Modeling Surfaces of Arbitrary Topology using Manifolds[1]

Cindy M. Grimm
cmg@cs.brown.edu (401) 863-7693

John F. Hughes
jfh@cs.brown.edu (401) 863-7638

The Science and Technology Center
for Computer Graphics and Scientific Visualization

ABSTRACT

We describe an extension of B-splines to surfaces of arbitrary topology, including arbitrary boundaries. The technique inherits many of the properties of B-splines: local control, a compact representation, and guaranteed continuity of arbitrary degree. The surface is specified using a polyhedral control mesh instead of a rectangular one; the resulting surface approximates the polyhedral mesh much as a B-spline approximates its rectangular control mesh. Like a B-spline, the surface is a single, continuous object. This is achieved by modeling the domain of the surface with a manifold whose topology matches that of the polyhedral mesh, then embedding this domain into 3-space using a basis-function/control-point formulation. We provide a constructive approach to building a manifold.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling, Curve, Surface, Solid, and Object Representations, Splines

1 Introduction

Surfaces of arbitrary topology are currently attracting a good deal of attention. While spline surfaces have proven a powerful modeling tool [BBB87] [Far88], modeling topologically arbitrary surfaces with them is hard, because they require a rectangular parameterization. To create complex surfaces, especially free-form ones such as Figure 17, we need a surface model which is computationally inexpensive and yet capable of modeling arbitrary topologies. Ideally, this surface model retains the power of spline surfaces: a compact representation, guaranteed continuity, and flexibility. This paper presents such a method of surface modeling.

Our approach differs from previous techniques in that the surface is constructed from pieces of surface which overlap substantially instead of abutting only along their edges. Mathematicians have studied such surfaces for many years [MS74] [ST67] using the technology of *manifolds*. Although the underlying mathematics is somewhat complicated, manifolds have advantages that make this complexity worthwhile.

[1] This work was supported in part by grants from NSF, ARPA, IBM, NCR, Sun Microsystems, DEC, HP, and ONR grant N00014-91-J-4052, ARPA order 8225.

Consider, for example, the problem of mapping textures onto two adjacent patches in a conventional spline surface. Matching the texture along the boundary between the patches may be difficult, since there is no common parameterization between the patches. With a manifold, however, the "edge" of one texture region is well within the adjacent texture region, so that the two textures can easily be blended.

Similarly, if one tries to make a smooth path on a surface, and the path crosses a patch boundary, maintaining smoothness of the path and its derivatives may be difficult. But there are no boundaries on a manifold because as a path gets near the edge of a patch, it has already entered the adjacent patch, and its derivatives can be computed in that new patch's coordinate system. This can be used to make various forms of user-interaction (sliding one object along another, for example) much smoother. Similarly, differential equations such as those used to generate reaction-diffusion textures can be solved by blending partial solutions across patch overlaps.

This paper begins with a survey of previous and related work, then sketches a high-level view of the surface construction technique. This is followed by a discussion of manifolds in general and how to build a manifold for a given surface. Next we discuss adding geometry to the manifold. Finally, we conclude with results and future work.

2 Previous work

Several different approaches to arbitrary-topology surface models have been suggested. Subdivision [CC78] [Loo87] produces a smooth surface by repeatedly subdividing a polyhedral mesh and in the limit yields a G^1 surface. This method is very general, but does not admit an analytical form (although recent work [HDK93] has made subdivision more tractable). Another approach is to "fill in" any non-rectangular parts of a mesh with n-sided patches [HM90] [LD89]. This is analogous to Bézier surfaces, in that it ensures continuity across the boundaries of patches by maintaining constraints on control points. A similar technique is to produce a collection of triangular (and possibly rectangular) elements from an initial mesh and stitch them together into a surface using the geometric information in the original mesh [Loo94]. In [WW94], the initial sketch is a set of contours over which a triangulated surface is stretched, using variational modeling techniques [WW92] to control the shape of the surface.

