A Factored Interpolatory Subdivision Scheme for Quadrilateral Surfaces

Scott Schaefer, Joe Warren

Abstract. This paper presents a new C^1 interpolatory subdivision scheme for quadrilateral meshes based on the idea of factoring a complex scheme into several simpler passes. To illustrate this point we first factor the well-known four point rule for interpolatory curve subdivision into linear subdivision followed by a simple differencing pass. This subdivision method is then extended to surfaces by generalizing each of these passes to quadrilateral meshes (including those with extraordinary vertices). Finally, we demonstrate that this extension can also be interpreted as defining a subdivision scheme for interpolating curve networks.

§1. Introduction

Subdivision has become an excellent tool for modeling due to its ability to construct smooth surfaces with minimal effort. Using these techniques, artists are able to manipulate a coarse description of a model. The computer then subdivides the model to obtain a new, smooth version that follows the original shape. Moving the vertices of 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 the 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 [2]. Kobbelt [3] provided an interpolatory subdivision scheme for arbitrary quadrilateral meshes that extended the curve method

Curve and Surface Fitting: Saint-Malo 2002
XXX (eds.), pp. 1–10.

in [2], and Zorin [7] proved that Kobbelt's surface scheme was C^1 for all valences of vertices.

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. [4] present a subdivision scheme as being 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.

Zorin and Schröder [8] extended the ideas of Lounsbery et al. and developed an approximating scheme that generalized B-splines of bidegree up to nine. 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. Stam [5] produced a similar method that generalized B-splines of arbitrary degree to surfaces using an even/odd technique. Although both methods are capable of producing surfaces with high degrees of continuity, the surfaces produced are still only C^1 at extraordinary vertices.

We develop a new subdivision scheme for surfaces similar to [4] in that we factor our subdivision scheme into two different steps: linear subdivision and differencing. Our method also 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 [2] and show how it factors into linear subdivision followed by differencing. Next, the curve scheme is extended onto quadrilateral surfaces by generalizing the different passes onto vertices of arbitrary valence. Finally, we show how this surface subdivision scheme is a subdivision scheme for interpolating curve networks as well.

§2. Curve Subdivision

We begin by constructing an interpolatory scheme for curves. A subdivision scheme for curves is interpolatory if it 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 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

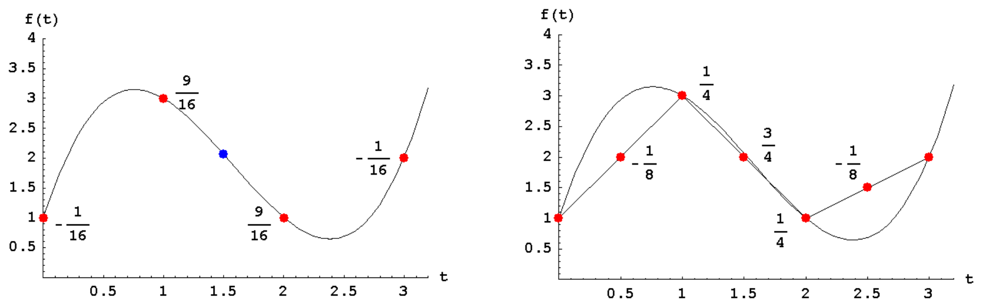

Fig. 1. The coefficients used to generate the point in the middle on the cubic polynomial interpolating those four points (left). The coefficients to generate the same point after performing one round of linear subdivision (right).

vertices, p^{k+1}_{2i+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 [2] 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 being inserted into the curve should be positioned in the middle of the four points and such that it lies on the cubic function defined by those points. This 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 then be solved for in terms of the four control points (see figure 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 (1). This is exactly the well-known four point scheme of [1] and [2] with the tension parameter set to 1.

$$\begin{aligned} p^{k+1}_{2i} &= p^k_i \\ p^{k+1}_{2i+1} &= \frac{-1}{16}p^k_{i-1} + \frac{9}{16}p^k_i + \frac{9}{16}p^k_{i+1} - \frac{1}{16}p^k_{i+2} \end{aligned} \tag{1}$$

When analyzing subdivision schemes, it can often be useful to represent a subdivision mask as a generating function [6]. We define a generating function $s[x]$ for curves to be $s[x] = \sum_{i \epsilon \mathbb{Z}} s_i x^i$ 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 $p^k[x] = \sum p^k_i x^i$. 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 they make it simple to break apart a subdivision scheme into separate passes over the mesh (such as linear subdivision and repeated averaging as

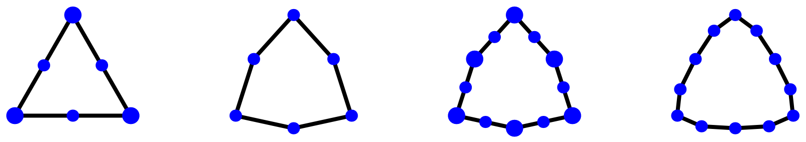

Fig. 2. Subdividing a curve by performing linear subdivision and then differencing. Two rounds of subdivision are shown.

in [8]); just factoring the generating function provides the necessary tool to split a subdivision step into separate passes over the mesh. We define this subdivision method with multiple passes to be 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 allows very large, complex passes to be calculated in a simple manner.

