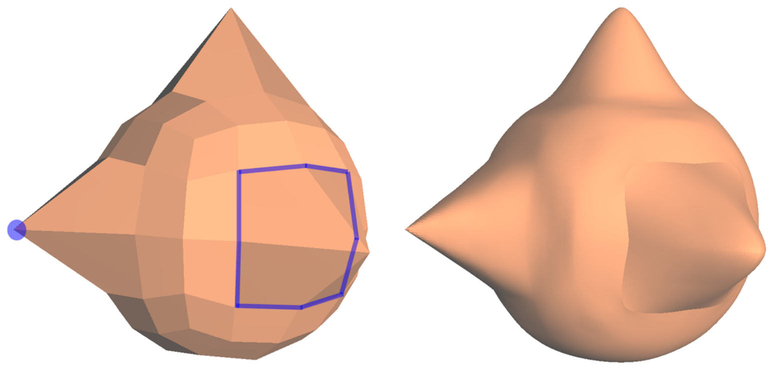

Figure 4.4 : Example of subdividing a mesh with normal discontinuities described using a tagged mesh format. Tagged edges and vertices are highlighted on the coarse model for clarity.

processing a vertex of dimension one, only add the edge differences into that vertex of the vertices edge adjacent to that vertex that are of dimension one as well. Dimension zero vertices are not modified at all because this is an interpolatory subdivision scheme and no new dimension zero vertices are added during the course of subdividing the mesh.

Figure 4.4 (left) shows an example of a mesh before subdivision with the crease edges and points highlighted. On the right of the figure is the same surface after several rounds of subdivision. Notice that the normals of the surface diverge in the specified areas.

Finally, figure 4.5 contains an example that merges the tagged discontinuity method with the surfaces of revolution presented in section 3.3. This model of a king chess piece is modeled almost completely as a surface of revolution with several tagged edges producing circular creases in the model. However, the top of the model contains a six-point cross that does not have a circular cross-section. In fact, the cross contains crease edges along all of the edges of the cross and crease points at its vertices to force linear interpolation of this portion of the mesh.

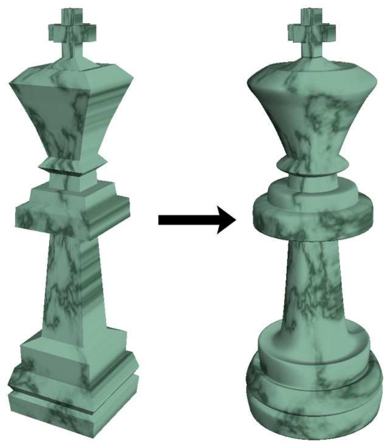

Figure 4.5 : A model with several creases (point and edge) that generate the final shape.

## 4.3 Non-Manifold Subdivision

A manifold point on a surface is a point where there exists a small neighborhood around the point such that the surface is topologically a disk. Non-manifold points may exist on some surfaces as well as non-manifold lines where an edge in the mesh is shared by more than two polygons. Although it is not immediately obvious how non-manifold surfaces fit into a chapter on normal discontinuities, the normals of the surface typically will diverge around the non-manifold point or line.

The subdivision rules for non-manifold surfaces turn out to be very simple to construct. The theme here is the same as before: reproduce the curve rule along the edges of the surface. In section 4.1 this technique was used to construct rules for open surfaces. In fact, the same rules developed in section 4.1 work for non-manifold edges as well. The case of a non-manifold point, however, is not handled, but it is very rare that any arise in practice.