Unlike the previous methods, our approach produces a surface which is one continuous piece and hence does not require constraints to maintain continuity. Adding to (or removing from) the surface is similar to adding or removing a row of control points from a B-spline surface – continuity is automatically guaranteed.

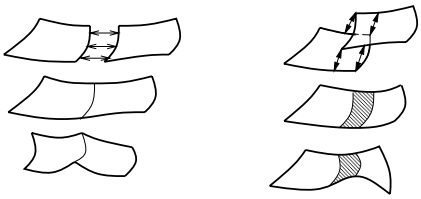

Figure 1: Left: Gluing two patches together along their thin edge then bending the patches along the crease. Right: Gluing two patches together along a region then bending the patches together.

3 Overview

Spline patches are a powerful modeling tool but stitching them together into complex surfaces has proven difficult. As an analogy, consider building surfaces out of stretchy pieces of fabric that can be "glued" to each other. The pieces of fabric are spline patches, and the glue consists of mathematical operations such as control point constraints. Previous methods have focused on gluing these fabric pieces together by applying glue thinly along the abutting edges of the fabric pieces (see left of Figure 1). The problem with this technique is that a change to one of the fabric pieces is not reflected in the adjoining patch except along the glued edge. The smoothness of the joint is maintained by adjusting the adjoining patch afterwards.

Our approach is to apply glue to the top of one fabric piece and the bottom of another piece and then glue them together by *overlapping* the two pieces. Now, when the first piece is stretched or moved, the second, overlapping piece follows naturally with it (see right of Figure 1). This eliminates the need to re-establish the continuity of the join after every change to the surface. In the curve domain, this is the difference between Bézier curves and B-spline curves; Bézier curves are joined together into larger curves by constraining the control points at the end of one curve and at the beginning of the following curve. B-splines, on the other hand, are extended by adding in another overlapping curve segment. We prefer the B-spline approach because the domain is continuous and no constraints are required; to extend B-splines to arbitrary topologies, however, we first need a mechanism for adding overlapping pieces to a surface.

We begin by taking several pieces of fabric and gluing them together into a larger object by overlapping them. To describe the object, we need to describe the pieces of fabric and how they overlap. This is very similar to the familiar concept of an atlas of the world; each page of the atlas is rectangular (i.e., a piece of fabric) but the collection of pages describes a spherical object, the world. The pages of the atlas overlap enough to get from one page to the next. For example, the page for France contains part of Spain, and the page for Spain contains part of France. When traveling from France to Spain there is a time when one is located on both pages; the two maps may not be identical where they overlap but there is enough information to establish a correspondence between the two pages.

With an atlas we begin with an object, the world, and create a set of pages that cover the world, with each page overlapping with its neighbors. Suppose we did not have the world, but instead had just the pages of an atlas. We could put the pages onto stretchy pieces of fabric and glue them together using the information on the overlapping parts. This glued-together object is then a "world". This is a *constructive* approach to building a world as opposed to an analytical one. Because we do not have a world (i.e., the surface) *a priori*, we use this constructive approach to building a surface out of pieces of surface.

There is one more consideration; when we build our world from the pages of the atlas, how do we know what the world looks like?

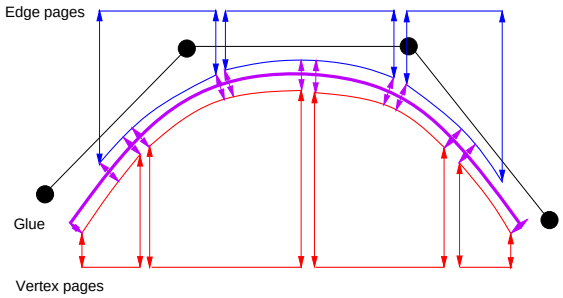

Figure 2: Stretching the pages of the atlas out to approximate the polygon, then gluing them together

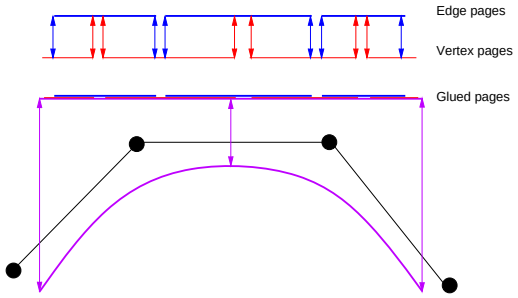

Figure 3: Glue the pages of the atlas together, then stretch them out to approximate the polygon.

