# Spherical Manifolds for Adaptive Resolution Surface Modeling

Cindy M. Grimm

Washington Univ. in St. Louis*

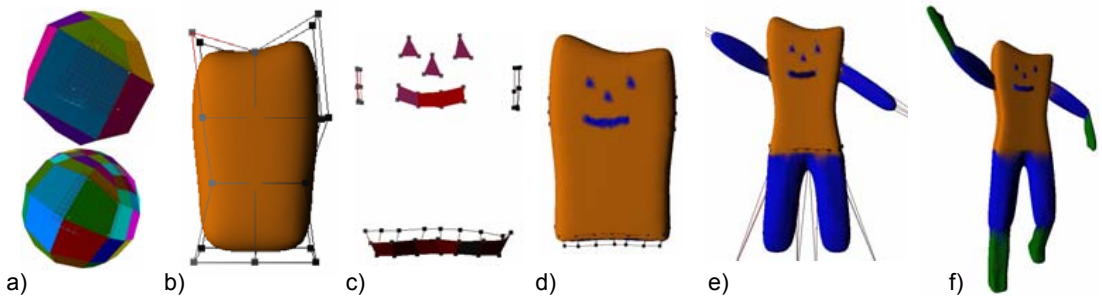


Figure 1: Creating a surface with spherical topology. a) Sketch mesh (22 faces) and first subdivision level mesh embedded in the spherical domain. b) Initial geometry of sketch mesh and resulting surface (129 overlapping surface patches). c) Geometry specifying the next hierarchical level (average patch overlap, 3.4) This geometry is created by drawing on the surface in b). d) The resulting surface, colored by hierarchical level. e) Editing the first hierarchical level to produce arms and legs. f) Adding and editing a second hierarchical level.

## Abstract

We present a surface modeling technique that supports adaptive resolution and hierarchical editing for surfaces of spherical topology. The resulting surface is analytic, $C^k$, and has a continuous local parameterization defined at every point. To manipulate these surfaces we describe a user-interface based on multiple, overlapping subdivision-style meshes.

**CR Categories:** I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

**Keywords:** Surface modeling, splines, hierarchical, parameterization, hypberbolic geometry, arbitrary topology

## 1 Introduction

In the 1880's mathematicians began studying the class of surfaces that are *manifold*, *i.e.*, surfaces that are locally Euclidean. They made the observation that any surface of this type, no matter how complicated, could be locally analyzed by mapping portions of the surface to the plane. These local maps were, in general, easier to reason about than studying the entire surface. By taking smaller or larger portions of the surface, they could analyze properties of the surface at different scales. This technique also made it possible to "move" the analysis continuously across the surface without encountering seams; the transition from one local map to an overlapping one was well-defined.

This paper presents a constructive method, based on the idea of locally planar maps that overlap, for building analytic surfaces of spherical topology. These surfaces have the property that we can define a map from any open disk on the surface to the plane. We use this property to allow free-form addition of "patches", at any resolution and position on the surface. Each patch provides additional degrees of freedom for manipulating the surface. By placing patches only where needed, we reduce the total complexity of the surface. $C^k$ continuity for any $k$ is guaranteed without the use of geometric constraints between patches.

To support adaptive resolution editing the user constructs detail patches which then over-ride the coarser geometry. The geometry of these detail patches is expressed in terms of the coarse geometry, so that the detail moves automatically with the coarse geometry. Unlike previous hierarchical approaches we place no constraints on where the detail patches are placed or how they are aligned with respect to the coarse geometry. We also provide a sound mathematical framework for defining how the detail patches are blended into the existing surface, producing $C^k$ blends without the use of geometric constraints.

Figure 1 outlines the construction process. Like many existing approaches, the user first sketches the rough shape by creating a coarse mesh. They can adjust the surface by editing the coarse mesh and its subdivision levels. Unlike existing approaches, the user can now add detail to the current surface by drawing on it to create a new control mesh. There are no constraints placed on the geometric shape or location of this new mesh. Changes to this new mesh are blended into the existing surface. Editing the original mesh results in the coarse changes propagating appropriately to the detail mesh.

There are two advantages to approaching modeling in this way. First, any function over the entire spherical domain can be built from functions defined on planar topologies. Since planar functions are well-understood, this allows us to leverage off a large body of previous work. Second, the placement of the additional patches can be matched to the desired function resolution at that point on the surface. This makes it possible to add arbitrary amounts of detail

*email:cmg@cs.wustl.edu

where needed.

**Contributions:** First, we present a practical method for representing, manipulating, and locally parameterizing spheres and meshes embedded in them. Second, we demonstrate how to build an analytical surface, with the look and feel of a subdivision surface, from an input mesh. Third, we provide a re-formulation of the manifold embedding equation [Grimm and Hughes 1995] that supports adaptive, hierarchical editing. This equation represents a mathematically sound method for defining $C^k$ surface pasting functions. Although in this paper we restrict ourselves to spherical topologies, the approaches described here easily extend to other topologies.

### 1.1 Example editing session

The user begins by creating a sketch mesh that approximates the basic desired shape (see Figures 1). The resulting surface admits to subdivision-style editing, both of the original mesh and the first, second, or third level subdivision.

The user next outlines where they want new surface patches to be placed by drawing on the existing surface. This creates a second mesh which is embedded in the existing surface. There are no constraints on this mesh; it can cover all or some of the surface, it can overlap itself, and it can be aligned arbitrarily with respect to the initial sketch mesh.

Using the second mesh, and its first, second and third subdivision levels, the user then edits the surface, only affecting the area underneath the mesh. If the original sketch mesh is changed, the second mesh moves along with it, much as a hierarchically-defined spline patch would.

The user can continue to add new levels anywhere on the surface. Each new level-mesh "over-rides" the surface underneath it; however, the vertices of the higher level mesh are kept in the coordinate frame of the surface, so changes to lower levels propagate appropriately to higher levels. The new levels do not need to lie inside of the previous level.