Figure 4.6 shows an example of subdivision of an acorn that is represented as a non-manifold surface where the initial weights are set uniformly to zero to reproduce a surface of revolution. The bottom portion of the figure shows a cross-section of the

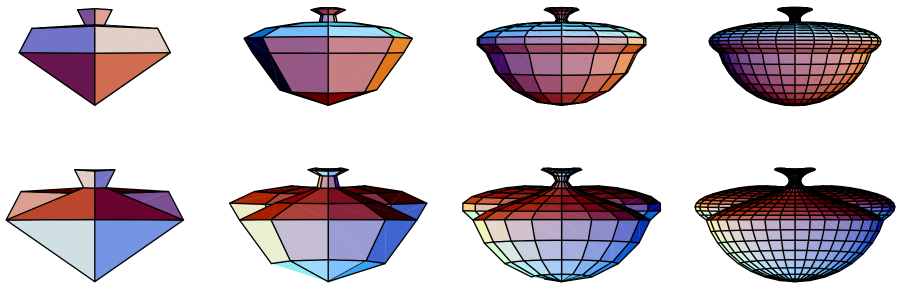

Figure 4.6 : Subdivision of an acorn modeled as a non-manifold surface.

surface for several different levels of subdivision to illustrate the non-manifold edges present in the surface. Notice that the surface is only $C^0$ along the non-manifold edges.

# Chapter 5

# Conclusion

We have derived a non-stationary, interpolatory subdivision scheme for curves and arbitrary quadrilateral nets capable of reproducing circles and surfaces of revolution. The curve method was extended to surfaces and extraordinary vertices by factoring the subdivision scheme into linear subdivision and differencing. This extension revealed that this subdivision scheme could also be thought of as a subdivision scheme for curve networks. The non-stationary property of this subdivision scheme was given by a tension parameter that changed with the level of subdivision and was updated using a simple recurrence. We also generalized the range of this tension parameter, which allowed for a wide range of shapes to be produced. Next, we showed that the non-stationary subdivision scheme presented was just a weighted version of the stationary subdivision scheme. Finally, we ended with two different techniques based on open meshes and tagged meshes to introduce normal discontinuities into the subdivided models.

# Bibliography

[1] Bajaj, C., Schaefer, S., Warren, J., and Xu, G. A subdivision scheme for hexahedral meshes. *The Visual Computer 18* (2002), 409–420.

[2] Deslauriers, G., and Dubuc, S. Symmetric iterative interpolation processes. *Constructive Approximation 5* (1989), 49–68.

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

[4] Dyn, N., and Levin, D. Analysis of asymptotically equivalent binary subdivision schemes. *Journal of Mathematical Analysis and Applications 193* (1995), 594–621.

[5] Dyn, N., and Levin, D. Subdivision schemes in geometric modelling. *Acta Numerica 12* (2001), 1–72.

[6] Dyn, N., Levin, D., and Gregory, J. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics (TOG) 9*, 2 (1990), 160–169.

[7] Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W. Piecewise smooth surface reconstruction. *Computer Graphics 28*, Annual Conference Series (1994), 295–302.

[8] Ivrissimtzis, I., Dodgson, N., Hassan, M., and Sabin, M. On the geometry of recursive subdivision. *International Journal of Shape Modeling 8*, 1 (June 2002), 23–42.

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

[10] Lounsbery, M., DeRose, T. D., and Warren, J. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics 16*, 1 (1997), 34–73.

[11] Morin, G., Warren, J., and Weimer, H. A subdivision scheme for surfaces of revolution. *Computer Aided Geometric Design 18* (2001), 483–502.

[12] Stam, J. On subdivision schemes generalizing b-spline surfaces of arbitrary degree. *Computer Aided Geometric Design 18* (2001), 383–396.

[13] Warren, J., and Weimer, H. *Subdivision Methods for Geometric Design: A Constructive Approach.* Morgan Kaufmann, 2002.

[14] Zorin, D. A method for analysis of $c^1$-continuity of subdivision surfaces. *SIAM Journal of Numerical Analysis 37* (2000), 1677–1708.

[15] Zorin, D., and Schröder, P. A unified framework for primal/dual subdivision schemes. *Computer Aided Geometric Design 18*, 5 (2001), 429–454.