The pages and their overlaps provide information on the *topology* of the object but no information on the *geometry* of it. With a real atlas we have some implicit knowledge of what the world looks like, but this knowledge is external to the atlas. There are two possible ways to add geometrical information into the atlas; consider the case of making an atlas for a curve. The rough shape of the desired curve is given by a control polygon. For each vertex and edge of the polygon we create have a page in the atlas. A page corresponding to an edge of the polygon overlaps with the two vertex pages corresponding to the vertices of the edge. The curve is built by gluing the vertex and edge pages together and adding geometrical information to describe what the curve looks like. This can be accomplished in two ways:

- First describe what each page looks like, then glue the pages together. This corresponds to taking the pages of the atlas, stretching them out to approximate the control polygon, then gluing them together (see Figure 2).
- First glue the pages together, then describe where they go. This corresponds to gluing the pages of the atlas together, then stretching them out to approximate the control polygon (see Figure 3).

We take the second approach because it is simpler. In the first approach, the gluing stage must be repeated every time the geometry of the surface changes, i.e., when the control polygon is moved. In the second approach, the gluing process is performed exactly once, and is independent of the particular geometry (but not topology) of the object.

Although this construction process is excessive for defining a curve, imagine constructing a surface from a polyhedron. Building an atlas provides us with a local description of the surface that is continuous and upon which we can navigate, i.e., perform operations such as calculating geodesics. The surface is relatively simple to describe locally but we can still perform global operations because the atlas pages overlap, allowing us to easily move from one

area of the surface to another. In contrast, the traditional method of stitching patches together is to abut them, resulting in joins between the patches that must be dealt with separately.

4 Outline of the construction process

To build our surface we begin with a polyhedral mesh (created by the user) that describes the basic shape and topology of the surface. This is in analogy with a B-spline control mesh, except that the polyhedral mesh is not limited to a rectangular topology. We formally define this mesh in Section 5.

Next we define the pages of the atlas and how those pages overlap. We create one page for each *element* in the polyhedron, i.e., one page for each vertex, edge, and face. How the pages overlap is determined by the adjacency relationships in the polyhedron. For example, a face page overlaps with the pages for the vertices and edges of the face. Figure 20 shows a sample polyhedron and a surface colored by page type. In Section 6 we formally define an atlas and show how to construct an atlas using the polyhedron as a guide.

Finally, we add geometry ("shape") to the atlas. We do this in a manner similar to the one used for B-splines. On each page we build several basis functions and associate a control point with each function. This tells us where the middle part of each page goes; because the pages overlap, the location of the edges of the page will be influenced by the control points of the overlapping pages. This is covered in Section 8.

5 The polyhedron

Construction of a surface starts with a polyhedral "sketch." To simplify later steps, we require that every interior vertex have exactly four faces adjacent to it, i.e., vertices are of valence four. A polyhedron of this form can be constructed from an arbitrary polyhedron by taking the dual of the first subdivision [CC78][Kin77] (see Appendix D).

The polyhedral sketch must satisfy some technical restrictions that essentially say it "looks like" a surface: every vertex must be an end of some edge, every edge must be an edge of some face, at most two faces can meet at an edge, each *interior* vertex must have 4 edges and 4 faces adjacent to it, and each *boundary* vertex must have n edges and $n-1$ faces adjacent to it. Furthermore, the polyhedron must be *orientable*, i.e., it must contain no embedded Mobius strips. These technical restrictions make the polyhedron an "oriented surface" in the sense of algebraic topology [Spa66]. Finally, we require that each face have 3, 4, 5, or 6 edges.