## 2 Related work

The problem of analytically modeling surfaces of arbitrary topology has attracted a great deal of attention, as has the problem of editing free-form surfaces. We summarize the primary approaches to the problem, but a complete summary is beyond the scope of this paper.

The three basic approaches are hole filling with $n$-sided spline patches, subdivision surfaces, and alternative domains. Hole filling [Peters 2002; Hollig and Mogerle 1990] has a rich history and has evolved from the desire to extend spline patches to surfaces of arbitrary topology; a recent review by Peters [Peters 2004] discusses parameterization and curvature issues with this approach. The surface is defined by a network of patches which may also define geometric constraints [Loop 1994; Loop and DeRose 1990; Warren 1992]. More recent approaches also combine the remaining degrees of freedom into geometrically useful controls [Sederberg et al. 2004; Zheng 2001]. Our approach is fundamentally different than patch filling because we do not use geometric constraints to maintain continuity. On a more subtle level, most patch-filling approaches define some form of parameter-space extension into neighboring patches in order to construct their continuity constraints. In our approach, the overlapping parameterization is inherent in the surface construction process.

Subdivision surfaces [Doo and Sabin 1978; Catmull and Clark 1978] were originally developed as an alternative approach to extending splines to arbitrary topology surfaces. Interactive, multiresolution editing was first described by Zorin [Zorin et al. 1997] and Pulli [Pulli and Lounsbery 1997]. Detail editing [Khodakovsky and Schröder 1991; Biermann et al. 2002b] creates fine-level features using a combination of special-purpose refinement rules. Stam [Stam 1998] showed that, except at extraordinary points, subdivision surfaces can be represented by spline patches, which provides a local parameterization for most points on the surface, and can be extended in most places [Stam 2003] to adjacent areas. De Rose [DeRose et al. 1998] also demonstrated local parameterization in terms of texture mapping, and explicit control over crease features. A hybrid approach [Gonzalez-Ochoa and Peters 1999] uses a hierarchical mesh to specify a network of patches and geometric constraints that maintain continuity. Our surfaces support subdivision-style editing, but produce an analytical surface of any continuity. We also support both hierarchical editing and the addition of detail anywhere and at any scale. Subdivision surfaces provide editing at any point by subdividing enough — however, the influence of an individual vertex shrinks with every subdivision step.

Several papers describe surface construction techniques using manifolds or alternative domains. Grimm's approach [Grimm and Hughes 1995] begins with a mesh and builds a manifold with one chart per mesh element. The approach of Navau and Garcia [Navau and Garcia 2000] first subdivides the mesh to isolate extraordinary vertices. They then embed sections of the mesh in the plane so that the overlap regions are rectangular and blend together in the middle in a $C^k$ fashion. Subdividing the mesh to isolate the extraordinary vertices can result in a large number of patches; however, the patches themselves are simpler than the ones presented by Grimm [Grimm and Hughes 1995]. Ying [Ying and Zorin 2004] creates a manifold over a mesh by "unwinding" the faces of a vertex into the plane, then building blend and embedding functions over these vertex charts. Gu et. al. [Gu et al. 2005] construct an affine manifold over the mesh structure. These manifold approaches are similar to ours in that they produce a smooth surface from a mesh; however, they do not support adding additional patches at arbitrary places and sizes on the manifold. Grimm [Grimm 2002] describes an approach that also uses charts defined on the sphere except that they define only a fixed number of charts (six) and no hierarchical support.

He et. al. [He et al. 2005] present a technique for defining splines directly on the sphere.

Surface pasting [Chan et al. 1997; Biermann et al. 2002a; Leung and Mann 2003] and hierarchical editing [Forsey and Bartels 1988] are approaches to adding local detail to the surface without adding more degrees of freedom across the entire surface. For spline patches, this involves creating a new patch that is "glued", using geometrical constraints, to the original patch. Continuity is not always guaranteed. Detail patches are also constrained to lie within the boundary of an original patch. We allow new patches to be placed anywhere, guarantee continuity, and require no geometric constraints.

## 3 Initial surface construction

Our surface is represented by a set of patches (which are defined by the mesh) and information about how to glue those patches together. Unlike traditional spline approaches, these patches are overlapped and then blended together; they are not joined by matching continuity along the edges. This provides a great deal of freedom — patches can overlap arbitrarily and are not confined to lie within, or abut, existing patches.

The key to this approach is defining the patches using local, invertible, $C^\infty$ parameterizations of the sphere [Grimm 2004]. The overlap regions and glue functions then arise naturally by looking at how the domains of these local functions overlap on the sphere. There exist standard techniques for creating local parameterizations

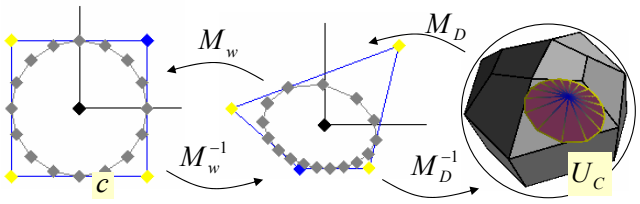


Figure 2: Defining a mapping between a subset of the sphere and a disk in the plane. Left: Mapping the circle to an ellipse using a projective transform $M_w^{-1}$. Right: Mapping the ellipse to a disk on the sphere using the inverse of a stereographic projection $M_D^{-1}$.

of any size anywhere on the sphere. Theoretically, we can cover all but one point of the sphere with a single parameterization; in practice we limit ourselves to a maximum size of one hemisphere to prevent undue distortion.

Unlike previous approaches [Ying and Zorin 2004; Grimm and Hughes 1995; Navau and Garcia 2000; Gu et al. 2005] that use the topology of the sketch mesh to define their overlaps and glue functions, we only use the sketch mesh to define the number of patches and their locations on the sphere. This enables us to subsequently add more patches that overlap the existing ones in arbitrary ways.

