Sharp Features on Multiresolution Subdivision Surfaces

Henning Biermann†, Ioana M. Martin‡, Denis Zorin†, Fausto Bernardini‡

†New York University* ‡IBM T. J. Watson Research Center†

Abstract

In this paper we describe a method for creating sharp features and trim regions on multiresolution subdivision surfaces along a set of user-defined curves. Operations such as engraving, embossing, and trimming are important in many surface modeling applications. Their implementation, however, is non-trivial due to computational, topological, and smoothness constraints that the underlying surface has to satisfy. The novelty of our work lies in the ability to create sharp features anywhere on a surface and in the fact that the resulting representation remains within the multiresolution subdivision framework. Preserving the original representation has the advantage that other operations applicable to multiresolution subdivision surfaces can subsequently be applied to the edited model. We also introduce an extended set of subdivision rules for Catmull-Clark surfaces that allows the creation of creases along diagonals of control mesh faces.

1 Introduction

Interactive editing is of great importance for creating geometric models for a variety of applications, ranging from mechanical design to movie character creation. Often times, modeling begins with an existing object on which the user performs a sequence of editing operations that lead to the desired shape. Of particular interest are small-scale features such as engravings and embossed details that are encountered on many real-life objects.

Traditionally, geometric modeling has relied on non-uniform rational B-splines (NURBS) for surface design. However, NURBS have well-known limitations such as the inability to address arbitrary topology, tedious cross-boundary continuity management, and difficulty representing different resolution levels. In addition, editing operations such as those considered in this paper typically require features to be aligned with iso-parameter lines or patch boundaries, or other complex manipulations in parameter space.

*{biermann,dzorin}@mrl.nyu.edu, www.mrl.nyu.edu

†{ioana,fausto}@us.ibm.com, www.research.ibm.com/vgc

While techniques such as free-form deformations [19], wires [20], and procedural modeling [18] offer alternative ways to edit three-dimensional objects, typically they do not present a unified representation that includes both the original surface and the edits. Thus, in many cases, the resulting representation is not the same as the original, but an extension of it. The main drawback of this approach is that algorithms that have been developed for the original representation are not directly applicable to the result and special cases may have to be considered.

The past few years have seen considerable advances in subdivision theory and many common NURBS operations have been translated into the subdivision setting. Subdivision theory [22], parametric evaluation [21], and applications such as interactive editing [23, 12], trimming [15], boolean operations [2], and geometry compression [11] have contributed to the increasing popularity of multiresolution subdivision surfaces. To date, they have been used in commercial modelers (e.g., Alias/Wavefront's Maya, Pixar's Renderman, Nichimen's Mirai, and Micropace's Lightwave 3D) and are currently making their way through in game engines and hardware implementations. Subdivision algorithms are attractive because of their conceptual simplicity and efficiency with which they can generate smooth surfaces starting from arbitrary meshes. Multiresolution subdivision surfaces offer additional flexibility by allowing modeling of details at different resolution levels and ensuring that fine-scale edits blend naturally with coarse shape deformations.

In this paper, we address the problems of feature placement and feature creation by providing a set of tools that allow fine-scale editing and trimming operations to be applied anywhere on a surface. We use multiresolution subdivision surfaces as our representation and we ensure that this representation is preserved after editing. This gives us the flexibility to integrate our technique with other algorithms developed for multiresolution subdivision surfaces. In our implementation, we use a subdivision scheme for quadrilateral meshes, however a similar algorithm can be developed for triangular meshes. Our contributions include:

1. An algorithm to produce sharp features at arbitrary locations on a piecewise-smooth multiresolution surface

without remeshing the control mesh. The sharp features are created interactively, along curves drawn by the user on the target surface.

2. An extended set of rules for the Catmull-Clark subdivision scheme that allow the creation of creases and boundaries along diagonals of quadrilateral mesh faces.

3. A unified solution to offsetting and trimming operations. Using our technique, a sharp crease having a user-defined profile may be applied along a given curve. Alternatively, the portion of the surface delimited by the curve can be trimmed off, creating a hole in the surface.

The remaining sections of the paper are organized as follows: in section 2 we overview surface modeling methods and we emphasize the main differences between our approach and existing techniques. The core of our method is presented in section 3. In section 4 we present our results and we illustrate applications. Finally, in section 5 we summarize our work and we point out open issues.

2 Background and Related Work

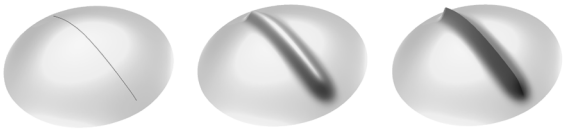


Figure 1. Surface editing. Features are added to an initial surface (left). Smooth and sharp features are fundamentally different: smooth features (center) add bumps to the surface, while sharp features create tangent plane discontinuities (right).

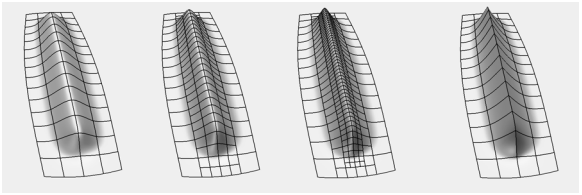


Figure 2. Smooth surface representations do not capture sharp features. Multiresolution Catmull-Clark surfaces can approximate sharp features by adding detail coefficients at finer levels (left three pictures). Instead, we use piecewise-smooth multiresolution surfaces to exactly represent sharp features without any detail coefficients (right).