The polyhedral sketch contains three types of information: the *geometric* information given by the locations of the polyhedron vertices; the *local topological* information as given by the "incidence" relations—which vertices lie on which edges, which edges are in which faces; and the *global topological* information that can be derived from this local data: the number of components or pieces of the sketch, the number of boundary components, and the genus.

6 The atlas, or manifold

We have informally described an object consisting of pieces of fabric "glued together". This concept is called a *manifold*. Manifolds were introduced in the 1890s and formalized in the 1920s in order to describe objects whose topology was more complex than that of Euclidean space. The notion was that an object "locally like" Euclidean space could be studied in much the same way as Euclidean space. In one view, a manifold is a structure imposed on a set – a division of the set into overlapping regions, each of which is in correspondence with a portion of the Euclidean plane. Consider a world atlas. Every point on the world can be found in at least one page in the atlas and sometimes in several. A path from one point to another can be found by tracing a line through the pages. Where the path must cross from one page to another, the two pages overlap enough that one can locate oneself on the second page. The individual pages are regions of $\Re^2$ but taken together they represent a sphere [MYV93]. There are also implicitly defined *transition functions* from one page to another. These are the "glue" we use to glue the pages together; they establish a correspondence between the region of one page and a region of another. Thus Brussels and its environs may appear on two different atlas pages: the page for Benelux countries, and also in the upper right corner of the page for France. The labels for Brussels and the surrounding towns, etc., establish a correspondence between the upper right corner of the France page and the lower left corner of the Benelux page.

6.1 Formal definition

In the traditional definition of a manifold the object exists and the manifold consists of *charts*, or mappings from the object to pieces of $\Re^n$. (The images of the charts are our atlas pages. From now on, we will refer to the atlas pages as charts.) This is an analytical view; because we do not have an object but are building it we depart from this view and define a *constructive* view of manifolds. Our constructive definition of a manifold starts with charts and information on how they overlap (the charts and the transition functions). We call this a *proto-manifold*. From the proto-manifold we build a manifold using an equivalence relation, i.e., we glue the charts together using "this place on this chart is the same as that place on that other chart. In [Gri] we show that this definition is equivalent to the traditional one.

Definition 1 A C^k differentiable proto-manifold K of dimension n *consists of:*

1. *A finite set A of connected open sets in $\Re^n$. A is called a* proto-atlas. *Each element $c \in A$ is called a* chart.
2. *A set of subsets $U_{ij} \subset c_i$, where c_i and c_j are charts in A and where $U_{ii} = c_i$.*
3. *A set of functions Ψ called* transition functions. *A transition function $\psi_{ij} \in \Psi$ is a map $\psi_{ij} : U_{ij} \to U_{ji}$ where $U_{ij} \subset c_i$ and $U_{ji} \subset c_j$. Note that U_{ij} and U_{ji} may well be empty. The following conditions on ψ_{ij} must hold:*
   - *(a) ψ_{ij} is $1-1$, onto, and C^k-differentiable*
   - *(b) $\psi_{ij}^{-1} = \psi_{ji}$*
   - *(c) $\psi_{ii}(x) = x, x \in c_i$*
   - *(d) The "cocycle condition": $(\psi_{ij} \circ \psi_{jk})(x) = \psi_{ik}(x)$ for $x \in U_{ik} \bigcap U_{ij}$ (see Figure 4)*

The charts $c \in A$ are the pages of the atlas. The subsets U_{ij} describe what part of chart i overlaps with chart j. The function ψ_{ij} defines the exact correspondence between points in U_{ij} and points in U_{ji}.

Next we build the manifold. If $K = (A, \Psi)$ is a proto-manifold then there is a relation $\sim$ defined on $\sqcup_{c \in A} c$ (where $\sqcup$ denotes disjoint union) such that if $x \in c_i$, $y \in c_j$, then $x \sim y$ iff $\psi_{ij}(x) = y$. Conditions (1)–(3) in Definition 1 ensure that $\sim$ is an equivalence relation [Gri].

Continuing the analogy, each chart c is a page of the world atlas, each transition function ψ_{ij} is a correspondence between parts of two charts, and the equivalence relation $\sim$ says that "the place labeled Brussels on page 93 is the same as the place labeled Brussels on page 24."