After defining the overlap structure we need to create geometry. We do this by defining geometry and a blend function for each patch (Section 4 discusses how to add patches that over-ride existing geometry). We define a bijection between the subdivision surface of the sketch mesh and the sphere. Each individual patch is then fit to its corresponding part of the subdivision surface; this insures that the patches already mostly agree where they will be blended together.

In the following sections we provide formal definitions of the domain $S^2$, how we build patches (charts) on $S^2$, and how we blend the results together.

### 3.1 Overview

More formally, we begin with a concrete representation of the sphere $S^2$. We next define a general method for creating a *chart* on $S^2$ using the composition of a stereographic projection (which takes all but one point of the sphere to the infinite plane) and a projective transform. The latter function allows us to better control the shape of the chart. We define the amount of the sphere covered by the chart by defining the co-domain of the chart, then taking the inverse of the above. The co-domain of the chart is defined to be a unit circle centered at the origin; the inverse projective transform takes the circle to an ellipse in the plane, then the inverse stereographics projection takes the ellipse to a disk on the sphere (see Figure 2).

Each chart maps a portion of $S^2$ to a circle in the plane. This saves us from having to define an embedding function on a part of $S^2$ itself; instead, we first map the portion of $S^2$ to the plane. Then, we can use any standard plane embedding function on the circle (polynomial, spline, *etc.*) to define the local shape. The final patch is then a composition of the chart mapping and the planar embedding function.

More formally, to embed $S^2$ we define an embed $E_c : \mathbb{R}^2 \to \mathbb{R}^3$ and a blend $B_c : \mathbb{R}^2 \to \mathbb{R}$ function for each chart. The final embedding $E : S^2 \to \mathbb{R}^3$ is a blended combination of these individual embeddings:

$$E(P) = \frac{\sum_c B_c(\alpha_c(P))E_c(\alpha_c(P))}{\sum_c B_c(\alpha_c(P))} \quad (1)$$

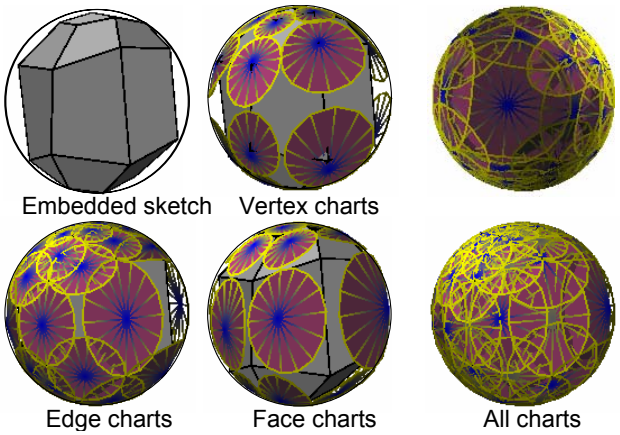


Figure 3: Upper left: The sphere with the base mesh from Figure 1 embedded on it. Upper right: The charts corresponding to the faces in the mesh. Lower left: The edge charts. Lower right: The vertex charts.

where $B_c$ is defined to be zero where $\alpha_c$ is not defined. If the following hold, than $E(P)$ is $C^k$ [Grimm and Hughes 1995]:

- The functions $\alpha_c$, $E_c$, and $B_c$ are all at least $C^k$.
- The $k$ derivatives of $B_c$ go to zero by the boundary of the domain of the function $\alpha_c$.
- There is at least one non-zero function $B_c$ at every point in $S^2$.

In order to define a complete, initial embedding of $S^2$ we need every point of $S^2$ to be covered by some chart. This is the role of the sketch mesh $\mathscr{M}$ (see Figure 3). We first embed $\mathscr{M}$ into the domain $S^2$ using any existing technique [Gotsman et al. 2003; Grimm 2004; Saba et al. 2005]. Each element $v, e, f$ of $\mathscr{M}$ now covers some portion of the domain ($D_{v,e,f} \subset S^2$). For each face $f$ we create a chart that covers the interior of the spherical polygon $D_f$. For each edge $e$ we create a chart that covers the edge and extends midway into the adjacent face polygons. The vertex $v$ charts are centered on the vertex and also extend midway into the adjacent face polygons.

To build the surface geometry we create geometry for each chart that approximates the subdivision surface of $\mathscr{M}$. We can apply the subdivision process to both the mesh $\mathscr{M}$ and the mesh embedded in the domain. The former defines the desired geometry; the latter creates a one-to-one correspondence between points in the domain $S^2$ and points on the desired $\mathbb{R}^3$ geometry.

### 3.2 Domains

Here we define $S^2$, and how we represent points, edges, and polygons on $S^2$. In traditional Euclidean geometry *i.e.,* a mesh in $\mathbb{R}^3$, it suffices to store the topological information of the mesh, and the geometric information just at the vertices. The geometric information of the edges and faces[1] is constructed from the vertex information:

$$G(v) = (x,y,z) \quad (2)$$

$$G(\{v_1,v_2\}) = (1-t)G(v_1)+tG(v_2), \quad t \in [0,1] \quad (3)$$

$$G(\{v_i\}) = \sum_i \beta_i G(v_i), \quad \sum \beta_i = 1, 0 \le \beta_i \le 1 \quad (4)$$

[1] We extend barycentric coordinates to $n$-sided faces, $n > 3$, by introducing a vertex in the middle [Levy 2001].

Obviously, if we simply take convex geometric combinations of points on the sphere we will produce points that lie inside of the sphere, not on it. We solve this problem by re-projecting the points back onto the sphere by casting a ray from the origin through the point to the sphere. This is called the Gnomonic mapping [Praun and Hoppe 2003] and is invertible.

