Fine Level Feature Editing for Subdivision Surfaces

Andrei Khodakovsky Peter Schröder

Multi-Res Modeling Group
Caltech

Abstract

In many industrial design modeling scenarios the designer wishes to edit small feature lines—such as variable width and height creases—on otherwise smooth surface patches. When the path of such a feature does not align with an iso-parameter line or crosses patch boundaries it becomes increasingly difficult to maintain good editing semantics of the underlying surface. In this paper we describe an algorithm and implementation allowing the interactive creation and manipulation of fine scale feature curves on subdivision surfaces. In particular, our approach addresses the problem of defining the path of such feature curves independent of the location of surface iso-parameter lines and global patch boundaries. The feature lines are modeled as swept displacement curves with variable profiles, providing a rich toolbox of shapes. Furthermore, the hierarchical editing semantics of subdivision surface based representations carry through to our extended setting, ensuring "good" behavior of the feature lines under coarse scale surface edits.

CR Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation - *Display Algorithms, Viewing Algorithms*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling - *Curve, Surface, Solid and Object Representations, Hierarchy and Geometric Transformations, Object Hierarchies*; G.1.2 [**Numerical Analysis**]: Approximation - *approximation of surfaces and contours, wavelets and fractals*

Additional Keywords: multiresolution editing, hierarchical modeling, subdivision, feature modeling, surface deformation.

1 Introduction

Subdivision methods such as Catmull-Clark [2], Doo-Sabin [8], or Loop [17] (among others) elegantly address some of the difficulties of traditional spline based model settings. Subdivision methods can model arbitrary topology surfaces in a globally smooth fashion while their simple data structures and efficient computational kernels support highly scalable algorithms, such as level-of-detail display and compression, for example. For an overview of basic theory and applications the reader is referred to [24] and the references therein.

These attractive properties of subdivision have led to the development of hierarchical modeling algorithms—and their multiresolution extensions—with considerable flexibility and speed [31].

Figure 1: *Example of a complex shape involving an irregular topology smooth surface with multiple feature curves overlaid on it (Design study for bicycle helmet; Khrysaundt Koenig).*

The maturity of these algorithms has reached a stage where subdivision is being employed in large scale movie production [7] and available in a number of commercial products (Geometique, Radical Entertainment, 3D Studio Max, SurfReyes, Lightwave, etc.).

While the ideas behind subdivision for surfaces are more than 20 years old, important theoretical progress has only recently been achieved (examples include [23] and [32]). Broader interest in these methods is still relatively new and many specific algorithms, often having analogues in the more traditional NURBS setting, remain yet to be worked out in the subdivision setting. Examples of important practical results reported only recently include moment computations [20], exact evaluation at arbitrary parameter locations [29], fast ray tracing [14], variational modeling [13], and arbitrary boundary curve interpolation [16], among others.

For modeling applications the most popular subdivision methods are based on generalizations of traditional spline patch methods. Catmull-Clark [2] generalizes bi-cubic patches, while Loop [17] generalizes quartic box-splines. The resulting surfaces are still made up of individual patches, but global continuity conditions are "compiled" into the knot insertion rules and need not be separately enforced. Note that subdivision surfaces reduce to ordinary spline surfaces (bi-cubic, respectively quartic box-spline) in case the control mesh has regular topology.[1] Aside from their ability to deal well with arbitrary topology control meshes the principal advantage of subdivision flows from a change in the overall approach to the creation of surfaces. Instead of treating them as *analytical* objects with evaluation functions—which subdivision surfaces still possess—the surface is treated as a *procedural* object which is approximated to arbitrary precision through repeated subdivision, or knot insertion. This opens up a wealth of optimizations allowing even very complicated geometry to be managed interactively on low end hardware [31], a critical ingredient in providing interactive modeling environments. Through the connection with classical multiresolution analysis and wavelets it becomes possible to describe hierarchical geometry in a rigorous fashion and enable hierarchical editing (such as first proposed by Forsey and Bartels [9])

[1] Regular is defined as all interior control vertices having valence four (Catmull-Clark) and six (Loop) respectively.

as well as many other practically relevant algorithms such as compression [18] and progressive transmission [3], for example.

An important modeling capability concerns the addition of small scale, curve-like features such as grooves or raised creases to a patch based surface. Such tools are well known and appreciated in the traditional NURBS framework [21]. We extend these capabilities to a subdivision based [31] multiresolution setting [9] and in particular allow the user to draw the desired curve directly onto the surface in world space [4] avoiding distortion issues associated with feature curves defined in parameter space. Doing so in a hierarchical modeling framework ensures that features added at a fine scale respond gracefully to edits performed at coarse scales without loosing their smoothness and overall shape.

Figure 1 shows an example of a complex shape with multiple feature curves overlayed on an irregular topology patch network, created with our interactive modeling system.

1.1 Related Work

There are many possible ways to add the features we desire to a given patch based surface. For example, free-form deformations [19, 26, 6, 12, 5, 28] are a popular way to modify a given surface shape. Free-form deformations achieve their effect through a suitable deformation of the embedding space, typically using a force field function. The advantage of these methods is that they are, in principle, applicable to any surface modeling setting. However, it is generally difficult to model non-smooth features, such as sharp creases, in this way. We chose direct, parameterized displacement in a local coordinate frame induced by the partial derivatives of the surface. This greatly simplifies precise control over the detailed shape of the feature lines. The modified surfaces maintain their globally smooth parameterizations which is essential for many other computations (e.g., raytracing or moment evaluation).

One possible way to describe the displacements we seek would be through the use of a 1D curve in parameter space giving, for example, a scalar displacement value along the local surface normal and a region of influence [21]. However achieving a desired shape on the surface by specifying an appropriate curve in parameter space involves an inverse problem, accounting for the distortion induced by the coordinate functions. This is expensive and particularly difficult to manage across patch boundaries where canonical parameterizations are typically non-smooth. Of course this observation applies not only to subdivision but also to classical NURBS approaches. Instead we allow the user to draw onto the surface directly to describe the desired curve in a manner similar to world space surface pasting approaches [4].

Earlier work incorporating features such as creases and variable radius fillets into the subdivision framework did so through modified subdivision rules [11, 25, 7, 27]. However, these approaches limit the location of feature curves to global patch boundaries and constrain the possible cross-section profiles. Instead we wish to allow the user to draw the desired feature curve *anywhere* on the surface and admit a richer set of varying profiles.

Since we are in a multiresolution setting it is most natural to describe the effect of the feature curve on the shape of the surface by going to the appropriate scale in the subdivision induced approximation hierarchy and achieve the desired effect through addition of local details. In this sense our approach is closely related to hierarchical modeling methods such as H-splines [9] and surface wavelets [18].