Subdivision surfaces [1] efficiently represent free-form surfaces of arbitrary topology. A subdivision surface is defined over an initial control mesh and a subdivision scheme is used to recursively refine the control mesh by recomputing vertex positions and inserting new vertices according to certain rules (masks). Recursive subdivision produces a hierarchy of meshes converging to a smooth limit surface. Most objects of interests to geometric design, however, are only piecewise smooth and exhibit sharp creases and corners. To model them using subdivision, special rules are needed to avoid smoothing of sharp details (Figure 1 and 2). Previous work in this area has focused on defining such special rules. Hoppe et al. introduce rules to create sharp features on subdivision surfaces in [9]. The work of DeRose et al. [5] extends Hoppe's approach to achieve creases of controllable sharpness by using subdivision rules parameterized by a sharpness factor. Our work builds upon subdivision schemes for piecewise-smooth surfaces [3] where control mesh vertices and edges are tagged in order to generate singularities, such as creases, darts, and corners. We also draw upon the curve interpolation work of Nasri [17]. In all of these techniques, there is the common requirement that features need to be aligned with the edges of the underlying control mesh. Therefore, the control mesh has to be designed with a particular feature in mind. However, a designer might want to first model an initial shape and apply small scale features in later stages of the design. It is our goal to support this kind of a modeling approach and to allow features to be placed at arbitrary locations on the surface.

Multiresolution subdivision surfaces are a natural extension of subdivision surfaces that accommodates editing of details at different scales, allowing general shape deformations as well as the creation of minute features. Multiresolution, however, does not solve the problem of sharp features as they can only be placed along edges at discrete locations in the mesh hierarchy.

Our technique removes this constraint by allowing sharp features to be created and edited along any user-defined set of curves on the mesh. The main idea is to view subdivision surfaces as parametric surfaces defined over the coarsest-level mesh similar to MAPS [13]. Similar to the approach in [2] we compute the image of a given curve in the parametric domain and we reparameterize the surface to align the parameterization with the curve on some level of the multiresolution hierarchy. Subsequently, we apply special rules to generate non-smooth features.

The closest work to the technique presented in this paper is that of Khodakovsky and Schröder. In [10] they describe a method for interactive creation of feature curves at arbitrary locations on a surface. To create a feature along a curve, a perturbation according to a given profile is applied in the neighborhood of the curve, while maintaining smooth boundary conditions. There are no restrictions on the position of the curve with respect to the underlying surface, however, the representation used is no longer a pure

multiresolution surface. In order to create a sharp feature with this technique, it is necessary to enforce the feature profile at each level of the multiresolution hierarchy. Both the surface and the feature curve are needed to represent the resulting surface. Thus, one cannot directly use techniques developed for subdivision surfaces (i.e., evaluation). In our method, we address the issue of arbitrarily placed sharp features within the multiresolution subdivision setting, thus allowing for greater flexibility in combining feature editing with other existing subdivision tools.

A by-product of our method is the ability to perform trimming of surfaces by simply discarding the portion of surface inside a given curve. Trimming is an important design operation that has been traditionally difficult to perform on parametric surfaces. Our work is complementary to the trimming approach of Litke et al. [15] where quasi-interpolation is used to approximate a trimmed surface with a combined subdivision surface [14]. Similarly, quasi-interpolation may be combined with our approach to obtain trimmed multiresolution surfaces within a specified tolerance.

3 Feature Editing Algorithm

The input to our feature editing algorithm consists of a Catmull-Clark [4, 6] multiresolution subdivision surface, a set of feature curves on the surface, and a user-selected profile to be applied along these curves. The result is a multiresolution subdivision surface with offset or trim features along the given curves.

The basic idea is to model sharp features with piecewise-smooth surfaces. We view feature curves as boundaries between smooth surface patches. Sharp features occur along patch boundaries where patches with distinct tangent planes are joined.

Our multiresolution surface representation enables us to represent sharp features as boundaries or creases by tagging the control mesh edges. In general, the creases generated in this fashion do not coincide with the user specified feature curves. Moreover, it may not be possible to create topologically equivalent curves due to the control mesh topology. Therefore, before we can represent a feature, we need to change the surface parameterization.

Our algorithm proceeds in two steps: *Reparameterization* and *Feature Creation*. We first reparameterize the surface by sliding the control mesh along the surface in order to sample the feature curve with vertices of the mesh. Hence, we are able to approximate the feature curve by edges or face diagonals of the control mesh. In the following subdivision step, we treat these edges and diagonals as creases in the control mesh, and apply piecewise-smooth subdivision rules to obtain a surface with a sharp feature. The surface patches on each side of the feature may be controlled separately. This allows to shape the feature according to a specified profile. Moreover, for trimming we can discard the surface on one side of the feature without changing the surface on the other side.

3.1 Reparameterization

The goal of this step is to align the parameterization of the given surface with a given feature curve. Recall that a multiresolution subdivision surface can be naturally parameterized over the coarsest level control mesh (Figure 3).

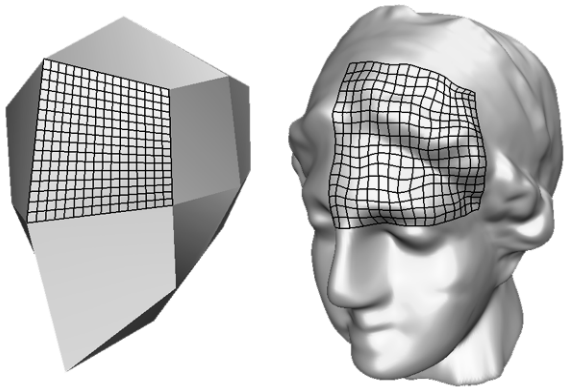


Figure 3. Parametric domain and surface. Multiresolution subdivision surfaces can be parameterized over the coarsest level control mesh (left). The subdivision operator maps vertices of the parametric domain to their image on the surface (right).

Some notation is necessary to describe the reparameterization. Let c denote an input curve defined on the parameter domain X of the surface, $c : [0, 1] \to X$. In general, c traverses the domain X at arbitrary positions. We want to reparameterize the domain X such that c passes through the vertices of X. Therefore, we compute a one-to-one mapping $\Pi : X \to X$ which maps vertices of X to curve points: $\Pi(v_i) = c(t_i)$, for some vertices $\{v_0, v_1, \dots\}$ and curve parameters $\{t_0, t_1, \dots\}$. The mapping Π is built to satisfy the following *approximation property* (AP):

(AP): *the piecewise linear curve* $[v_0, v_1, \dots]$ *has the same topology as* c *and either follows along mesh edges or crosses mesh faces diagonally.*

The reparameterization algorithm proceeds iteratively, alternating *Snapping* and *Refinement* steps. The snapping step moves mesh vertices onto the curve if they are sufficiently close. In the refinement step we simply subdivide the parameterization linearly. The algorithm terminates if the sequence of vertices $\{v_0, v_1, \cdots\}$ along c satisfies the approximation property (AP). Property (AP) is guaranteed to be satisfied after a finite number of steps for piecewise-linear curves c. After the reparameterization, the surface is *Resampled* to reflect the new parameterization.