We keep vertices as points on the unit sphere, $(x,y,z) : x^2+y^2+z^2 = 1$. Note that we do not need a parameterized definition of the sphere domain — the charts will provide us with a local parameterization.

### 3.3 Charts

Each element in the mesh produces a *chart*; informally, a chart takes a portion $U_c$ of $S^2$ to a portion $c$ of the plane. The term chart refers to the combination of the mapping function $\alpha_c$, $U_c$ (the domain), and $c$ (the co-domain). We will use the term chart to refer to all or one of the three, disambiguating where necessary.

Charts are constructed in a two-step process (see Figure 2). We first define a mapping $M_D$ from $U_c \subset S^2$ to the plane (stereographic projection) and then apply a second warping transformation (projective transform) $M_w : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that takes part of the plane to itself. Both of these mappings must be $C^\infty$ and invertible over the region of interest. They will not, in general, be globally invertible. The final chart mapping is then a composition of the two:

$$\alpha_c : U_c \subset S^2 \rightarrow c \subset \mathbb{R}^2 \quad = \quad M_w \circ M_D \tag{5}$$

A stereographic projection is specified by a point $P$ on the sphere around which the projection is centered. It is radially symmetric, invertible except for the point opposite the center of projection, and the distortion is minimal for small portions of the sphere. The generalized form first rotates the sphere to bring the point $P$ to the north pole, then projects the north pole to the origin, flattening out the sphere around it. A point $(Q_x, Q_y, Q_z)$ is mapped to the plane as follows:

$$\theta_0 = \tan^{-1}(P_y/P_x) \quad \phi_0 = \sin^{-1}(P_z) \tag{6}$$
$$\theta = \tan^{-1}(Q_y/Q_x) \quad \phi = \sin^{-1}(Q_z) \tag{7}$$

$$k = \frac{2}{1+\sin\phi_0\sin\phi+\cos\phi_0\cos\phi\cos(\theta-\theta_0)} \tag{8}$$
$$M_D(Q) = \big(k\big(\cos\phi sin(\theta-\theta_0)\big), k\big(\cos\phi_0\sin\phi-\sin\phi_0\cos\phi\cos(\theta-\theta_0)\big)\big) \tag{9}$$

Note that if $Q_x = Q_y = 0$ we define $\theta = 0$. The inverse $M_D^{-1}(s,t)$ is:

$$r = \sqrt{s^2+t^2} \qquad c = 2\tan^{-1}(r/2) \tag{10}$$

$$\phi = \sin^{-1}(\cos c\sin\phi_0 + (t/r)\sin c\cos\phi_0) \tag{11}$$
$$\theta = \theta_0 + \tan^{-1}\left(\frac{s\sin c}{r\cos\phi_0\cos c - t\sin\phi_0\sin c}\right) \tag{12}$$
$$M_D^{-1}(s,t) = \big(\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi\big) \tag{13}$$

The projective transform is a $3\times 3$ matrix $m$ that is invertible except for points $(x,y)$ that lie on the line given by the last row of the matrix ($m_{20}x+m_{21}y+m_{22}=0$).

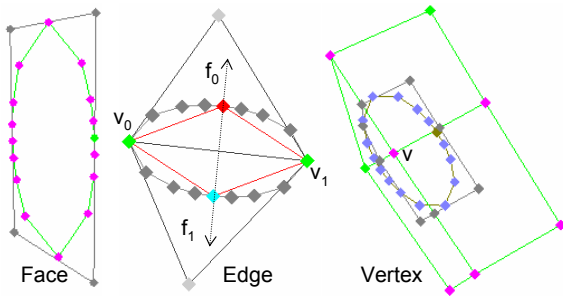


Figure 4: Defining the projective transform for each chart type. Shown are the 2D point locations of the neighboring vertices after the spherical projection, and the mapping of the unit circle (via $M_w^{-1}$) to this intermediate stage. Face: The ellipse is inscribed in the polygon formed by the face's vertices. Edge: The ellipse passes through the two vertices and extends midway into the adjacent faces. Vertex: The ellipse covers the vertex and extends midway into the adjacent faces.

$$[x,y,w]^T = m[s,t,1]^T \tag{14}$$
$$M_w(s,t) = (x/w, y/w) \tag{15}$$
$$[s,t,w]^T = m^{-1}[x,y,1]^T \tag{16}$$
$$M_w^{-1}(x,y) = (s/w, t/w) \tag{17}$$

It takes straight lines to straight lines and conics to conics so the image of the circle under $M_w^{-1}$ is an ellipse (see Figure 2). We only need to guarantee that the transform is invertible over the circle, *i.e.,* the line formed by the last row of the matrix does not intersect the circle. We use this transformation to better control the coverage of the chart. Note that if all of the charts have circular co-domains, then we can restrict $M_w$ to affine transformations; if we allow square co-domains then the projective transform allows for "keystoning".

We create a chart for each vertex, edge, and face in the sketch mesh. Recall that we have embedded the mesh in the sphere; we use the element's embedded location to determine the center point for the stereographic projection. The vertex charts use the corresponding vertex's position, the edge charts the center of the edge, and the face chart the centroid of the face. Once we have defined the stereographic projection, we can project the local neighborhood of the element to the plane, *i.e.,* the faces adjacent to the vertex, the two faces adjacent to the edge, and the vertices of the face.

We use slightly different heuristics for each of the element types to determine the best projective transformation (see Figure 4). We actually solve for the inverse of $M_w$, or the map from the unit circle to the projection of the element's local neighborhood. For the face charts, we are looking for the projection that takes the unit circle to the largest ellipse that still lies within the projected vertices of the face. For the edge charts, we are looking for a projection that places one diameter of the ellipse on the edge (so that the boundary passes through the edge's vertices) and extends the other diameter so that it covers the adjacent face centers, or crosses the face boundary, whichever comes first. Similarly, the ellipse for the vertex chart is centered on the vertex and extends out to cover the adjacent faces' centroids and adjacent edges' mid-points.