Before going into more details on our approach to curve based feature editing we briefly recall some facts about subdivision and its multiresolution extensions.

1.2 Subdivision Surfaces

There are many different subdivision techniques for surfaces and the ideas put forth in this paper are independent of the concrete scheme used. We use Loop's subdivision scheme [17], which acts on triangulated control polyhedra to define the surface.

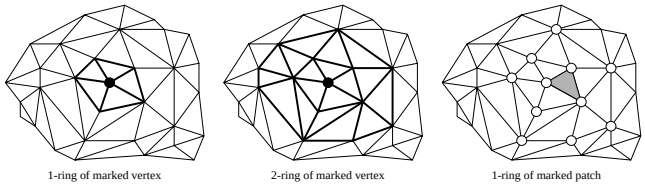


Figure 2: *Example of a control mesh. One control point has been marked together with its 1-ring (left). The middle shows the 2-ring of the marked control point. The basis function associated with the marked control point has its support entirely contained within the 2-ring. On the right the 1-ring of a patch. All marked vertices influence the shape of the highlighted patch.*

More precisely we consider the following setting. Let $\mathcal{M}^0 = \{\mathcal{P}^0, \mathcal{K}^0\}$ be an arbitrary topology 2-manifold (with boundary) control mesh with control points $\mathcal{P}^0 \in \mathbf{R}^3$, consisting of triangles $\{i,j,k\} \in \mathcal{K}^0$, edges $\{i,j\} \in \mathcal{K}^0$, and vertices $\{i\} \in \mathcal{K}^0$. $\mathcal{K}^0$ describes the combinatorial structure of the surface containing all control triangles, edges, and vertices; if a triangle is contained, so are its edges and vertices. The surface is a smooth function defined over this structure

$$S^0(u,v) = \sum_{i \in \mathcal{K}^0} p_i^0 B_i^0(u,v).$$

The B_i^0 are the basis functions associated with the subdivision scheme. The support of each basis function is the 2-ring around the given control point (see Figure 2, middle). The $p_i^0 \in \mathcal{P}^0$ are the control points, while (u,v) are defined over each control mesh triangle, typically through the use of barycentric coordinates. A given patch of the Loop surface is defined with reference to a triangle $\{i,j,k\}$ in the control mesh. Its degrees of freedom are the control points p_l^0 in the 1-ring of the triangle (see Figure 2, right). If the valences of $\{i\}$, $\{j\}$, and $\{k\}$ for triangle $\{i,j,k\}$ are all equal to six, the resulting patch is a regular quartic box spline patch. In case one (or all) of the corner vertices have valence other than six the patch consists of an infinite sequence of quartic box spline patches and we will refer to it as an extraordinary patch. Properties such as position and partial derivatives of regular patches are straightforward to evaluate, while extraordinary patches can be exactly evaluated through reduction to the regular case using eigenanalysis (for details see [29]). The resulting surfaces are C^2 everywhere, except at control points of valence other than six, where they are C^1.

With this setup the surface is completely parameterized by a (typically) coarse triangular control mesh. This control mesh represents the global shape and topology of the surface. One can now work with the surface in any algorithm which only assumes a parameterization. For example, the surface can be rendered by evaluating it on a suitably dense set of points and displaying the resulting piecewise linear approximation, just as is done in standard NURBS display libraries (e.g. [30]).

Instead of taking this point of view we treat the surface as the limit of repeated knot insertion, or subdivision. Figure 3 shows the Loop knot insertion stencils. The coefficient β depends on the valence of the central point and can be taken to be $\beta = 3/8k$ for $k > 3$ and $\beta = 3/16$ for $k = 3$ (this set of weights was first proposed by Habib and Warren [10]). There are also special rules at the boundaries and derived stencils to compute partial derivatives or

limit positions (see [11, 25, 1] for details). Aside from being particularly easy to implement and very efficient, this structure elegantly supports adaptive display and multiresolution representations. Subdivision converges quadratically to the limit, so that a small number of (adaptive) subdivision steps is sufficient for most tasks during interactive work.

Note: Standard graphics hardware can only display polygons (triangles). Consequently any smooth surface, be it a NURBS patch or a subdivision patch is by necessity displayed as a polyhedron. For performance reasons this is generally done through tessellation based on geometric approximation criteria [15]. When explicit (piecewise) polynomial representations are given this can be done through forward differencing. Another route is (adaptive) repeated knot insertion followed by a single step movement of control points to their associated smooth surface positions. Subdivision surfaces are most efficiently displayed in this way. This should not be confused with polygonal modeling. The subdivision surface is still a smooth object with a proper parameterization, exact evaluation functions, and all other attendant mathematical properties.

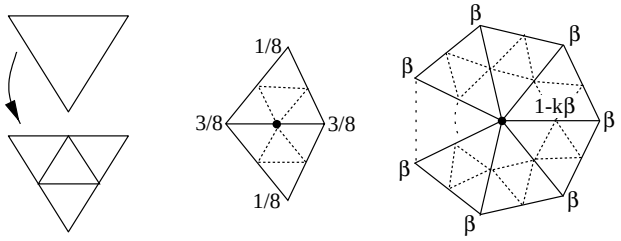


Figure 3: *In Loop's subdivision scheme a single subdivision step consists of quadrisecting all triangles (left column). New points are inserted through a weighted average of nearby points (middle stencil), while old points are moved to new positions based on a 1-ring filter (right stencil).*

1.3 Multiresolution Surfaces

Pure subdivision alone is not powerful enough for a flexible editing system. Figure 4 shows a model built with few patches (left). As expected it is very smooth with no fine details. This pure subdivision setting can be extended through the introduction of displacements at finer and finer levels of the subdivision hierarchy to enable multiscale modeling.