Representing our subdivision mask as a generating function, $s[x]$ for equation (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 $\frac{1}{2}x^{-1} + 1 + \frac{1}{2}x$, 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 1 shows the set of control points after linear subdivision and the weights of the points that result in the same subdivision scheme as on the left-hand side of figure 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 into it the average of the second differences of the vertices on the adjacent edges. Figure 2 shows an example of this process. First, we take the initial curve defined by a triangular shape and perform linear subdivision to produce a new curve. The old vertices from the previous curve are highlighted for clarity. Next, this new curve has the differencing pass applied to it. The resulting curve corresponds to one round of subdivision. We then continue this process to subdivide the curve further.

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 only need to concentrate 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 (1)). Now the rule for all of the vertices after linear subdivision has become uniform in that the differencing pass is applied

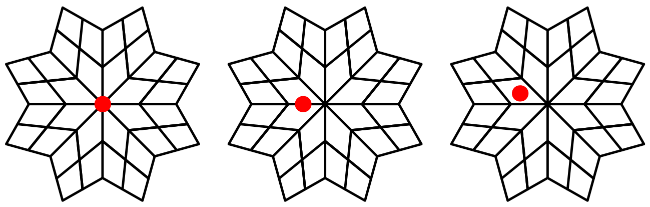

Fig. 3. Different types of points that need to be considered when subdividing the mesh. They are (from left to right) vertex, edge, and face points.

to all of the vertices regardless if 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 0 for the adjacent vertices since they 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, it will become more important with surfaces as there will not only be original vertices and edge vertices, but face vertices as well.

§3. Closed Surface Subdivision

We next generalize the curve subdivision scheme to surfaces by first taking the tensor product of the derived curve scheme's masks. This extension produces 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 deriving a generalization of the masks to extraordinary vertices (valence $\neq 4$) that reproduces the regular case and is smooth in the limit.

Kobbelt [3] 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 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 4.

We define a *curve network* to be a set of vertices and edges that connect those vertices. It is possible to define a curve network on a surface by choosing a subset of vertices and edges on the coarse control polygon to

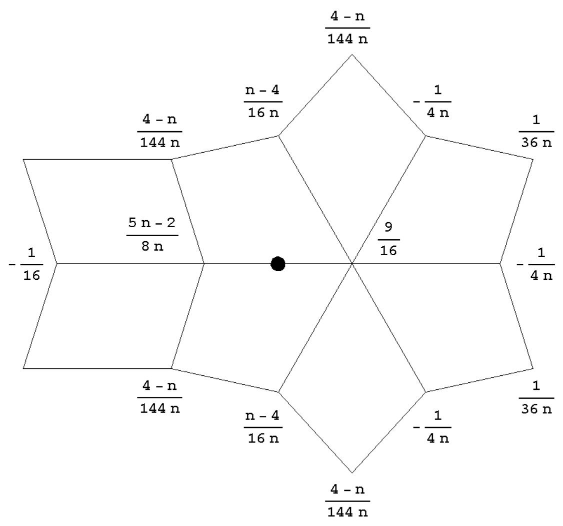

Fig. 4. Kobbelt's rule for positioning an edge point near an extraordinary vertex.

be part of that curve. If subdivision rules are defined for the curve network, then it 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 interpolating 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 4). The subdivision method that we present will have the property that the position of new edge points is only determined 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 onto 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. This property makes the method easier to implement due to the two uniform masks that are applied to the surface.

To extend the difference mask to surfaces we use the tensor product, which can be represented as the product of two generating functions $s[x,y] = s[x] * s[y] = \sum s_i s_j x^i y^j$. These coefficients can easily be viewed as a 2D matrix.

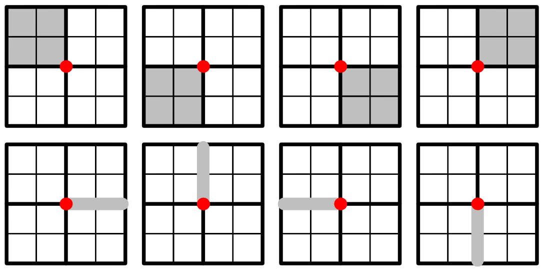

Fig. 5. Positioning of a vertex by adding in face differences and edge differences.

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

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

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 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.

In the curve case, the original vertices were interpolated because the edge differences were necessarily zero since the adjacent vertices were the result of linear subdivision. The same is true here; the edge differences are still zero and the face differences are as well for the same reason. Vertices that are added on the edges of quads during linear subdivision are only affected by the edge differences because the face differences are 0. Face points are the only points that are affected by both the face differences and the edge differences.