The above heuristics balance two competing goals. The first goal is to ensure that every point on $S^2$ is covered by at least one chart (preferably three). The second goal is to bound the complexity of the chart's embedding function. Each chart is fit to the corresponding part of the subdivision surface; if a chart stretches over too much of the embedded mesh then we will need a very high-order polynomial to capture the corresponding degrees of freedom.

### 3.3.1 Face charts

Let $n$ be the number of vertices of the face. We first build a unit polygon with $n$ sides so that the unit circle is inscribed in the polygon. Let $p_i$ be this polygon's vertices, and $q_i$ be the location of the projected face's vertices. We then solve a least-squares problem of the form:

$$\begin{aligned} m_{00}p_x + m_{01}p_y + m_{02} - m_{20}p_xq_x - m_{21}p_yq_x &= q_x \\ m_{10}p_x + m_{11}p_y + m_{12} - m_{20}p_xq_y - m_{21}p_yq_y &= q_y \\ &\vdots \end{aligned}$$

which has $2n$ rows. We set $m_{22} = 1$, which fixes the overall scale of matrix. If $n = 3$ we remove the projective transform, setting $m_{20} = m_{21} = 0$. Note that $n = 4$ exactly constrains the solution, up to a scale factor. If this fails to produce a non-folding projective transform (only possible with $n > 4$) we can employ the vertex optimization strategy. This has never happened in practice.

### 3.3.2 Edge charts

For the edge charts we build a four-sided polygon, with $q_0$ and $q_2$ set to the projection of the edge's two vertices. $q_1$ and $q_3$ are constructed by taking a line perpendicular to $\overline{q_0q_2}$ that passes through the mid-point of $\overline{q_0q_2}$. To figure out how far along this line to place the points, we take the minimum of the distance from the mid-point to the projected face centroids, or the intersection of the line with the projected face polygons.

If the edge is on the boundary, we use the distance from the single adjacent face.

### 3.3.3 Vertex charts

We first build a $2n$ polygon from the adjacent face's centroids and adjacent edge's mid-points, where $n$ is the number of adjacent faces. We then iteratively smooth this polygon and shift it until it is convex and its center is within some epsilon $\varepsilon$ of the projected vertex. To smooth, we move each polygon vertex closer to the average of its two neighbors. $\varepsilon$ is set to be 0.15 of the width of the polygon.

Next, we solve the same least-squares problem as we did for the face. Finally, we run a gradient descent algorithm that moves the center of the ellipse towards the projected vertex and the vertices $q_i$ outside of the ellipse (with a small weighting factor, $1/10$, to bring the ellipse boundary close to the $q_i$ to prevent excessive shrinking.

Both iterative procedures converge within 5-15 iterations.

If a vertex is on the boundary then we reflect the existing edge and face points to create a complete polygon.

## 3.4 Embedding the domain

**The embedding functions** we use are general polynomials of order $K$, and hence are $C^\infty$:

$$E_c(s,t) = \sum_{ij \subset [0,K]} a_{ij} s^i t^j \tag{18}$$

where $K = 5$ for the images in this paper. This number was arrived at experimentally, but can be justified as follows: Each chart covers roughly 16 faces of the second level subdivision mesh. Each of these faces is essentially a $C^2$ spline patch with one interval per side [Stam 1998], i.e., it is a degree three polynomial. To capture all of the degrees of freedom needed for the 16 patches would require a degree $3*4 = 12$ polynomial; however, there is substantial smoothing in the subdivision process which allows us to use a lower-degree polynomial. Implementation note: We use Horner's Rule [Borwein and Erd 1995] for evaluating the polynomial. This method is computationally stable and reduces the number of multiplications.

To solve for the $a_{ij}$, we take the standard least-squares approach ($Ax = B$). We generate a $n \times n$ grid of points over the chart (discarding those that do not fall in the unit circle). For each chart point $p$ we calculate the corresponding point $q$ on the final subdivision surface [Stam 1998]. We then solve for the polynomial that minimizes

$$\sum_i ||E_c(p_i) - q_i|| \tag{19}$$

We set $n = 10$ for the examples here.

To map from the chart point $p$ to the subdivision surface we subdivide the sketch mesh embedded in $S^2$ three times. We do not apply the geometric averaging step when subdividing in the domain, since we are not trying to smooth the domain mesh, just establish a correspondence between points in $S^2$ and the subdivision mesh. New edge vertices are placed at the midpoint of the original edge, and the new vertices placed where the old vertex was. Every point in $S^2$ can then be mapped (using barycentric coordinates) to a face in the subdivided, embedded mesh, and from there to the corresponding subdivided sketch mesh.

**The blend function** is a $C^k$ 1D radial spline surface formed by rotating a $C^k$ spline basis function with support $(-1,1)$ around the origin. The knot vector is uniformly spaced, starting at $-1$ and ending at 1 (which places the maximum value at 0). The blend functions determine the continuity of the final surface. Because the sketch mesh covers $S^2$, the charts overlap, and the blend functions are non-zero over the chart (reaching zero exactly at the boundary of the chart) the sum in the denominator of Equation 1 will be non-zero.

## 3.5 Sketch mesh

To create an initial set of charts, blend, and embedding functions we must first embed the mesh into $S^2$ [Grimm 2004; Gotsman et al. 2003; Saba et al. 2005; Praun and Hoppe 2003]. We currently split the mesh in half, embedding each half into a hemisphere. Once the mesh is embedded, we run an optimization[2] routine that moves the vertices toward the centroid of their star in $S^2$. Matching the original mesh's geometry is not important; it is more important for the embedded mesh to be as regular as possible. This prevents excessive skew in the charts.