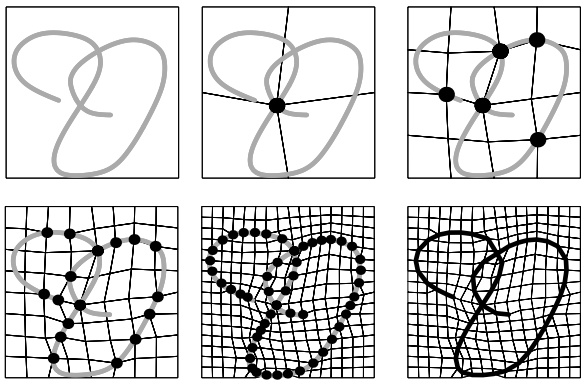


Figure 4. Reparameterization matching a feature curve. The quad is recursively split and vertices are snapped to the curve. After four subdivision steps, the curve is approximated by a sequence of mesh vertices: The approximated curve follows edges or passes through quads diagonally.

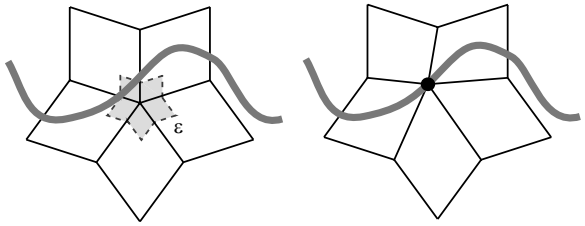


Figure 5. Snapping step. Vertices are snapped to closest curve vertex if the distance is less than a certain ε. Note that distances are measured in parameter space, where each face corresponds the unit square. A single snapping step reparameterizes the neighborhood of the snapped vertex.

Snapping. This step moves vertices onto the curve c if they are sufficiently close to it. First, the algorithm traverses the mesh along the curve on a given subdivision level. For all vertices of the traversed faces we compute the closest points on the curve. Distances are measured by parameterizing each quad intersected by c over the unit square and by computing distances to the curve in this parameter space. This approach presents an advantage over computing geometric distances in that it does not undersample small quads. Vertices v within a certain distance ε to the curve are snapped to the corresponding closest points $c(t)$ on the curve (see Figures 5 and 4). We assign $\Pi(v) := c(t)$.

The parameter ε controls the distortion of the reparameterization: small values keep vertices from moving too far, but require more snapping steps. In all cases, ε is less than 0.5 to ensure that the snapping region around each vertex is disjoint from the snapping regions of its neighbors. In our examples, we obtained good results with $\varepsilon = 0.3$.

In some cases, it may be necessary to align specific points in the parameter domain with mesh vertices. This is the case, for instance, where several curves intersect in a corner. Due to the chosen surface representation, we need a mesh vertex at the corner that connects the separate curve branches. Such constraints are enforced during snapping: constrained curve points have higher priority than unconstrained points. For a given vertex, the algorithm first tries to snap to the unmatched constrained vertices. If no snapping is possible, unconstrained points are considered as snap targets.

Refinement. The parameterization Π is piecewise linearly subdivided to increase the resolution and allow future snapping. The subdivision is done similarly to [13]. As in [13], local charts are used to refine the parameterization Π where neighborhoods are mapped to different faces of the coarsest level control mesh.

Resampling. After reparameterization, we resample the surface at the new parameter positions. Intuitively, this moves the control mesh on the surface and places mesh vertices on the feature curve. With the notation from above, we iterate over the vertices of the finest level control mesh. For every vertex v of X, we assign a new position by evaluating the input surface at parameter position $X(v)$. Finally, we apply multiresolution analysis to obtain a multiresolution representation with detail coefficients.

3.2 Feature Creation

The reparameterization step approximates the feature curves as chains of mesh edges or diagonals and mesh diagonals. Our idea is to view these curves as crease or boundary curves in the control mesh. In order to create surfaces with sharp features, we apply crease subdivision rules along the feature curves. The shape of the feature can be controlled by offsetting mesh vertices in the feature neighborhood according to a user given profile.

The profiles shown in Figures 14 have been created by changing the positions of mesh vertices that are close to a feature curve. Vertices are displaced normal to the surface and the displacement is determined by their distance to the curve. Note that for closed curves different shape profiles may be used for displacing points in the interior and points on the outside.

Trimming can also be performed by simply interpreting a feature curve as a boundary of the mesh. The feature curve cuts the control mesh into separate pieces. Each piece can be subdivided separately and processed further. The resulting surfaces are independent from each other, moreover, their boundary curves are cubic B-splines.

3.3 Subdivision Rules for Sharp Features

In this section, we explain how to subdivide control meshes with feature curves (Figure 6). As in the standard

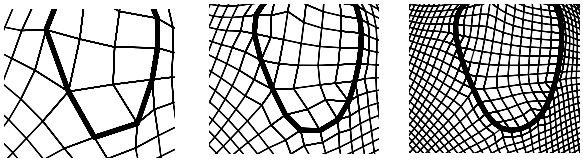


Figure 6. Subdivision of a control mesh with a feature curve. The feature curve is a sequence of mesh edges or mesh diagonals acting as a boundary in the mesh. The resulting surface consists of two smooth patches joined along a cubic B-spline boundary. Our subdivision scheme extends piecewise-smooth subdivision with rules for diagonally split faces.

Catmull-Clark subdivision, we iterate over the mesh on a given level and we compute positions of the vertices on the next level by refining the positions of existing vertices (vertex points) and by inserting new vertices on edges (edge points) and in the faces centers (face points). We apply special rules in the vicinity of the curves and standard rules everywhere else. Our rules extend the piecewise-smooth subdivision of [3]. The idea is to treat the feature curve as a crease and to refine the mesh on each side of the curve independently to create a tangent-plane discontinuity. As feature curves may pass through quads diagonally, we introduce new rules, that account for such situations.