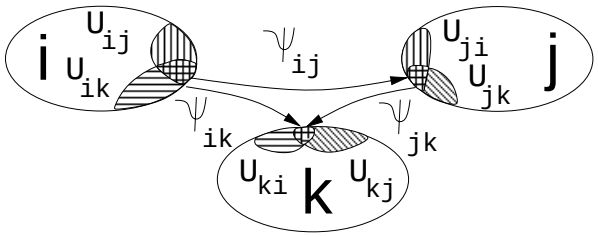

Figure 4: Three overlapping charts. The cocycle condition requires that $(\psi_{ij} \circ \psi_{jk})(x) = \psi_{ik}(x)$ for $x \in U_{ik} \bigcap U_{ij}$.

The relation $\sim$ lets us build a single object from the charts. If a point x in one chart is taken via ψ_{ij} to a point y in another chart ($\psi_{ij}(x) = y$) then those two points are a single point on the final object.

Definition 2 *Let $\sim$ be the equivalence relation described above and K a proto-manifold as defined above. Define M as the quotient of $\sqcup_{c \in A} c$ by $\sim$. Let Π be the map taking $x \in \sqcup_{c \in A} c$ to $[x] \in M$, where $[x]$ is the equivalence class of x.*

In [Gri] we prove under weak conditions satisfied in the construction below that M is a Hausdorff space upon which we can construct a traditional manifold structure. The correspondence between our definition and the standard one is relatively simple: the image of a chart c under the map Π is a subset $\Pi(c) = D_c \subset M$; the restriction of Π to c defines a one-to-one correspondence

$$\gamma_c : c \to D_c \subset M, \qquad \gamma_c(x) = \Pi(x)$$

between c and D_c. The inverse of γ_c is a map α_c from a subset D_c of M to c which is a subset of $\Re^2$, i.e.,

$$\alpha_c : D_c \to c \subset \Re^2, \qquad \alpha_c(x) = \gamma_c^{-1}(x)$$

The maps α_c are the "coordinate charts" in the standard definition of a manifold structure on the set M [Spi70]. The map γ_c is called a *local parameterization* in [MS74] and a *coordinate system* in [Ste74].

Note that the name "chart" refers to one of our subsets of $\Re^2$ and that α and α^{-1} denote the correspondence between these charts and subsets of the manifold.

6.2 Building a manifold from a polyhedron

In the following discussion we construct a manifold without boundary, i.e., the polyhedron has no boundary vertices. Extending this construction to a manifold with boundary is straightforward, as explained in Section 7.

We use the topological structure of the polyhedron as a guide for building the manifold. We construct one chart for each element in the polyhedron and define the overlaps of the charts by the adjacency relationships in the polyhedron. This produces three sets of charts: the *vertex charts* — those charts corresponding to the vertices of the polyhedron — the *edge charts*, and the *face charts*. We denote these by $\mathbf{V} = \{\text{chart for } v\}_{v \in \mathcal{V}}$, $\mathbf{E}$, and $\mathbf{F}$, respectively. The entire proto-atlas A is then $\mathbf{V} \bigcup \mathbf{E} \bigcup \mathbf{F}$.

Figure 20 shows a polyhedron and the resulting surface, which has been colored according to chart type. The vertex charts ($\mathbf{V}$) are in red. Each vertex chart has eight other charts overlapping it; four edge charts (in green) and four face charts (in purple).

To keep the notation simple, henceforth V indicates the chart associated with a vertex v and V' the chart for a vertex named v', and similarly for edges and faces.

A chart in one set never overlaps with a different chart in that same set. A chart *does* overlap with those charts that are "nearby" in the polyhedron. For example, a face chart only overlaps with the vertex and edge charts corresponding to the vertices and edges of the face. In summary:

1. $U_{VV'} = \emptyset, V \neq V'$ (and similarly for $U_{EE'}$ and $U_{FF'}$).
2. $U_{VE} \neq \emptyset$ iff $v \in e$.
3. $U_{VF} \neq \emptyset$ iff $v \in f$.
4. $U_{EF} \neq \emptyset$ iff e is an edge of f.