We create the charts as described in Sections 3.3. At this step we guarantee that the charts cover the sphere; if not, we can increase the size of the charts until they do (we have not had a problem in practice). We can conservatively check the coverage using the GPU as follows. Create a mesh for each chart (see Figure 3), shrinking the meshes slightly so that their projection is one pixel from their true projection. Now render all of the charts simultaneously from the six cardinal directions; any uncovered pixel inside the center of the rendered sphere indicates a gap (we ignore the edge portion because the chart mesh faces lie slightly inside of the sphere).

# 4 Adaptive resolution editing

Here we discuss how to add detail to an existing surface $S_i$. The user specifies the detail using a detail mesh; we again construct charts, blend, and embedding functions as described earlier for this new

[2] We do not optimize the spherical case beyond a few iterations because it collapses to the zero solution [Saba 2005].

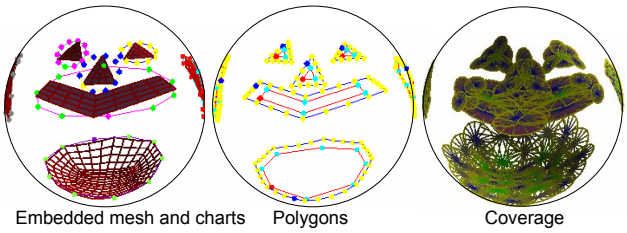


Figure 5: Left: The detail mesh embedded on the sphere. Each disjoint mesh component is covered by a chart. Middle: The boundaries of the mask function. It is one inside of the red polygon, and fades to zero by the blue polygon. Right: The corresponding charts.

mesh. To smoothly blend the new detail charts into the existing surface we define a mask function, $v_{i+1}$, which is one where we want the detail, and smoothly fades to zero outside of this area. We use this mask function in three ways. First, we use it to mask-out the blend functions of all of the patches in the existing surface $S_i$ that overlap the detail region. Second, we use it to fade-out the blend functions of the new patches. Third, we use it in the embedding fitting step (Equation 19) to blend the boundary patches' geometry into the existing surface.

We keep the detail mesh vertex locations in the coordinate frame of the surface $S_i$. When the user picks a point on $S_i$, this automatically determines a location on $S^2$ for that point. Therefore, as the user creates the detail, they simultaneously create a matching mesh embedded in the domain, and set the coordinate frame for the vertices.

We first discuss how the mask function is built from the detail mesh. Second, we define a new blend function that incorporates the mask function. Third, we define a modification to the chart fitting step; this last step is not necessary to ensure mathematical continuity, but it does increase visual smoothness in the blend region.

### 4.1 The mask function

To build the mask function we first create a chart that covers each disjoint mesh component (see Figure 5). The projection point is placed at the average of the vertices of the mesh component. We solve for the projective transform that takes the unit circle to the smallest-area ellipse that completely contains the projected mesh component. To find this transform, we use a Simplex solver [Nelder and Mead 1965] which typically converges in 100-200 iterations. The error function we use is the area of the ellipse plus a heavy (100 times the ellipse size plus the distance to the ellipse) penalty for each point outside of the ellipse.

Once we have the chart we can project the mesh component, and its second-level subdivision, into the chart. We form a polygon from the first ring of the second-level subdivision, taking every fourth vertex. This produces a polygon that is inset inside of the original boundary polygon by approximately $1/2$ the width of the original faces. The mask function is the built using the projected interior polygon by taking the convolution of it with a radial B-spline basis function of support $2r$, where $r$ is the smallest distance from the boundary of the interior polygon to the exterior one. The continuity of the mask function is the determined by the continuity of this basis function.

#### 4.1.1 The blend functions

We use $v_{i+1}$ to mask-out the blend functions of $S_i$ and to prevent the blend functions of $S_{i+1}$ from influencing the surface outside of the mask region. Label the blend functions $\hat{B}^i_c$, where $i \geq 0$ is the detail level. We modify the blend functions *before* we include them in the sum in Equation 1:

$$B^i_c \quad = \quad v_i \hat{B}^i_c \Pi_{j>i}(1 - v_j) \tag{20}$$

where we again define $\hat{B}^j_c$ to be zero outside of the domain of $c$. $v_0$ is defined to be one everywhere.

#### 4.1.2 Chart fitting

We also use the mask function when fitting charts at higher levels (Equation 19). Where the mask function is one we use the subdivision surface exclusively to find the $q_i$. Where the mask function is zero we use $S_{i-1}(p_i)$, blending in the blend region by $v_i$.

Implementation note: Although this equation (and equation 1), appear to have a large number of terms, in practice there are usually only 2 to 7 terms, corresponding to the charts that overlap at that point, which need to be evaluated. By keeping track of which charts actually overlap, we only need to check a handful of charts for inclusion in the sum or product.

## 5 Implementation Details

To enable derivatives, we use a C++ template-based differentiation approach [Stauning and Bendtsen n. d.]. In general, derivative calculation is within three to five times the speed of the original calculation. All of the normals used when rendering were calculated from the analytical derivatives.

Because the chart and blend functions do not change when the embedding functions change (Equation 1), we can cache much of this equation and only update the individual embedding functions and the final sum. This enables real-time manipulation of the surface.

To create the tessellation the user first specifies a desired edge length $l$, usually as a percentage of the average edge length in the sketch mesh. For each mesh (the sketch and any mask meshes) we take the first-level subdivision mesh (which has 4-sided faces) and determine a sampling rate for each face that produces points spaced roughly $l$ distance apart in the interior of each face, and $0.5l$ from the boundary of the face. At this point we have a set of samples which are fairly evenly spaced with respect to the *geometry* of the surface. We next use QHull to generate a convex hull of the points. This triangulation is water-tight, but is a Delauney triangulation for the sphere geometry, not our actual geometry, and tends to have long, skinny triangles.