We use vertex tags to identify the subdivision rules to be applied when traversing the mesh. Initially, we tag all vertices traversed by the feature curves as crease vertices. Additionally, we mark the faces that are cut diagonally by the curve.

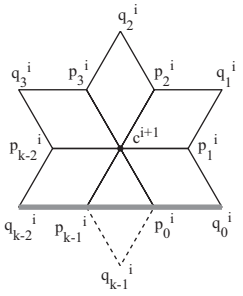


Figure 7. Special vertex rule in the neighborhood of a curve that passes through some of the quads diagonally (gray line). Point q^i_{k-1} is on the opposite side of the curve from c^i and it is not used in the computation of the refined position c^{i+1}. The reflection of c^i across the diagonal (p^i_{k-1}, p^i_0) is used instead.

Vertex points. A refined control point position corresponding to an untagged vertex is computed as a weighted average of control points in its neighborhood. For a vertex c with valence k (i.e., with k adjacent polygons), its new position c^{i+1} is a linear combination of its old position weighted by $(1-\beta-\gamma)$ and the positions of the vertices in its 1-ring each weighted by β/k if situated on an edge incident to c or by γ/k otherwise. We use coefficients $\beta = 3/(2k)$ and $\gamma = 1/(4k)$.

A special situation occurs when some of the quads in the 1-ring of c are split by a curve (see Figure 7). The previous rule is modified to ignore vertices that are not on the same side. We use $q' := p^i_0 + p^i_{k-1} - c$.

$$c^{i+1} = (1-\beta-\gamma)c^i + \frac{1}{k}\left(\beta p^i_{k-1} + \gamma q' + \sum_{j=0}^{k-2} \beta p^i_j + \gamma q^i_j\right)$$

A crease vertex is refined as the average of its old position with weight $3/4$ and the two adjacent crease vertices with weights equal to $1/8$.

Corner vertices (i.e., where two or more creases meet) require additional rules depending on the neighboring topology, similar to our discussion of creases. For the sake of brevity, we do not include them in this paper. Darts (i.e., smooth transitions of a crease into a surface) require no special consideration, as the standard rules are directly applicable at such points.

Face points. For faces that are not split diagonally by crease curves, we insert a point in the centroid of each face. For faces with a diagonal on the curve, the center point is computed as the average of the two diagonal endpoints on the crease.

Edge points. On an edge with both endpoints tagged, we insert a new vertex as the average of the endpoints. When both endpoints are untagged, the standard edge mask applies. The remaining case is that of an edge with one vertex tagged and the other untagged. A curve passing through a mesh vertex partitions the mesh in the neighborhood of that vertex into two *sectors*. We select the rule to be applied to an edge point adjacent to a tagged vertex c depending on the topology of the sectors around the tagged vertex. We distinguish three types of sectors:

1. *Consisting of quads only:* in this case the curve follows two edges incident to c and we can apply the crease rules described in [3]. This case is illustrated in Figure 8(a). We choose $\theta_k = \pi/k$, where k is the number of polygons adjacent to c in the sector considered, and apply an edge rule which is parameterized by $\gamma = 3/8 - 1/4\cos\theta_k$. The new edgepoints p^{i+1}_j $(j = 1, \cdots, k-1)$ are computed as

$$p^{i+1}_j = (\frac{3}{4} - \gamma)c^i + \gamma p^i_j + \frac{1}{16}(p^i_{j-1} + p^i_{j+1} + q^i_{j-1} + q^i_j).$$

2. *Beginning with a triangle (i.e., a split quad) and ending with a quad or vice versa:* in this case the curve

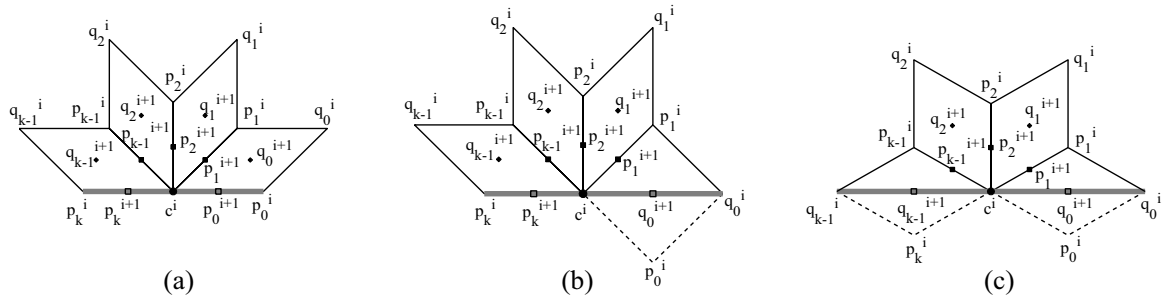


Figure 8. Special rules for edge points on edges with one endpoint on a curve (gray line). (a) Sector delimited by the curve consists of quads only: in this case the rules from [3]. (b) Sector begins with a split quad and ends with a full quad. In this case only p_0^i is on the opposite side of the curve and the only rule that needs to be modified is that for p_1^{i+1}. The reflection of c^i across the edge (p_1^i, q_0^i) is used. (c) Sector begins and ends with quads that are split by the curve. Points p_k^i and p_0^i are on the opposite side of the curve. The rules that would normally take into account these points to compute p_1^{i+1} and p_{k-1}^{i+1} are modified to use the reflections of the points p_1^i and p_k^i across the curve instead.

passes through the vertex following a mesh edge and a mesh diagonal. This case is illustrated in Figure 8(b). We use θ_k as above and apply the same edge rule to compute $p_2^{i+1} \cdots, p_{k-1}^{i+1}$. The edge point p_1^{i+1} between the triangle and the first quad is obtained as

$$p_1{}^{i+1} = (\frac{11}{16} - \gamma)c^i + (\gamma + \frac{1}{16})p_1^i + \frac{1}{16}(p_2^i + 2q_0^i + q_1^i).$$

3. *Beginning and ending with triangles:* in this case the curve follows two diagonals incident to c (Figure 8(c)). We choose $\theta_k = k - 1$ and apply the edgerule of the first case to find $p_2^{i+1}, \ldots p_{k-2}^{i+1}$. The edge points p_1^{i+1} between first triangles and quads is computed as

$$p_1{}^{i+1} = (\frac{13}{16} - \gamma)c^i + (\gamma - \frac{1}{16})p_1^i + \frac{1}{16}(p_2^i + 2q_0^i + q_1^i).$$

The rule for p_{k-1}^{i+1} is symmetric to this. In the special case of $k = 2$, we use $p_1^{i+1} = 1/4c^i + 1/2p_1^i + 1/8(q_0^i + q_1^i)$.