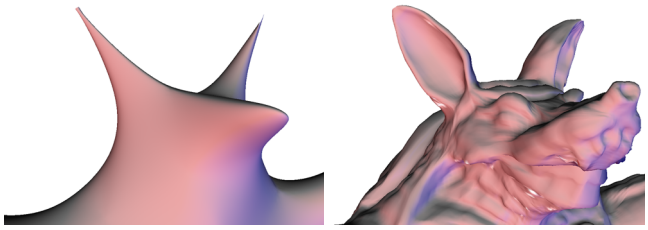


Figure 4: *Difference between pure subdivision (left) and subdivision with additional details added in each subdivision step (right). Adding details at all levels of subdivision, each parameterized in a local, surface dependent frame, provides a structure to support hierarchical editing semantics (first proposed in [9]).*

The key to this extension is the observation that the Loop basis functions observe a refinement relation induced by subdivision. Let $\mathcal{M}^0$ be the original control mesh and $\mathcal{M}^1 = \{\mathcal{P}^1, \mathcal{K}^1\}$ be the result of applying subdivision once to the original mesh. Both describe the same surface and each basis function associated with $\mathcal{M}^0$ is a linear combination of the dilated basis functions associated with $\mathcal{M}^1$. The weights of this linear combination are given by the subdivision stencils. This Ansatz is not unlike what one would do in a wavelet setting in which the Loop basis functions would serve as scaling functions [18]. The result is an increasingly dense sequence of function spaces given by the linear span, $V^l = \text{span}\{B_i^l | i \in \mathcal{K}^l\}$, of the basis functions at each subdivision level leading to $V^0 \subset V^1 \subset \ldots \subset V^L \subset \ldots$. This increasing sequence of spaces is exploited by associating a detail vector with each vertex at each level of subdivision in which it partakes. Detail vectors at some level l encode how much the control points for some surface $S^l \in V^l$ differ from the result of subdividing the control points at the coarser level $l-1$ of an approximation of S^l in V^{l-1}. This difference is represented in a local coordinate frame, just as in the H-splines setting [9], providing intuitive and powerful hierarchical editing semantics. Figure 4 shows an example on the right which differs from the left figure only through the introduction of detail vectors. The surface on the right is a member of V^5 and the surface on the left its approximation in V^0.

Exact evaluation at arbitrary parameter values can be achieved by simply summing all the contributions of all basis functions (weighted by their associated control points) overlapping a given point in parameter space. Extensive implementation details including adaptive approximation, incremental computation, and analysis algorithms, i.e., how to project a member of V^i into V^{i-1}, can be found in [31].

1.4 General Setup

Since we are focusing on small scale, curve-like feature editing we will deal with the perturbation of the surface in a small neighborhood of a curve drawn on the surface. This requires definition of a suitable neighborhood of the curve on the surface and the change of the surface inside this neighborhood according to some profile function. For each feature curve the user may choose a scale space V^l in which the feature should first appear. The curve is drawn onto an adaptive tessellation of the surface corresponding to the chosen subdivision level, using direct manipulation in 3D of a standard dragger tool [30]. The curve is represented as a quadratic spline control polygon. Its edges have to be inside triangles of the control mesh at level l. If any segment s intersects an edge of a level l control triangle, the segment is split (see Figure 5).

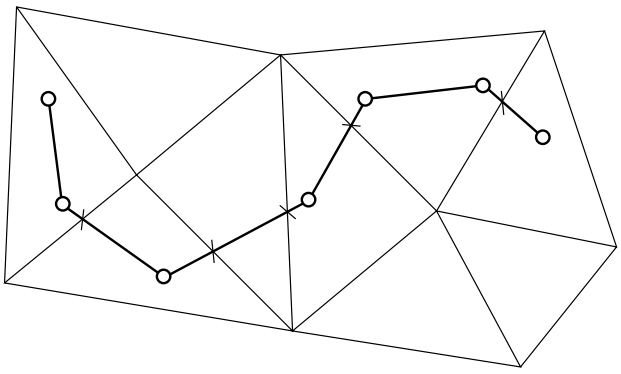


Figure 5: *The user draws directly onto the surface at a certain level of resolution, i.e., we may think of the curve as defined by a control polygon drawn at the selected level of the multiresolution hierarchy.*

Once the curve is established a neighborhood on the surface is

computed and control points of level l in this neighborhood are recomputed to follow the feature profile. The result is a modified surface $\tilde{S}^l \in V^l$ whose control points differ from the control points of the original surface $S^l \in V^l$ as $\tilde{p}_i^l = p_i^l + d_i^l$ for d_i^l non-zero only for i in the influence neighborhood of the curve.

Although we allow feature curves of arbitrary complexity there are some natural restrictions. Since our goal is to present an algorithm for the creation of *fine* level features, a natural requirement is that the feature size (width/height) be smaller than the local knot density at the (subdivision) level of definition. In other words, given the subdivision level at which the feature curve is defined, together with the region of influence, the control mesh at that level should admit local projection to the surface tangent plane without difficulty. The semantics of the perturbation can be thought of as that of a tool which, for example, carves a groove on the surface.

To summarize, our algorithm has the following properties:

- it deals with surfaces of arbitrary topology in a hierarchical editing environment;
- all perturbation data d_i^l is kept separate from the surface data, so the feature curves can be edited, moved as a whole, deleted, etc., without loosing any information about the original surface;
- feature curves are defined in world-space through direct manipulation to avoid non-intuitive distortion artifacts due to the parameterization (especially near points on the surface where multiple patches meet);
- feature curves may have arbitrary alignment with respect to isoparameter lines;
- feature curves may intersect each other as well as self-intersect;
- feature curves can be created on different multiresolution levels with features on the finer level using the coarser perturbation as their reference surface.

In the next sections we describe implementation details before presenting some examples built with our interactive modeling system, and discussing them.

2 Computing the Perturbation

For purposes of exposition we proceed in a number of steps. At first we consider the simplest case, a curve which consists of a single straight line segment. This then forms the basis for any collection of curves consisting of straight line segments only. Recall that the user defined curve is given as a sequence of samples. If these samples are dense enough using simply the influence of each linear segment is sufficient. In many cases however, we wish to treat the user supplied samples as the control points of a quadratic spline. This is achieved by noting that the thusly defined quadratic spline interpolates the midpoints of the edges of its control polygon and has a tangent at that point collinear with the control polygon edge. In that setting we continue to consider the influence of each control polygon edge as a straight line, but enhanced through a quadratic interpolation between adjacent segments.

Consequently, for all these cases the basic computational kernel derives from considering a single straight line segment.

2.1 Simplest Case: One Straight Line Segment

Assume that a single line segment I was drawn at some level l (see Figure 6). This level is now the "top feature level," since the feature will not exist on any coarser level. First, we fix the size r of the neighborhood around I and collect all triangles at level l which intersect the set $N_r = \{v | v \in \mathbf{R}^3, \mathrm{dist}(v, I) < r\}$. We will refer to the triangles which intersect N_r as neighborhood triangles and their associated control points as neighborhood points.