To produce better triangles we do a combination of edge-swaps and Laplacian-style filtering. The edge-swap routine swaps any edge where the opposite diagonal is closer to the desired length $l$ (after the first iteration we calculate a new desired length by taking the average length of all of the edges). The filtering step moves a vertex towards the centroid of its neighbors *on the surface*, not in the sphere. To find the vertex's new sphere location, we find the geometric average of its neighbors, then project that average point back onto the surface (a closest-point routine). Since the surface is in a 1-1 correspondence with the sphere, this also gives us the desired sphere point. We perform a total of three edge swaps and two filtering steps, starting with the edge swap. This produces a very regular tessellation with nearly equilateral triangles.

The running time of this algorithm is dominated by the time it takes to calculate $E(p)$ for each vertex. Calculating the final mesh in Figure 1 (42568 vertices) took approximately 10 seconds.

Like splines, the interaction time during editing is proportional to the number of surface vertices influenced by the changed controls and the number of charts that need to be re-fit, not the *total* number of surface vertices. Editing one vertex of the top-level subdivision

mesh for a given level typically requires 16-30 charts to be re-fit and 100-300 surface vertices to be re-calculated, leaving aside cascaded hierarchical changes. The system response is real-time when editing 10-20 *control* vertices, with a tessellation of approximately 25 surface vertices per chart.

## 6 Remarks

The use of a subdivision mesh at each level makes it simpler to construct surfaces which are visually smooth as well as mathematically continuous. In addition to providing a target smooth surface to approximate, the mesh also helps to structure the blend functions of a given adaptive level so that they are close to being a partition of unity.

Similarly, by embedding the boundaries of the adaptive meshes in the previous level surface, we automatically create adaptive geometry that blends with the existing surface even before applying the embedding equation. The user is, of course, free to "tear" the adaptive levels off of the previous surface — the result will still be mathematically continuous, if ugly.

## 7 Conclusion

We have presented a constructive method for building manifold surfaces of spherical topology that allows charts to be created at any scale and position. The use of $S^2$ as a domain both separates the representation of the topology from the geometry of the surfaces and simplifies the construction of charts. Although in this paper we discuss only spherical topologies, the approach extends to other topologies as well [Grimm 2004; Grimm and Hughes 2003].

We provide a re-formulation of the manifold embedding function that supports hierarchical surface editing by selectively masking out blend functions at lower levels. Unlike traditional surface pasting, this approach is mathematically simple, requires no geometric constraints, and can create joins of any continuity.

There are, of course, drawbacks to this approach. It is more computationally expensive to compute than traditional splines are. It also loses some of the basis function independence qualities and convexity normally associated with spline-based techniques.

The main benefit of our approach is the ability to treat functions on the sphere in a uniform manner, as a collection of local, planar maps. By blending between functions on local maps we eliminate the need to explicitly compute constraints and ad-hoc transitions between adjacent local functions. We also eliminate the need to decide *a priori* where we will need additional charts.

## 8 Acknowledgements

Thanks to the reviewers for their helpful comments. Funded in part by NSF grant CCF 0429856 and CCF 0238062.

## References

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics 21*, 3 (July), 312–321.

BIERMANN, H., MARTIN, I. M., ZORIN, D., AND BERNARDINI, F. 2002. Sharp features on multiresolution subdivision surfaces. *Graphical Models 64*, 2 (Mar.), 61–77.

BORWEIN, P., AND ERD, T. 1995. Horner's rule. pg. 8.

CATMULL, E., AND CLARK, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design 10*, 6 (Nov.), 350–355.

CHAN, K. K. Y., MANN, S., AND BARTELS, R. 1997. World space surface pasting. In *Graphics Interface '97*, 146–154.

DEROSE, T. D., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *SIGGRAPH*, 85–94.

DOO, D., AND SABIN, M. 1978. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design 10*, 6 (Nov.), 356–360.

FORSEY, D., AND BARTELS, R. 1988. Hierarchical B-spline refinement. *Computer Graphics 22*, 2 (July), 205–212. Proceedings of SIGGRAPH '88.

GONZALEZ-OCHOA, C., AND PETERS, J. 1999. Localized-hierarchy surface splines (less). In *I3D Symposium*, 7–16.

GOTSMAN, C., GU, X., AND SHEFFER, A. 2003. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics 22*, 3 (July), 358–363.

GRIMM, C., AND HUGHES, J. 1995. Modeling surfaces of arbitrary topology using manifolds. *Computer Graphics 29*, 2 (July), 359—369.

GRIMM, C., AND HUGHES, J. 2003. Parameterizing n-holed tori. *Mathematics of Surfaces X* (Sept. 17-19th), 14–29.

GRIMM, C. 2002. Simple manifolds for surface modeling and parameterization. *Shape Modelling International* (May).

GRIMM, C. 2004. Parameterization using manifolds. *International Journal of Shape Modelling 10*, 1 (June), 51–80.

GU, X., HE, Y., AND QIN, H. 2005. Manifold splines. *Symposium on Solid and Physical Modelling* (June), 27—38.

HE, Y., GU, X., AND QIN, H. 2005. Rational spherical splines for genus zero shape modeling. *Shape Modelling International* (June), 82–91.

HOLLIG, K., AND MOGERLE, H. 1990. G-splines. *Computer Aided Geometric Design 7*, 197–207.

KHODAKOVSKY, A., AND SCHRÖDER, P. 1991. Fine level feature editing for subdivision surfaces. In *ACM Solid Modeling*, 203–211.

LEUNG, R., AND MANN, S. 2003. Distortion minimization and continuity preserving in surface pasting. *Graphics Interface* (May), 193—200.

LEVY, B. 2001. Constrained texture mapping for polygonal meshes. *Computer graphics*, 417–424.

LOOP, C., AND DEROSE, T. 1990. Generalized B-spline surfaces of arbitrary topology. *Computer Graphics 24*, 4 (Aug.), 347–357. SIGGRAPH.