Tangents and normals. Our subdivision scheme has well-defined limit and tangent properties. We can efficiently evaluate limit positions of vertices and tangent directions by applying specific masks [8]. These masks correspond to the left eigenvectors of the subdivision matrix used at a given vertex. The masks are listed in appendix A.

3.4 Discussion of the Subdivision Rules

In this section, we briefly discuss some properties of the previous subdivision rules and we motivate our choice of the special rules for crease neighborhoods.

As we use cubic B-Spline subdivision along the features, the resulting curves are B-Splines defined only in terms of control points along the curves. Moreover, the surfaces on either side of the feature do not depend on each other. This is a consequence of our rules, as no stencil uses vertices from the opposite side of a feature.

Our rules have an easy geometric interpretation (Figure 8), but some subdivision theory is needed to understand the rules in more detail. We follow the usual eigen-analysis approach [6] to understand the asymptotic behavior of the subdivision operation. Consider a neighborhood of a crease vertex c as shown in Figure 8(a). Iterated subdivision contracts the neighborhoods to a single point. We want to ensure a well-shaped limit configuration and design rules which preserve a specifically chosen configuration. Technically speaking, we design rules that have certain desired subdominant eigenvectors (see Appendix A). We use geometric reasoning to reduce the cases of neighborhoods with triangles to the case of neighborhoods without. The reflections previously mentioned are chosen to map the desired eigenvectors to the eigenvectors of the no-triangle case (Figure 8(a)). Thus, we can design subdivision rules as follows: (i) apply reflection to complete triangles, (ii) subdivide using the usual no-triangle rules and (iii) discard the unnecessary points. The subdivision scheme defined in this way has the desired eigenvectors.

For a complete analysis a larger neighborhood and the corresponding subdivision matrix need to be analyzed. This is beyond the scope of this paper, and instead, we visualize here only the asymptotic behavior of the subdivision rules with illustrations of the characteristic maps (Figure 9).

4 Applications and Results

Figure 13 illustrates the steps of the algorithm for a trimming sequence. The creation of sharp features and trim regions is illustrated in figures Figures 10 and 11. Note the arbitrary position of the curves with respect to the underlying meshes. The models were created interactively on a Pentium III workstation.

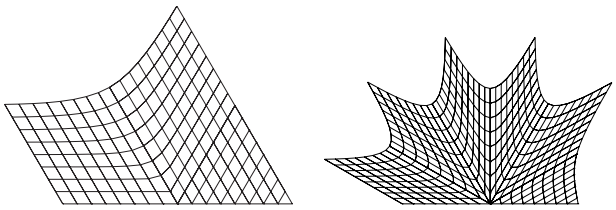


Figure 9. Characteristic map for crease vertex neighborhoods with a single triangle. We show maps for valence 3 (left) and valence 6 (right). The maps show the behavior near the curve vertex. Note that the maps are smooth and one-to-one.

Figure 14 illustrates the creation of offset features along a curve according to various user-specified profiles. The profiles are based on distance to the curve. In general, computing distances on surfaces is a difficult problem [16]. In our examples (i.e., Figure 14), we measure the distance in space, which is a reasonable approximation for small distances on surfaces with low curvature. Alternatively, one could use geodesic distances on the surface [10].

Also, we use our algorithm to create features on multiresolution surfaces (Figure 12(a)). Feature curves with intersections are illustrated in Figures 13 and 12(b).

Finally, we demonstrate how our trimming approach approximates the result of a precise trimming operation (Figure 15). It is the nature of our technique that the resulting surface is different from the input surface (even for a flat feature profile). The differences are due to resampling and the use of piecewise-smooth base functions in the feature neighborhood. Also, the specified feature curves are resampled only at vertices of the control mesh. However, we can control the approximation by resampling surface and feature curves at different levels in the hierarchy. In general, resampling on a finer level reduces the error, but is computationally more expensive. In our implementation the user controls the level on which the resampling takes place.

5 Conclusions and Future Work

In this paper we present an efficient method for creating sharp features along an arbitrarily positioned set of curves on a Catmull-Clark multiresolution subdivision surface. We view our surface as a parametric surface defined over the initial control mesh and we change the parameterization to align it with the pre-image of the feature curves in the parameter domain. The result is a surface represented in the same way as the input surface with the curves passing through mesh edges or face diagonals. This property allows us to apply special subdivision rules along the curve to create sharp profiles. Another application of this algorithm is trimming, which can be achieved simply by discarding the portion of the mesh situated inside the trim curve.

Our algorithm takes as input arbitrarily shaped curves, with or without self-intersections, as well as multiple intersecting curves. The number of curves intersecting at a point, however, is limited by the number of connections available between a vertex and its neighbors along edges and diagonals (eight in the regular case).

Multiresolution surface representations that allow for topology changes within the hierarchy [7] could be used to resolve this restriction in future research. Other research might combine quasi-interpolation or surface fitting with our trimming approach and study the approximation along the lines of [15]. Also, we are interested to see whether the special diagonal subdivision rules are useful in a general geometric modeling context. For practical applications, it might be useful to work out expressions for direct evaluation.