3.1 Extraordinary Vertices

Unfortunately, this scheme in its current incarnation does not handle extraordinary vertices. Figure 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

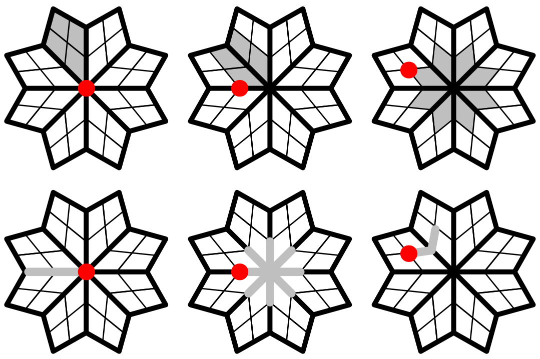

Fig. 6. Face differences and edge differences that need to be computed in the extraordinary vertex case.

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 it should behave when it is centered on an extraordinary vertex (see middle of bottom row).

To calculate the face difference at an arbitrary valence vertex, we process each polygon containing the specified vertex and accumulate the mask $1-x-y+xy$. This process will result in the mask shown in figure 7 (left). This mask is then divided by $4n$ where n is the valence of the vertex.

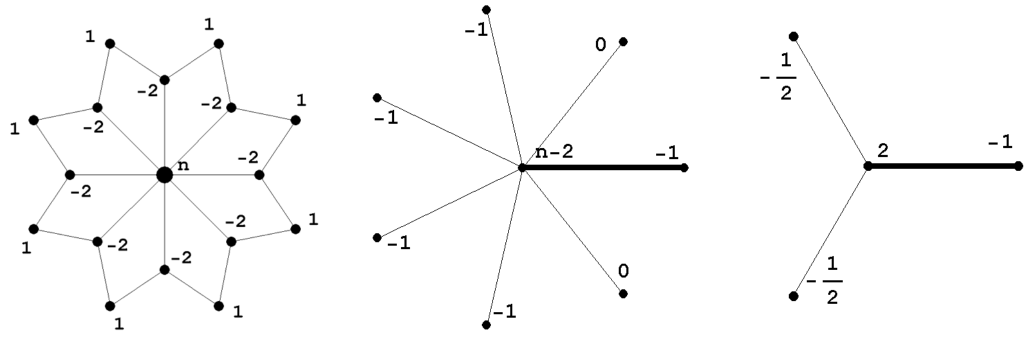

Fig. 7. Face difference at an extraordinary vertex (left). Edge difference rule at an extraordinary vertex across the highlighted edge (middle). Edge difference rule at a valence three vertex (right).

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 7 (middle) 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

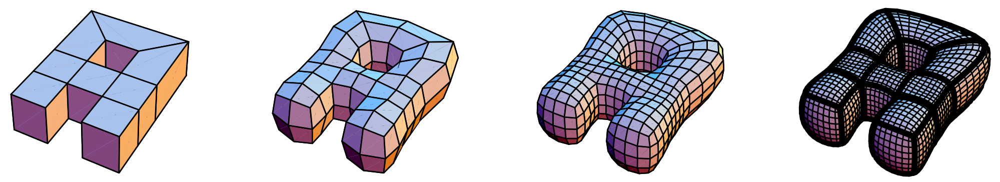

Fig. 8. Subdivision of an "A" containing vertices of valence three to five. Far right also shows the curve network interpolated by the surface.

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

However, there is a problem with the mask shown in figure 7 (middle). 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 7 (middle), we use the mask in figure 7 (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 8 shows an example mesh being 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. The mesh on the far right also shows the curve network defined by the edges of the original mesh that the surface interpolates.

§4. Conclusions

We have derived a C^1 interpolatory subdivision scheme for curves and closed quadrilateral surfaces. After factoring the curve subdivision scheme into linear subdivision and differencing, we extended this method onto surfaces by generalizing the individual passes onto extraordinary valence vertices. This extension revealed that this subdivision scheme could also be thought of as a subdivision scheme for curve networks.

References

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

2. Dyn, N., J. Gregory, and D. Levin, A four point interpolatory subdivision scheme for curve design, Comput. Aided Geom. Design **4** (1987), 257–268.

3. Kobbelt, L., Interpolating subdivision on open quadrilateral nets with arbitrary topology, Computer Graphics Forum **15**(3) (1996), 409–420.

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

5. Stam, J., On subdivision schemes generalizing B-spline surfaces of arbitrary degree, Comput. Aided Geom. Design **18** (2001), 383–396.

6. Warren, J., and H. Weimer, *Subdivision Methods for Geometric Design: A Constructive Approach*, Morgan Kaufmann, San Francisco, 2002.

7. Zorin, D., A method for analysis of C^1-continuity of subdivision surfaces, SIAM Journal of Numerical Analysis **37** (2000), 1677–1708.

8. Zorin, D., and P. Schröder, A unified framework for primal/dual subdivision schemes, Comput. Aided Geom. Design **18**(5) (2001), 429–454.

Scott Schaefer
Department of Computer Science
Rice University
6100 Main St.
Houston, TX 77005
`sschaefe@rice.edu`

Joe Warren
Department of Computer Science
Rice University
6100 Main St.
Houston, TX 77005
`jwarren@rice.edu`