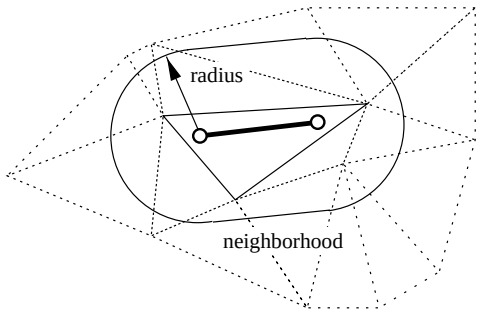


Figure 6: *A given curve segment lies entirely inside some triangle. In a first step all triangles are identified which will be influenced by this segment, i.e., overlap the region of influence, N_r.*

The surface will be perturbed by shifting all neighborhood points so they will follow some profile. To make this more precise, let us fix some neighborhood point p which is on the surface. It is closest to some point $b \in I$ (see Figure 7). Define the plane P_b which passes through the points b and p and contains the normal to the surface at b (z-axis in Figure 7). Let $C_b = S^l \cap P_b$ be an intersection curve of the plane P_b and the surface S^l. All points which lie on the curve C_b will be recomputed according to the same local frame. The frame is given by the normal to the surface at base point b (z-axis), the normal to the plane P_b (x-axis) and the tangent to C_b at b (see Figure 7). The origin of the local frame is at the base point b.

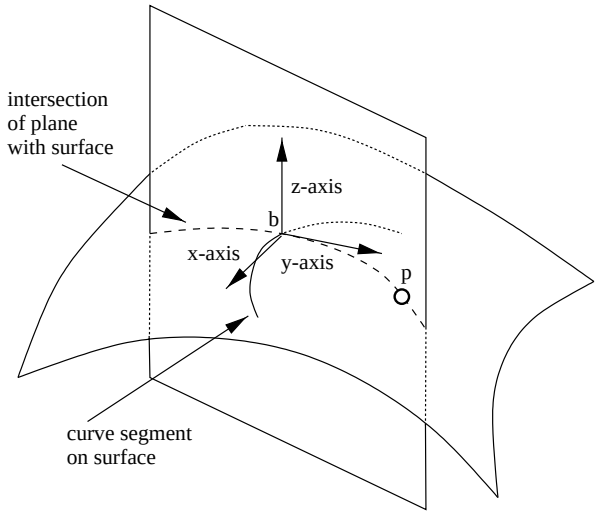


Figure 7: *Given a neighborhood point p on the surface and a closest point $b \in I$, let P_b define a plane through p and b, containing the surface normal at b (z-axis). This plane intersects the surface in a curve and the feature edit can thus be described as a function of a parameter running along the plane/surface intersection curve (see Figure 8).*

Now split P_b into two half-planes according to the local y coordinate, P_{b^+} for $y \geq 0$ and P_{b^-} for $y < 0$. The curve C_b has two parts $C_{b^\pm}$ lying in $P_{b^\pm}$. We will deal with them separately. Let us take C_{b^+} and define a new curve C'_{b^+} which lies in the same plane P_{b^+}, passes through the same boundary point and has the same tangent vector at that point (see Figure 8). The derivative at the base point b will define the "crease" of the feature along the length of the curve. Generally we will use a zero derivative at b for smooth features. Now, we move all surface points in the influence neigh-

borhood along the local z-axis towards the new curve C'_{b+}.

Notice that we defined the plane intersecting the surface to contain p and the closest point $b \in I$. If b is an interior point of I we will have the plane normal parallel to the line segment I. If on the other hand b is one of the endpoints of I we will typically not have parallelism between the x-axis and the plane normal. Our definition ensures that we get rounded endcaps as indicated in Figure 6.

2.1.1 Measuring Distance Along the Surface

Note that the only thing we really need from the plane P_b is the location of that point on the surface at which to match the boundary of the displacement curve C' (see "boundary condition" in Figure 8). There we have to match both position and derivative in order to ensure a smooth join between the perturbed region and the original surface. If the neighborhood is completely flat, the intersection of the boundary point with the half-plane P_{b+} would be given by the line along the vector $p-b$. So a ray from the point p in the direction $p-b$ would intersect the boundary at the same point as P_{b+}.

We will use this observation to find the boundary point in the general case without actually constructing the plane P_b. Take the ray from the point p which goes in the direction of $p-b$ and project it on the surface measuring out a distance r, the size of the neighborhood. When the surface on the given scale is approximately flat, the modified method of finding the boundary point coincides with the previous method (construction of P_b) to high accuracy. Note that we now control the size of the neighborhood by computing the length of a projected ray. It is possible to use either a distance along the surface or a distance in the tangent plane of the local frame at the base point. In our implementation we use distance along the surface with much success.

2.1.2 Ray Projection

The only remaining item is the projection of rays onto the surface. This could be done by measuring the length of a path on the actual smooth surface. In practice this would be accomplished through an adaptive integrator, which effectively takes straight line steps of sufficiently small size to achieve a desired accuracy. We apply this idea to the surface tessellation at the given resolution, which, by assumption, already satisfies the desired approximation criterion. Consequently the ray projection becomes a matter of traversing a triangulation and "unrolling" the length of the ray.

Starting the ray at the base point is well defined, since it is inside a triangle. A new ray is formed through projection into the plane of this triangle (Figure 9, top). The new ray, if it is long enough, intersects the boundary of the triangle at some point and then leaves the surface again. There are two cases which need to be treated separately. The point of intersection may be a vertex of the triangle (or may lie very close to the vertex) or it may be a general edge point. In either case we first find the tangent plane at that point. This plane can be derived explicitly from the subdivision rules. For an edge point we use linear interpolation between the two edge endpoint normals (Figure 9, middle).

Once the plane has been established, we project the rest of the ray onto this plane. For an edge intersection point we switch to the neighboring triangle and continue by projecting the ray onto the plane of the neighbor (Figure 9, bottom). For the vertex point we consider the 1-ring of triangles around the vertex and project this ring to the tangent plane. There we find the point at which the ray exits the 1-ring. Taking this point from the tangent plane back down to the surface we continue to "unroll" the original ray.

It is at this point in the algorithm that we require the assumption that the surface at the selected feature level be sufficiently flat. We need to guarantee that all vertices have 1-rings which possess a one-to-one projection to their tangent planes. Since the size of the 1-ring is supposed to be comparable to the size of the feature, our restriction guarantees that the original surface does not have its own features at this scale.

It is important to emphasize how we project in the case of general edge points. When the ray intersects an edge we project twice. This is not the same as one projection directly onto the plane of the new triangle. The problem with single projection is that geometrically close rays which lie on different sides of a vertex would quickly diverge (see Figure 10). Theoretically double projection does not guarantee stabilization but we find that it works well in practice (for a more indepth discussion of these issues see [22]).