If we perform a similar construction in 2 dimensions then we have one chart for each vertex and one for each edge (see Figure 3). An edge chart E overlaps with two vertex charts V and V', where $e = \{v, v'\}$. In this case the charts are all segments of the real line; note how each edge chart is "nearly" covered by the two neighboring vertex charts. By "nearly" we mean the chart E is covered by the closure of U_{EV} and $U_{EV'}$, i.e.,

$$E = \overline{U_{EV} \bigcup U_{EV'}}.$$

To duplicate this in 3D we ensure that the edge and face charts are "nearly" covered by the overlapping vertex charts (see Figures 6 and 7).

In the 2D example, the interior vertex charts are also covered by the two overlapping edge charts, i.e.,

$$V = \overline{U_{VE} \bigcup U_{VE'}}.$$

We would also like to cover the vertex and face charts in this way but this turned out to be too restrictive. We do, however, require that the charts overlap as much as possible.

6.2.1 The charts

A chart is a connected, open subset of $\Re^2$. An overlap region is an open subset of the chart.

The vertex charts are unit squares centered at the origin. A vertex chart overlaps with four faces (the four quadrants) and four edges (Figure 5 shows the vertex chart and the overlap regions). A vertex-edge overlap region U_{VE} overlaps with the two regions U_{VF_0} and U_{VF_1} where f_0 and f_1 are the two faces adjacent to e.

The edge charts are diamonds with the left and right ends chopped off (see Figure 6). An edge chart overlaps with two vertex charts (the left and right half of the diamond) and two face charts (the upper and lower half of the diamond).

The face chart for an n-sided face is an n-sided regular polygon centered at the origin (slightly smaller then a unit polygon). The edge charts overlap the edges of the polygon, while the vertex charts overlap the corners. The region U_{FE} overlaps the two regions U_{FV_0} and U_{FV_1}, where v_0 and v_1 are the endpoints of e.

The details of the charts and their overlaps are given in Appendix B.

6.2.2 The transition functions

The transition functions are the glue that holds the charts together. We have described what parts of the charts to glue together but not the exact correspondence between them.

Transition functions must meet two requirements: they must be C^k, and the cocycle condition must hold (see Figure 4). Additionally, we would like the functions to be as close to the identity function as possible. By this we mean that a transformed (via $\psi_{cc'}$) image on $U_{c'c}$ should look as much like the original image on $U_{cc'}$ as possible.

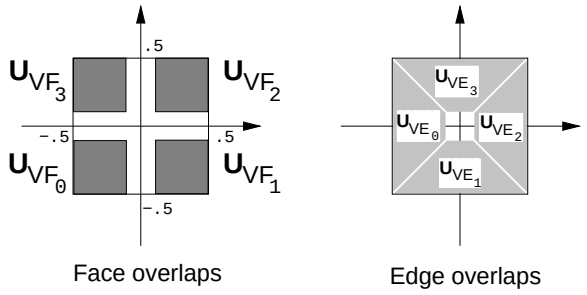

Figure 5: A vertex chart. Left: the four regions U_{VF_i} Right: the four regions U_{VE_i}
.

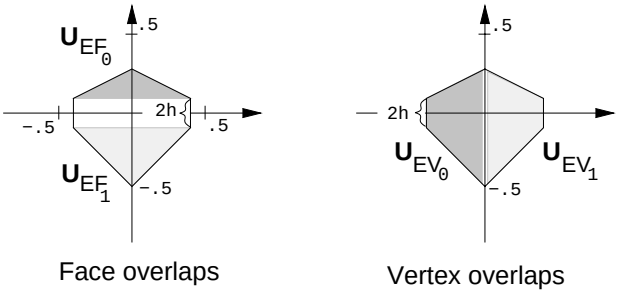

Figure 6: An edge chart. Left: the two regions U_{EF_i} (F_0 is 3-sided, F_1 is 4-sided). Right: the two regions U_{EV_i}
.