LOOP, C. 1994. Smooth spline surfaces over irregular meshes. *Computer Graphics 28*, 2 (July), 303–310.

NAVAU, J. C., AND GARCIA, N. P. 2000. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design 17*, 7 (August), 643–671. ISSN 0167-8396.

NELDER, J. A., AND MEAD, R. 1965. A simplex method for function minimization. In *Computer Journal*, vol. 7, 308–313.

PETERS, J. 2002. $C^2$ free-form surfaces of degree (3,5). *Computer Aided Geometric Design 19*, 2 (Feb.), 113–126.

PETERS, J. 2004. Smoothness, fairness and the need for better multi-sided patches. *Topics in algebraic geometry and geometric modeling 334*, 197–207.

PRAUN, E., AND HOPPE, H. 2003. Spherical parameterization and remeshing. *ACM Transactions on Graphics 22*, 3 (July), 340–349.

PULLI, K., AND LOUNSBERY, M. 1997. Hierarchical editing and rendering of subdivision surfaces. Tech. Rep. TR-97-04-07.

SABA, S., YAVNEH, I., GOTSMAN, C., AND SHEFFER, A. 2005. Practical spherical embedding of manifold triangle meshes. *Shape Modelling International* (June), 256–265.

SABA, S. 2005. Barycentric spherical parameterization of genus-0 3d meshes. *MSc Thesis, Technion*.

SEDERBERG, T. W., CARDON, D. L., FINNIGAN, G. T., NORTH, N. S., ZHENG, J., AND LYCHE, T. 2004. T-spline simplification and local refinement. *ACM Transactions on Graphics 23*, 3 (Aug.), 276–283.

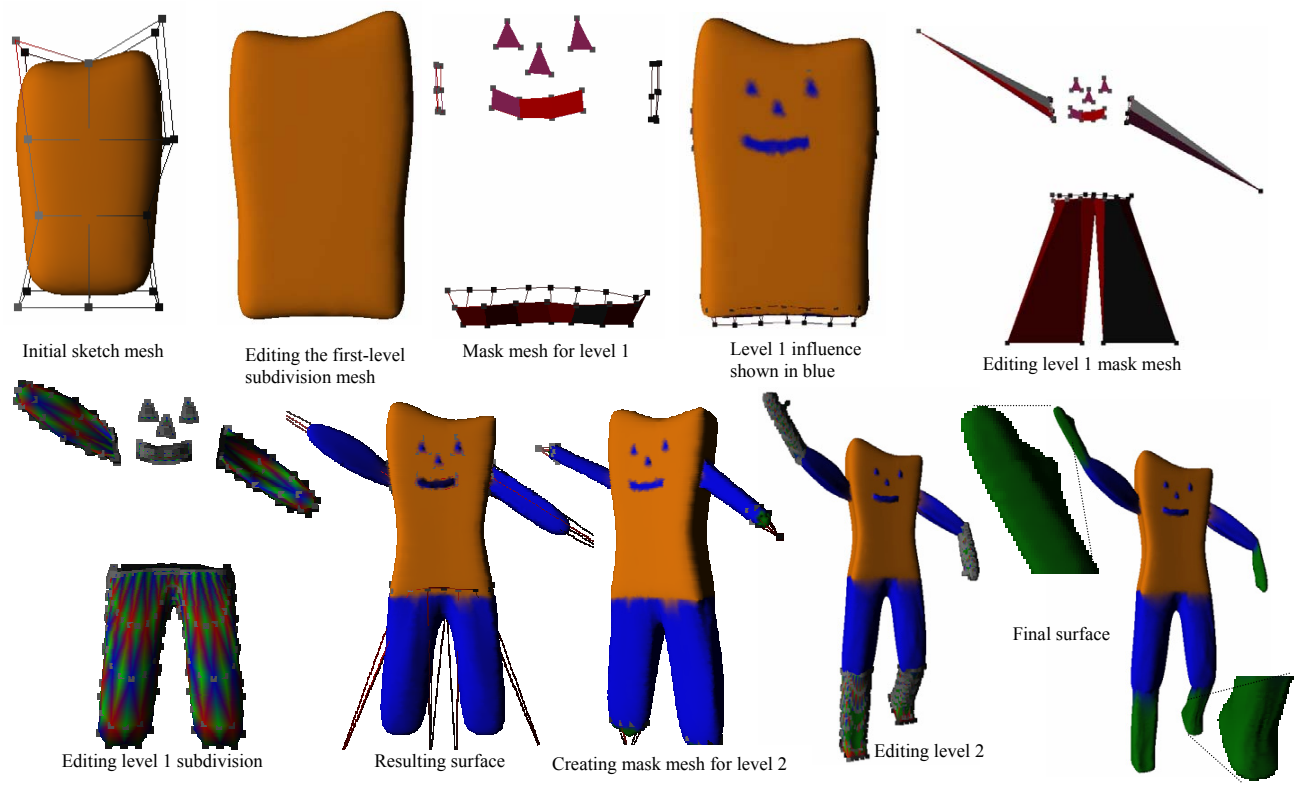


Figure 6: The entire modeling sequence.


STAM, J. 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Computer Graphics 32*, 395–404.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics 22*, 3 (July), 724–731.

STAUNING, O., AND BENDTSEN, C. Fad: http://www.imm.dtu.dk/nag/proj_km/fadbad/.

WARREN, J. 1992. Creating multisided rational bézier surfaces using base points. *ACM Transactions on Graphics 11*, 2 (Apr.), 127–139.

YING, L., AND ZORIN, D. 2004. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Transactions on Graphics 23*, 3 (Aug.), 271–275.

ZHENG, J. J. 2001. The n-sided control point surfaces without twist constraints. *Computer Aided Geometric Design 18*, 2 (Mar.), 129–134.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. *Computer Graphics 31*, 259–268.