A Eigenvectors

We list the right and left eigenvectors corresponding to the special subdivision rules previously described. We denote the dominant left eigenvector by l^0 and the left subdominant eigenvectors by l^1 and l^2, respectively. We denote the dominant right eigenvector by x^0 and we use x^1 and x^2 for the right subdominant eigenvectors. Recall that the eigenvector coefficients are applied to the control points of a polygon ring/fan. The eigenvector l^0 is used to compute the limit position of a point, whereas l^1 and l^2 are necessary for computing the tangents. The *crease degree* is the number of polygons adjacent to a crease or corner vertex with respect to a specific sector. We use the following notation: k denotes the crease degree of a crease vertex, the subscript c denotes the coefficient corresponding to the center vertex, we mark edgepoint coefficients with the subscript p, and facepoint coefficients with q. The dominant right eigenvector x^0 is the vector consisting of ones. Unlisted coefficient values are null.

• Sector consisting of quads only. Let $\theta_k = \pi/k$.

$$l^0_c = 2/3,\ l^0_{p0} = l^0_{pk} = 1/6$$

For $k = 1$,

$$\begin{array}{llll} x^1_c = 1/18, & x^1_{p0} = -2/18, & x^1_{p1} = -2/18, & x^1_{q0} = -5/18 \\ x^2_c = 0, & x^2_{p0} = -1/2, & x^2_{p1} = 1/2, & x^2_{q0} = 0 \\ l^1_c = 6, & l^1_{p0} = -3, & l^1_{p1} = -3, & l^1_{q0} = 0 \\ l^2_c = 0, & l^2_{p0} = -1, & l^2_{p1} = 1, & l^2_{q0} = 0 \end{array}$$

otherwise $l^2_c = x^1_c = x^2_c = 0$ and

$$x^1_{pi} = \cos i\theta_k, \quad x^2_{pi} = \sin i\theta_k, \quad i = 0, \cdots, k$$

$$x^1_{qi} = \cos i\theta_k + \cos(i+1)\theta_k, \quad i = 0, \cdots, k-1$$

$$x^2_{qi} = \sin i\theta_k + \sin(i+1)\theta_k, \quad i = 0, \cdots, k-1$$

$$l^2_{p0} = 1/2,\ l^2_{pk} = -1/2$$

$$\alpha = \frac{\cos\theta_k + 1}{k \sin\theta_k (3 + \cos\theta_k)}$$

$$l^1_c = 4\alpha(\cos\theta_k - 1), \quad l^1_{p0} = l^1_{pk} = -\alpha(1 + 2\cos\theta_k)$$

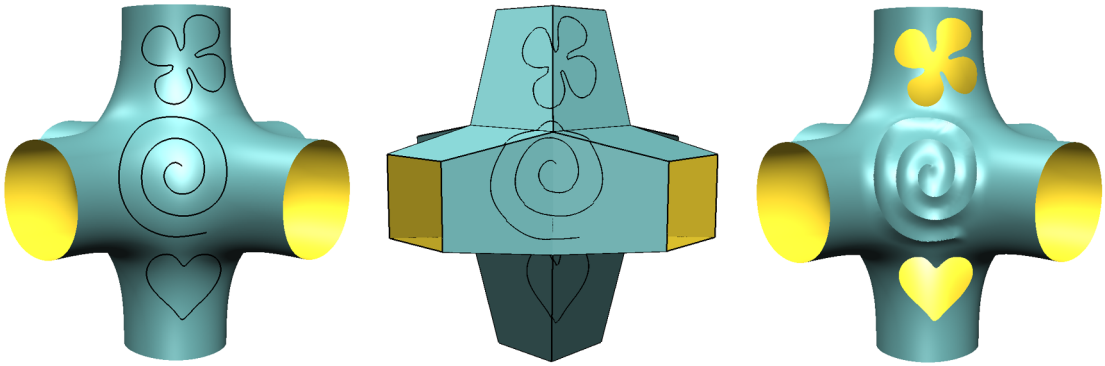


Figure 10. Surface editing. Left: surface with user specified feature curves. Closed curves are trim curves, the open curve indicates an offset feature. Center: image of the feature curves in the parametric domain. Right: resulting surface with sharp features and trimmed regions. The displacement of the points along the offset curve is a quadratic function of the distance to the curve.

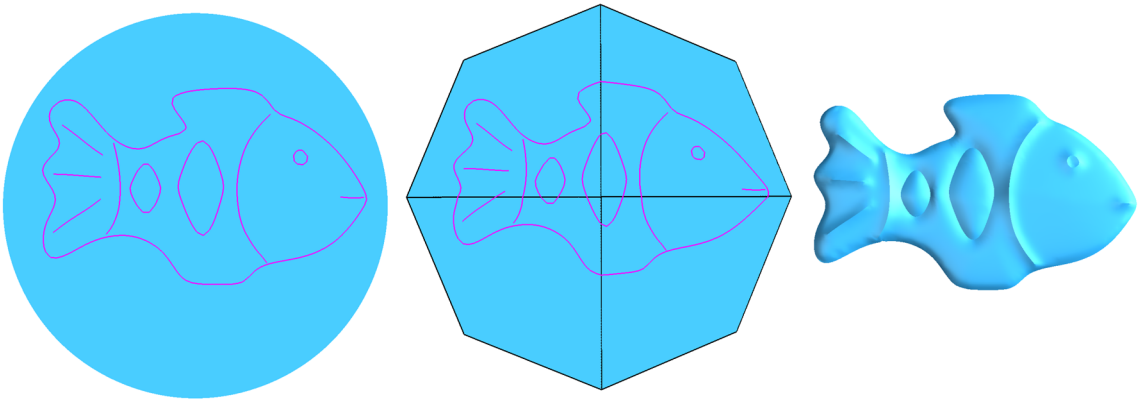
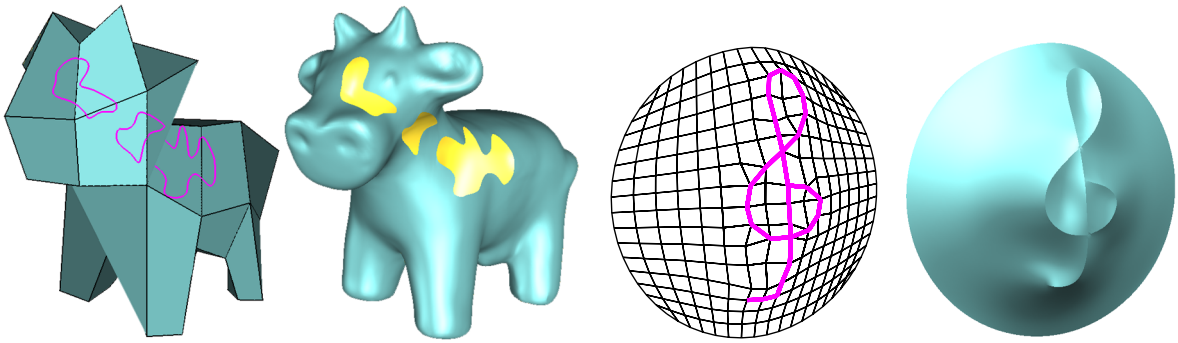