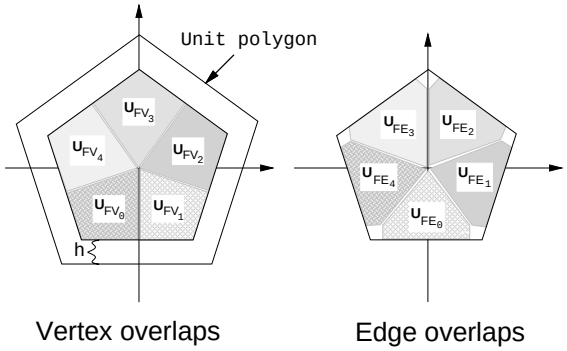

Figure 7: A face chart. Left: the n regions U_{FV_i} Right: the n regions U_{FE_i}

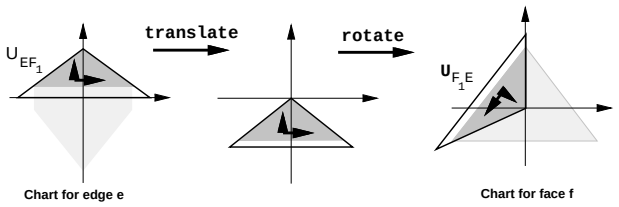

Figure 8: Mapping from an edge chart to a face chart by translating then rotating.

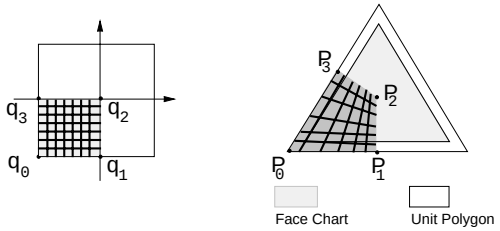

Figure 9: Mapping from a face chart to a vertex chart using a projective transformation.

Charts within the set $\mathbf{V}$ (and similarly $\mathbf{E}$ and $\mathbf{F}$) do not overlap. Thus the transition function $\psi_{VV'}$ is either the empty function (if $V \neq V'$) or the identity (if $V = V'$), and similarly for edge-edge and face-face transition functions.

This means we need only consider the transition functions between charts in distinct chart sets, e.g. vertex-to-face transitions. If we define the function $\psi_{CC'}$ then the function $\psi_{C'C}$ is defined as its inverse. Thus the transition functions we define can be divided into three categories: edge-to-face, vertex-to-face, and vertex-to-edge.

Satisfying the cocycle condition is a matter of ensuring that the function taking the edge chart to the vertex chart directly is the same as the combination of functions taking an edge chart to a face chart to a vertex chart (see Figure 10). We first define the edge-to-face and vertex-to-face functions, and then define the edge-to-vertex functions as compositions of the other two.

The edge-to-face transition function is a plane isometry; the region U_{EF} is simply translated and rotated (see Figure 8). The vertex-to-face transition function takes the quadrant U_{VF} and "stretches" it using a projective transformation to fit it into the corner of the face chart (see Figure 9).

The edge-to-vertex transition function is built as a blend of two functions which are compositions. Examining Figure 10, on the top half of the diamond the function must be the composition of the edge-to-face and face-to-vertex functions for the top face, while on the bottom half of the diamond the function must be the corresponding composition of the bottom face functions. These two composed functions will not, in general, agree along the $x-$axis. To fix this, we have left a *gap* between the two composed functions; this lets us *blend* between the two composed functions in the gap. (This is the reason the face charts are slightly smaller than the unit polygon – to give us room to do the blend.) Care must be taken to ensure that this function is 1-1, onto, and C^k; details of this (and the other functions) are given in Appendix C.