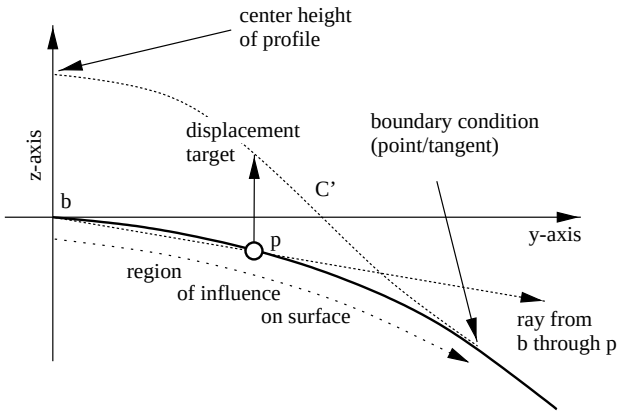


Figure 8: *View of the setup in the plane intersecting the surface in b (see Figure 7), we are ready to move all neighborhood points p towards a new location on the curve C'_{b+}. This new curve is given by interpolation and derivative constraints. At the local origin (the anchorpoint $b \in I$ on the surface) we fix a height and (typically) a zero derivative. At a distance r along the surface (the region of influence) we require C'_{b+} to match in value and derivative. We are in effect replacing a section of the surface with a new surface fixed by the boundary conditions.*

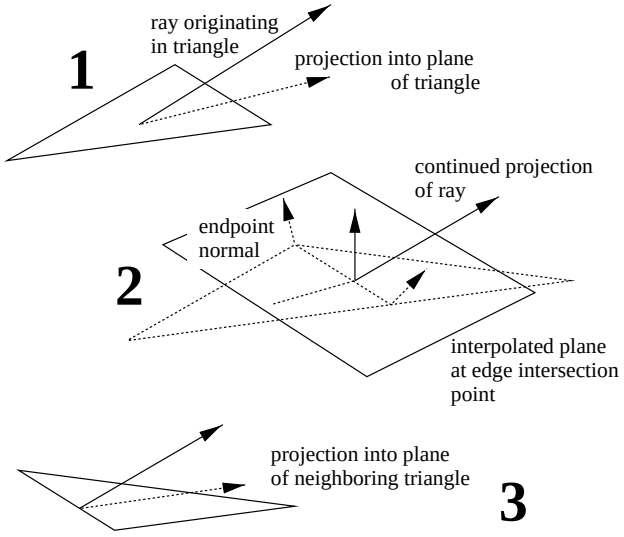
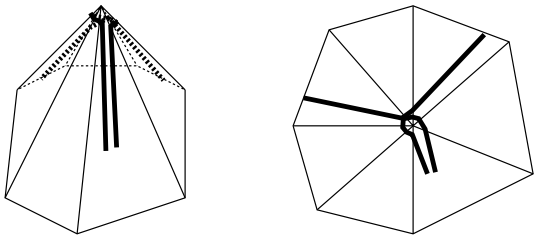


Figure 9: *To trace the region of influence of a line segment we must compute the projection of rays onto the surface. This is done in steps as we "unroll" the ray.*



perspective view top down view

Figure 10: *Projection of rays to the surface is delicate near vertices, where slightly different rays can diverge. We use special projection rules for rays grazing vertices and otherwise perform a 2-step projection whenever crossing any edge to stabilize these situations.*

2.1.3 Discussion

We should emphasize that our way to compute offsets in a single frame for all points on the curve C_b is different from two related approaches described in [28] and [4]. In the former a density function in a tubular neighborhood of a skeleton curve is defined to smoothly decay towards the boundary of its finite region of influence. This function, together with a number of parameterized linear transformations with respect to a reference curve warps all surface points in its volumetric region of influence. In our setting there is no underlying offset parameterization and the influence of the feature "travels" only along the surface, not through space.

Similar to the surface pasting setting [4] we use displacements in local coordinate frames. However, we do so in a discrete setting without the need for a mapping into some domain. Control points of the multiresolution representation are directly manipulated. Additionally, our approach *removes* a section of surface and replaces it with a shape molded according to the boundary conditions found in the region of influence. This approach better fits the intuition of a designer who wants a tool to make, for example, a groove and expects that the shape of the feature is relatively insensitive to the shape of the base surface.

In our implementation we took a cubic polynomial for the profile function. Since we have constraints for the value and derivative of the function at the boundary, our shape depends on the surface at the boundary. Obviously, one can not expect complete independence unless the value of the derivative at the boundary is fixed. We can also construct features with fixed derivatives though the perturbed surface will no longer be smooth. Using cubic polynomials we have a two parameter family of profiles. The free parameters are the value of the function and its derivative at the origin. These values allow us to vary the height of the perturbation and the crease along the base curve.

2.2 Summary

At this point we have the following algorithm. For each control point p in N_r we find b and the distance vector $p - b$. Using the latter together with the normal to the surface at b we construct a local frame in which we will recompute p. First we project the ray from p in the direction $p - b$ on the surface until we reach the boundary of the region of influence. There are two ways to measure distance: 3D Euclidean and along the surface. Since we use projections onto the surface for finding boundary points, it is more natural to always consider distances as measured along the surface. Note, the definition of the neighborhood N_r uses 3D Euclidean distance, which never exceeds the distance along the surface. Then we calculate the new position of the control point p according to the profile with the given boundary parameters.

An example of multiple curves which overlap is shown in Figure 11, with an indication of the width of the neighborhood.

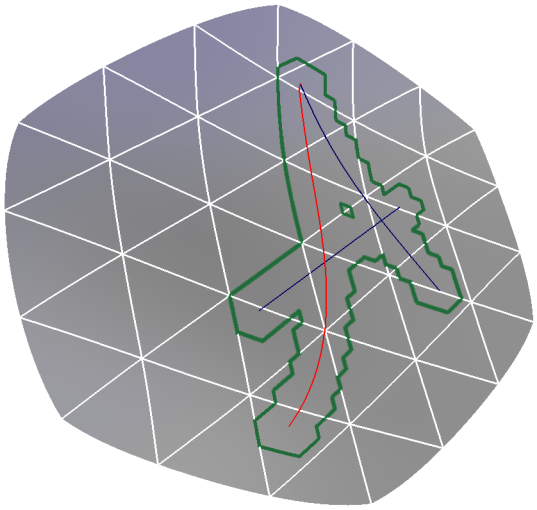


Figure 11: *Example of three user specified feature curves which overlap. The surface shown is the level at which the curves were drawn. Note that the feature lines do not align with any patch boundaries. The blue outline delineates all the neighborhood triangles. For the final surface resulting from these feature curves see Figure 16.*

2.3 Multiple Segments

Now we consider the case when we have multiple line segments. Again we start by collecting all triangles which belong to the neighborhood of the segments. For each triangle we record all segments on the surface that influence it before recomputing any control point positions. The new control point position is now computed with respect to all segments.