Figure 11. Surface shaping. A fish pin is cut from a disk using a trim curve along its outer contour and shaped with offset curves in the interior.



(a) (b)

Figure 12. Left image pair: trim features on multiresolution surfaces. Right image pair: offset features with intersections. The control mesh on the right shows how reparameterization aligns the feature with edges and diagonals. Mesh vertices are placed at curve intersections.

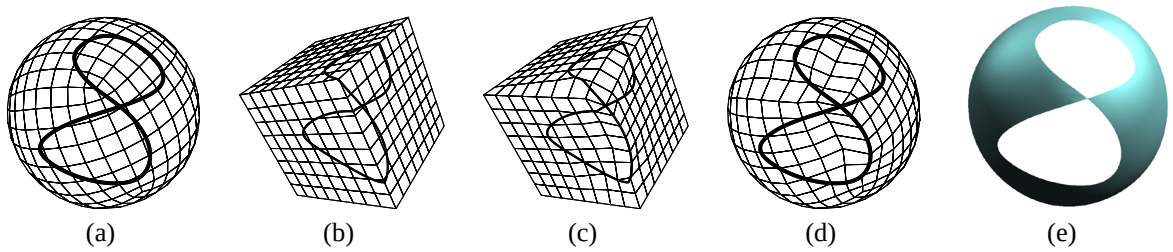


Figure 13. Steps of the algorithm. (a) Wireframe rendering of a surface with feature curve. (b) The surface is parameterized over a cube. (c) Reparameterization aligns mesh edges with the feature. (d) Resampling with respect to new parameterization. (e) Trimming by discarding a piece of the control mesh and subsequent subdivision.

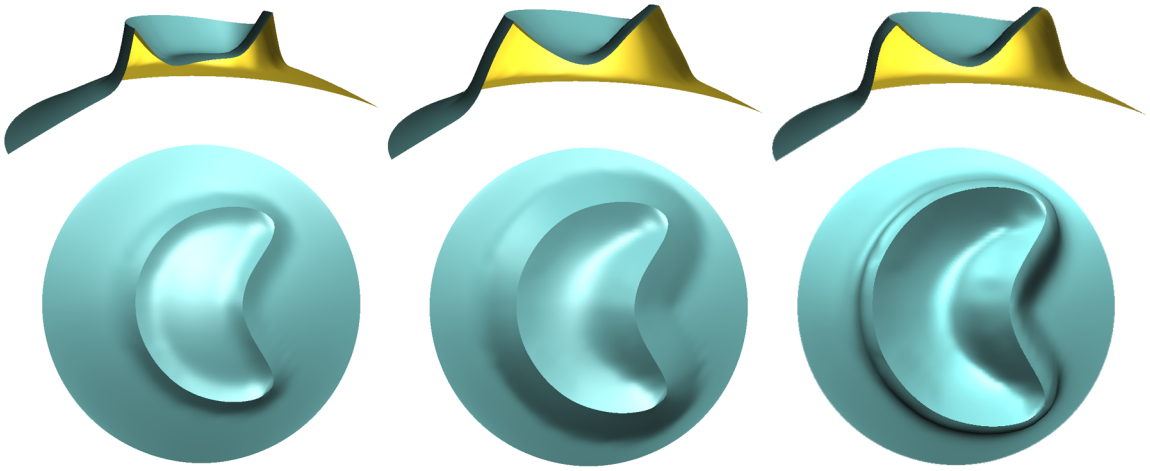


Figure 14. Different offset profiles for a feature curve. In all three cases, the interior profile is a quadratic function of the distance to the curve. Left: linear exterior profile. Middle: linear exterior profile; the size of the neighborhood altered is doubled with respect to the previous image. Right: Gaussian exterior profile.

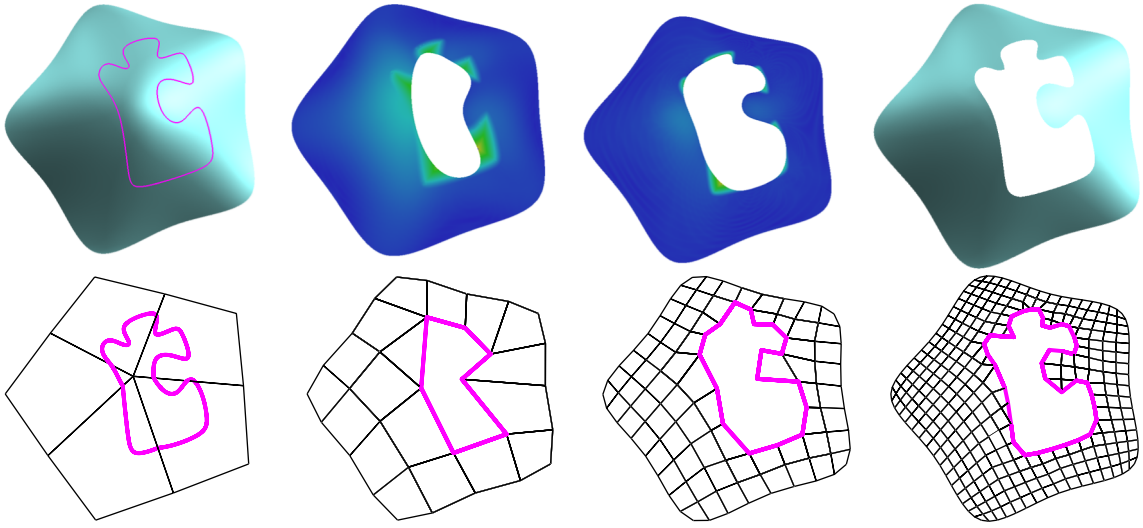


Figure 15. Trim operations are applied on different levels of the hierarchy. Top row: distance between the resulting surface and an analytically trimmed surface. Bottom row: corresponding control meshes. The largest error occurs where the surface boundary does not capture the intended trim curve. The error of surface obtained by trimming on level 4 is less than 1% (of the length of a base mesh edge).

$$l^1_{pi} = \frac{4\sin i\theta_k}{(3+\cos\theta_k)k}, i = 0, \cdots, k-1$$

$$l^1_{qi} = \frac{(\sin i\theta_k + \sin(i+1)\theta_k)}{(3+\cos\theta_k)k}, \quad i = 0, \cdots, k-1$$

• Sector beginning and ending with triangles. Let $\theta_k = \pi/k$.

$$l^0_c = 2/3,\ l^0_{pk} = l^0_{q0} = 1/6,\ l^1_{pk} = 1/2,\ l^1_{q0} = -1/2$$

$$x^1_c = x^2_c = 0$$

$$x^1_{pi} = \cos i\theta_k - \theta_k/2, x^2_{pi} = \sin i\theta_k - \theta_k/2, i = 1, \cdots, k-1$$

$$x^1_{qi} = \cos i\theta_k - \theta_k/2 + \cos(i+1)\theta_k - \theta_k/2, i = 0, \cdots, k-1$$

$$x^2_{qi} = \sin i\theta_k - \theta_k/2 + \sin(i+1)\theta_k - \theta_k/2, i = 0, \cdots, k-1$$

$$l^2_c = -4\sin(\theta_k/2)$$

$$l^2_{p_k} = l^2_{q_0} = \frac{3(\sin^2(\theta_k/2) - 1)}{\sin(\theta_k/2)}$$

$$l^2_{p_i} = 4\sin(i\theta_k - \theta_k/2), i = 1, \cdots, k-1$$

$$l^2_{q_i} = \sin(i\theta_k - \theta_k/2) + \sin((i+1)\theta_k - \theta_k/2), i = 1, \cdots, k-1$$

• Sector beginning with triangle, ending with quad.
Let $\theta_k = \pi/(k-1)$.

$$l^0_c = 2/3,\ l^0_{pk} = l^0_{q0} = 1/6,\ l^1_{pk} = -1/2,\ l^1_{q0} = 1/2$$

$$x^1_c = x^2_c = 0,\ x^1_{q0} = 1,\ x^2_{q0} = 0$$

$$x^1_{pi} = \cos i\theta_k, \quad x^2_{pi} = \sin i\theta_k, \quad i = 1, \cdots, k$$

$$x^1_{qi} = \cos i\theta_k + \cos(i+1)\theta_k, \quad i = 1, \cdots, k-1$$

$$x^2_{qi} = \sin i\theta_k + \sin(i+1)\theta_k, \quad i = 1, \cdots, k-1$$

$$\alpha = \frac{1}{\cos^2\theta_k + k\cos\theta_k + 3k - 1}$$

$$l^2_c = -4\sin\theta_k$$

$$l^2_{p_k} = \alpha\frac{\sin\theta_k(2 + 5\cos\theta_k - \cos^2\theta_k)}{2(\cos\theta_k - 1)}$$

$$l^2_{q_0} = \alpha\frac{\sin\theta_k\cos\theta_k(5 + \cos\theta_k)}{2(\cos\theta_k - 1)}$$

$$l^2_{p_i} = 4\alpha\sin i\theta_k,\ l^2_{q_i} = \alpha\sin i\theta_k + \sin(i+1)\theta_k, i = 1, \cdots, k-1$$

References

[1] Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes.

[2] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form surfaces. *Proceedings of SIGGRAPH 2001*, August 2001.

[3] H. Biermann, A. Levin, and D. Zorin. Piecewise-smooth subdivision surfaces with normal control. *Proceedings of SIGGRAPH 2000*, pages 113–120, July 2000.

[4] Ed Catmull and James Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *CAD*, 10(6):350–355, 1978.

[5] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. *Proceedings of SIGGRAPH 98*, pages 85–94, July 1998.

[6] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *CAD*, 10(6):356–360, 1978.

[7] I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. A degree estimate for polynomial subdivision surface of higher regularity. Technical report, California Institute of Technology, Pasadena, CA 91125, 2001. preprint.

[8] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. *Proceedings of SIGGRAPH 93*, pages 35–44, August 1993.

[9] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of SIGGRAPH 94*, pages 295–302, July 1994.

[10] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. In *Proceedings of ACM Solid Modeling*, 1999.

[11] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. *Proceedings of SIGGRAPH 2000*, pages 271–278, July 2000.

[12] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998.

[13] A. W. F. Lee, W. Sweldens, P. Schröder, Lawrence Cowsar, and David Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998.

[14] A. Levin. Interpolating nets of curves by smooth subdivision surfaces. *Proceedings of SIGGRAPH 99*, pages 57–64, August 1999.

[15] N. Litke, A. Levin, and P. Schröder. Trimming for subdivision surfaces. *CAGD*, To appear.

[16] J.S.B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science, 2000.

[17] A. H. Nasri. Surface interpolation on irregular networks with normal conditions. *CAGD*, 8:89–96, 1991.

[18] K. Perlin and L. Velho. Live paint: Painting with procedural multiscale textures. *Proceedings of SIGGRAPH 95*, pages 153–160, August 1995.

[19] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[20] K. Singh and E. Fiume. Wires: A geometric deformation technique. *Proceedings of SIGGRAPH 98*, pages 405–414, July 1998.

[21] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. *Proceedings of SIGGRAPH 98*, pages 395–404, July 1998.

[22] D. Zorin. *Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, Pasadena, 1997.

[23] D. Zorin, P Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997.