These segments are divided into two groups: those which give positive offsets in their local frame and those which give negative offsets. Going through the offset computations for each segment we keep a running max (for positive offsets) and min (for negative offsets).

Note that in general this procedure is not commutative and depends on the order in which the segments were created. This is so because the displacements are computed in different tangent plane coordinate frames for each of the base points. But if the original surface on the given scale is flat enough the difference is very small. At the last step we interpolate between negative and positive values.

2.4 The General Case: Arbitrary Curve

Any line can be approximated by a set of straight segments so we have enough tools to build features along an arbitrary curve. Here we discuss two ways for optimization.

First, it may happen that a curve consists of a large number of very small segments. This would incur considerable computation for each control point in the influence neighborhood. To deal with this problem we only consider segments which are *locally* closest to a given control point. For a given control point p let b_i^p be the base point of the segment i. If this point is an end point of the i-th segment and is not an end point of the neighbor, we skip the segment i (see Figure 12). This procedure saves computation time and also solves the commutativity problem for adjoining segments.

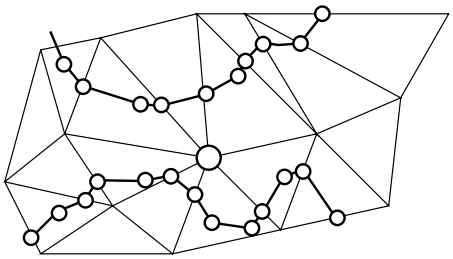


Figure 12: *All line segment candidates which may influence a given neighborhood point must be tested to find the closest one. If in a given test the endpoint of a segment is the closest point, then its segment neighbor must also consider the same endpoint closest, otherwise the line segment is rejected.*

The second problem is that sometimes the base feature level is too coarse to represent the desired curve smoothly. This causes noticeable creases to appear where segments meet. To smooth the profile we use quadratic curves to interpolate between any two segments and use the points on the quadratic interpolant as origins for the local frames. Each quadratic curve piece starts in the middle of a segment and has the same tangent as the segment. It ends in the middle of the next segment with the tangent vector of that segment. The interpolation allows us to improve the quality of the profile for coarse line segments. We are effectively constructing a quadratic spline using the original user input as the control polygon.

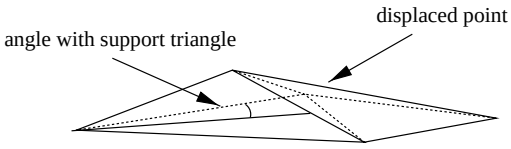


Figure 13: *Recursive quadrisection according to the Loop subdivision scheme, and any additional detail perturbations, stops when a local flatness criterion is met.*

2.5 Adaptive Computations

The perturbation is computed according to the multiresolution representation. First, we consider all control points of the level at which the curve was drawn. Then control points of finer levels are recomputed. Each triangle is quadrisected as usual. We fix some threshold for the surface flatness and continue to subdivide the triangle only if an edge midpoint violates this threshold. The threshold measures the angle between the plane of the parent triangle t and the line connecting the midpoint with its opposite vertex on t (see Figure 13). We check this condition for both triangles which share the midpoint. Subdivision of a triangle is stopped if all three edge midpoints are close enough to the plane of the parent triangle and its neighbor across the corresponding edge.

2.6 Feature Curves in the Hierarchy

We allow feature curves to exist on different subdivision levels. As mentioned earlier, for a finer level feature curve the base surface is given by the perturbation from any coarser level (see Figure 14). Since features do not exist at levels coarser than the level at which they were created, any changes in finer features do not affect coarser ones. On the other hand, any modification to the coarser feature propagates recomputations to all finer level features. Figure 15 demonstrates this with a coarse level surface edit.

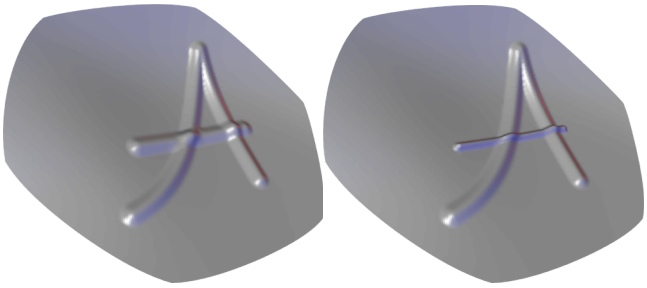


Figure 14: *Feature curves defined at different levels are layered according to the definition level and can be manipulated independently (see Figure 16 top left for the single layer version).*

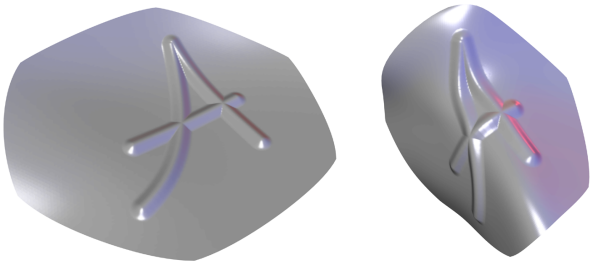


Figure 15: *Applying a standard hierarchical editing modification to the coarse surface carries along any features at finer scales as one would expect.*

2.7 Additional Tools

As an illustration of some possible extensions which can be considered in our setting we implemented two additional ideas: variable sized neighborhoods and variable height. The former allows one to have differently sized feature neighborhoods on the left and right of the curve.[2] To make the algorithm well defined around the end points of the curve we smoothly interpolate between the left and right values. Using this tool one can create shapes with different slopes on the left and right sides of the base line (see Figure 16, bottom middle).

The second modification can be used, for example, to linearly increase the height of the profile along the curve. During control point recomputation we project to the base curve and compute the length from the beginning of the curve to the point of projection. Then the height factor for a control point is scaled by this length. In particular, this tool allows for the creation of grooves which gradually rise (see Figure 16, bottom right).

In our current implementation the following parameters can be adjusted independently for each feature curve

- size of the neighborhood (Figure 16, top row);
- ratio of the neighborhood sizes on the left and right (Figure 16, bottom middle);
- profile height (Figure 16, top row and bottom row left);
- profile crease;
- ratio of height at the beginning and at the end of the curve (Figure 16, bottom right);
- threshold for adaptive computations

Figure 17 shows a more complex example of an irregular topology shape (top left) with a variety of feature curves overlaid on it.

[2] Any curve is considered oriented based on how it was drawn.

Note that the holes are not trimmed but rather the original patch network was layed out to have these holes.

3 Conclusion

We have described an algorithm to enable the creation and control of curve-like, fine-level features in a subdivision based multiresolution geometric editing environment. The user chooses a level of the hierarchy on which to draw a free form curve. The curve is realized as a quadratic spline with attributes such as the neighborhood size, width, and height, which control the perturbation of finer resolution levels in the vicinity of the curve. The perturbation is performed through direct displacement according to a profile curve which "grafts" a new surface piece into the old surface, respecting smooth boundary conditions. The resulting semantics is that of a tool, carving features into the surface. There are no constraints on the location of the curve with respect to control mesh edges. Details of the curve must however be resolvable at whatever level the curve was defined. Multiple curves are allowed to overlap or selfintersect and a rich set of editing operations is supported on the curves themselves. Since the algorithm is realized in a subdivision framework there is no need to explicitly manage domain space mappings or to worry about the behavior of features subject to coarse scale edits. The main assumption of the algorithm is that the feature size is small compared to the local scale of the underlying surface. We do not believe this to be a limitation as larger scale features are better dealt with in the usual hierarchical editing approach.

Subdivision, and its multiresolution extensions, are a strong contender for many modeling scenarios. Their chief advantage is their ability to perform gracefully in the arbitrary topology setting. Additionally, they naturally support many state of the art algorithms such as level-of-detail display, progressive transmission, and adaptive approximation for numerical modeling tasks. The present paper adds to the growing palette of available tools for subdivision surfaces.

Acknowledgments

The research reported here is supported in part through a NSF CAREER award (ASC-9624957). Other generous support was provided by the Sloan Foundation, the Okawa Foundation, an IBM partnership award, and Alias|wavefront. Very special thanks to Khrysaundt Koenig for living on the bleeding edge of our modeler. Thanks also to the folks at DesignWorks/USA, Steven Schkolne, and Jim Barry for many discussions on modeling. We also wish to thank the anonymous reviewers for their helpful comments.

References

[1] BIERMANN, H., LEVIN, A., AND ZORIN, D. Piecewise Smooth Subdivision Surfaces with Normal Control. Tech. rep., Courant Institute, New York University, January 1999.

[2] CATMULL, E., AND CLARK, J. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Assisted Design 10*, 6 (1978), 350–355.

[3] CERTAIN, A., POPOVIĆ, J., DEROSE, T., DUCHAMP, T., SALESIN, D., AND STUETZLE, W. Interactive Multiresolution Surface Viewing. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 91–98, 1996.

[4] CHAN, K. K. Y., MANN, S., AND BARTELS, R. World Space Surface Pasting. In *Graphics Interface '97*, 146–154, May 1997.

[5] CHANG, Y., AND ROCKWOOD, A. P. A Generalized de Casteljau Approach to 3D Free–Form Deformation. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, 257–260, July 1994.

[6] COQUILLART, S. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, vol. 24, 187–196, August 1990.

[7] DEROSE, T., KASS, M., AND TRUONG, T. Subdivision Surfaces in Character Animation. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 85–94, July 1998.

[8] DOO, D., AND SABIN, M. Analysis of the Behaviour of Recursive Division Surfaces near Extraordinary Points. *Computer Aided Design 10*, 6 (1978), 356–360.

[9] FORSEY, D. R., AND BARTELS, R. H. Hierarchical B-Spline Refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, Vol. 22, No. 4, pp. 205–212, August 1988.

[10] HABIB, A., AND WARREN, J. Edge and Vertex Insertion for a Class of Subdivision Surfaces. Preprint. Computer Science, Rice University, 1996.

[11] HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. Piecewise Smooth Surface Reconstruction. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, 295–302, 1994.

[12] HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. Direct Manipulation of Free-form Deformations. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, 177–184, July 1992.

[13] KOBBELT, L. A Variational approach to subdivision. *Comput. Aided Geom. Des. 13* (1996), 743–761.

[14] KOBBELT, L. P., DAUBERT, K., AND SEIDEL, H.-P. Ray Tracing of Subdivision Surfaces. In *Proceedings of 9th Eurographics Rendering Workshop*, 1998.

[15] KUMAR, S., MANOCHA, D., AND LASTRA, A. Interactive Display of Large-Scaled NURBS Models. *IEEE Transactions on Visualization and Computer Graphics 2*, 4 (1996), 323–336.

[16] LEVIN, A. Combined Subdivision Schemes for the Design of Surfaces Satisfying Boundary Conditions. To appear in CAGD, 1999.

[17] LOOP, C. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[18] LOUNSBERY, M., DEROSE, T., AND WARREN, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *Transactions on Graphics 16*, 1 (January 1997), 34–73.

[19] MACCRACKEN, R., AND JOY, K. I. Free-Form Deformations with Lattices of Arbitrary Topology. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 181–188, August 1996.

[20] PETERS, J., AND NASRI, A. Computing Volumes of Solids Enclosed by Recursive Subdivision Surfaces. *Computer Graphics Forum 16*, 3 (1997), C89–C94.

[21] PIEGL, L. A., AND TILLER, W. *The NURBS Book*, 2nd ed. Monographs in Visual Communications. Springer Verlag, 1997.

[22] POLTHIER, K., AND SCHMIES, M. Straightest Geodesics on Polyhedral Surfaces. Tech. Rep. 327, Technical University of Berlin, SFB 288, June 1998.

[23] REIF, U. A Unified Approach to Subdivision Algorithms Near Extraordinary Points. *Computer Assisted Geometric Design 12* (1995), 153–174.

[24] SCHRÖDER, P., AND ZORIN, D., Eds. *Course Notes: Subdivision for Modeling and Animation.* ACM SIGGRAPH, 1998.

[25] SCHWEITZER, J. E. *Analysis and Application of Subdivision Surfaces*. PhD thesis, University of Washington, 1996.

[26] SEDERBERG, T. W., AND PARRY, S. R. Free-Form Deformation of Solid Geometric Models. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, vol. 20, 151–160, August 1986.

[27] SEDERBERG, T. W., ZHENG, J., SEWELL, D., AND SABIN, M. Non-Uniform Recursive Subdivision Surfaces. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 387–394, July 1998.

[28] SINGH, K., AND FIUME, E. Wires: A Geometric Deformation Technique. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 405–414, July 1998.

[29] STAM, J. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 395–404, July 1998.

[30] WERNECKE, J. *The Inventor Mentor*. Addison-Wesley, 1994.

[31] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 259–268, 1997.

[32] ZORIN, D. N. *Subdivision and Multiresolution Surface Representations.* PhD thesis, Caltech, Pasadena, California, 1997.

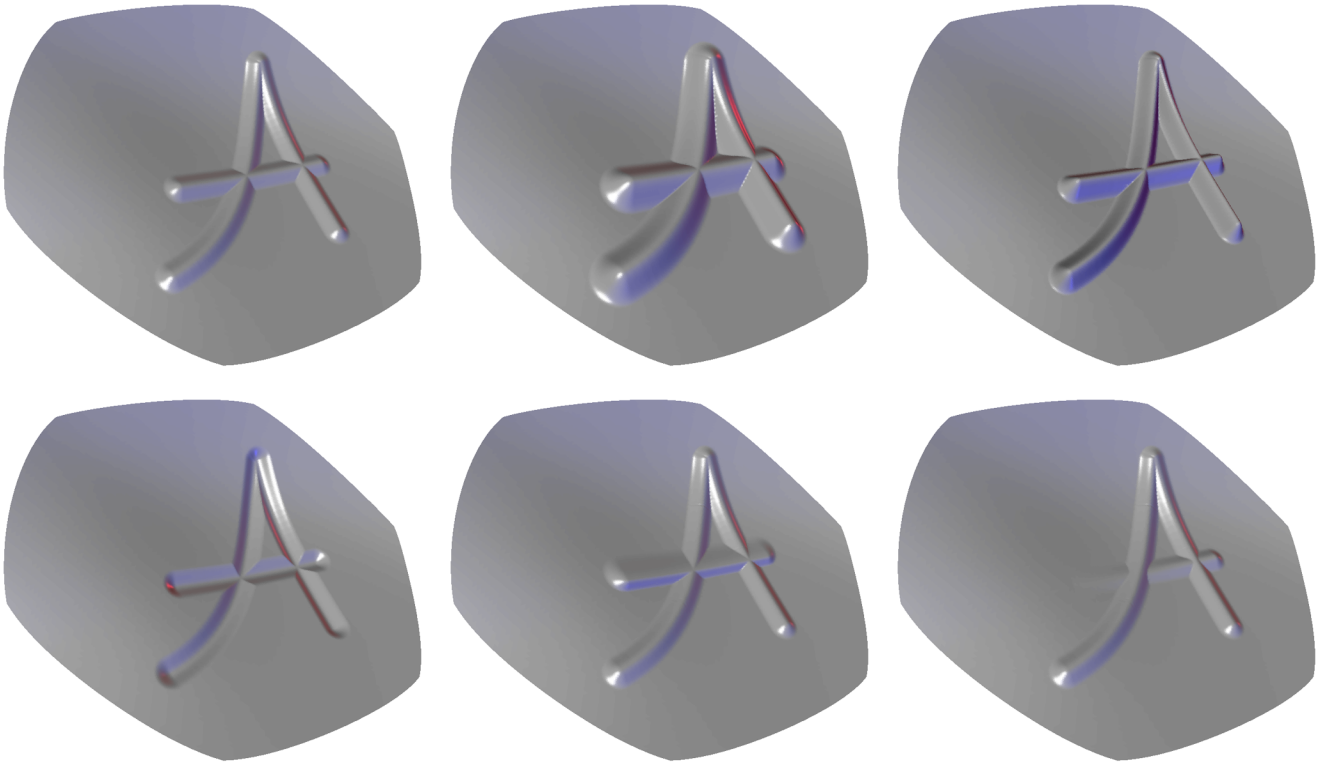


Figure 16: *Given a simple set of lines (see Figure 11) this sequence illustrates some of the parameters the user can control (see the itemization in Section 2.7).*

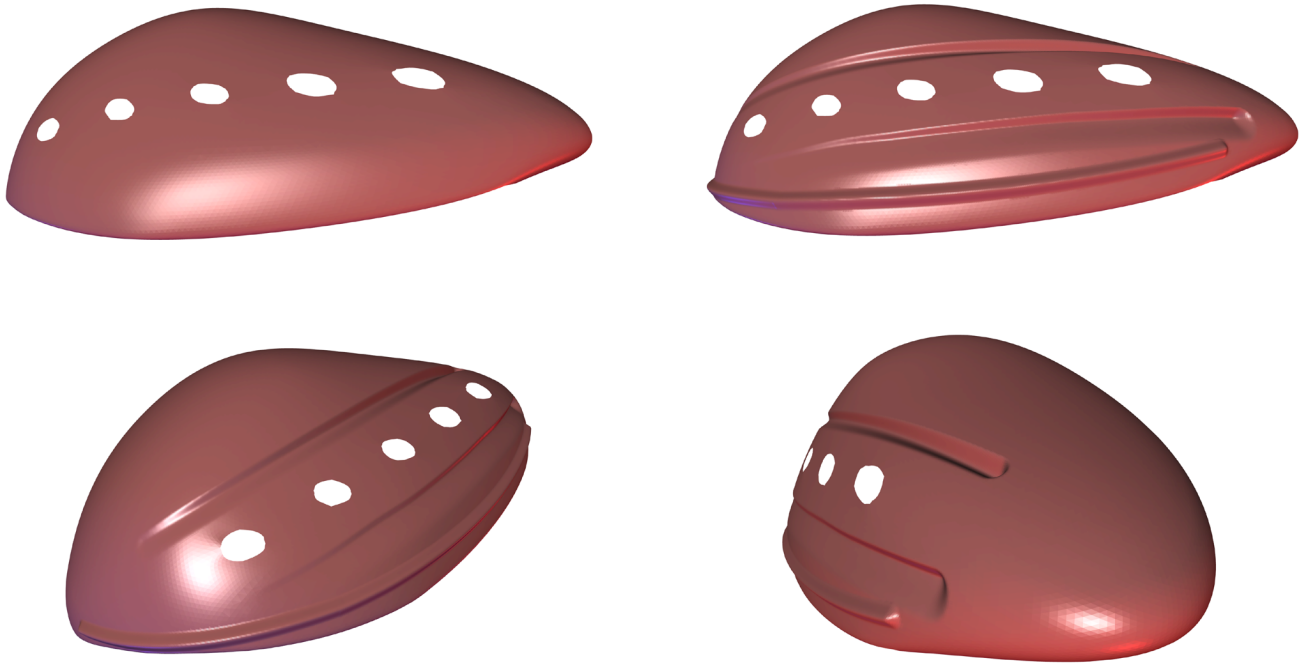


Figure 17: *A more complex example of using feature curves on an irregular topology base shape (top left). Overlaying a number of feature curves during an interactive editing session resulted in the shape on the top right. The bottom images provide additional views (Design study for a bicycle helmet; Khrysaundt Koenig).*