6.3 The manifold

A formal proof that these charts and transition functions form a proto-manifold (Definition 1) appears in [Gri]. Informally, the charts are defined to be open sets in $\Re^2$ and the transition functions are defined to be 1–1, onto, and C^k. The cocycle condition is satisfied because in the only non-trivial case, where three charts overlap (one each of

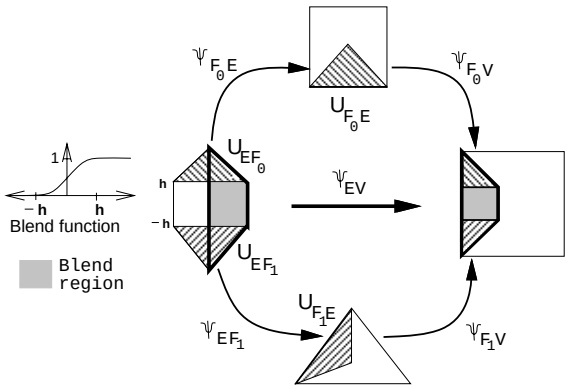

Figure 10: Mapping from an edge chart to a vertex chart using the upper and lower composition functions.

a vertex, edge, and face chart), we have defined the functions to be compositions of the others.

7 Manifolds with boundary

To extend our definition to a manifold with boundary we allow the charts of the proto-manifold to be half-balls in $\Re^n$. We use a manifold with boundary when the polyhedral sketch has a boundary, i.e., when an edge has a single adjacent face. Since a boundary never occurs in the middle of a face chart, we need only alter our descriptions of vertex and edge charts. An edge chart E corresponding to a boundary edge is simply a triangle joined with the $x-$axis. A vertex chart V corresponding to a boundary vertex is a contiguous subset of the quadrants of the unit square, with some part of the axes included (e.g., a vertex chart corresponding to a corner vertex would consist of a single quadrant).

The definitions of the transition functions remain unchanged except for the edge-vertex functions, since the overlap regions remain unchanged. The edge-vertex function becomes just the single composed function one one-half of the edge chart. The α_c functions also remain the same.

8 Adding geometry to the manifold

We now have a collection of stretchy fabric pieces glued together but with no geometric structure (they are just "collapsed on the floor in a pile"). Rather than describe what the entire object looks like all at once, we just describe what the middle of each chart looks like. Because of the way the charts overlap, this will determine the geometry of the entire manifold.

To define the geometry we use a basis-function control-point formulation. The basis functions $\{B_s : M \to \Re\}_{s \in S}$ are a finite collection S of local, C^k functions that sum to one everywhere; they are analogous to traditional B-spline basis functions. The basis function B_s determines how much the control point $G_s \in \Re^3$ influences the surface Q at a given point:

$$Q(p) = \sum_{s \in S} G_s B_s(p)$$

We next describe how to build the basis functions.

8.1 Basis functions

To build the basis functions we first define a set of *proto-basis functions* on the chart and associate a control point with each proto-basis function. Specifically, we have a set $\{\hat{b}_s\}_{s \in S}$ of C^k functions

$$\hat{b}_s : c \to \Re$$

where $s = \{c, i\}$, $c \in A$ and $i \in 1 \ldots n_c$. The number n_c is dependent upon the desired continuity k and the type of chart (vertex, edge, or face). We define these functions in Section 8.1.1; for now, suffice it to say they are C^k and go to 0 by the boundary of c and are similar to a tensor-product basis function.

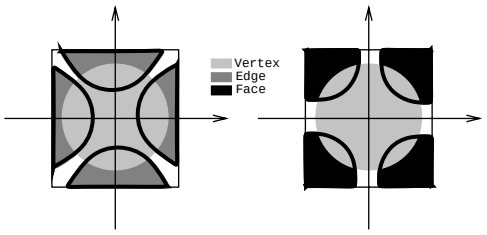

Figure 11: The center part of overlapping charts in a vertex chart.

The $\hat{b}$ functions are defined on the individual charts and hence do not interact with functions on other charts. We next extend the $\hat{b}$ functions to the manifold where they will interact. Imagine building piles of sand on each page of the atlas. When the pages of the atlas are glued together the piles of sand are no longer on a single page but possibly on several pages glued together, each page of which may have its own piles of sand. Formally these piles of sand are built by extending the proto-basis functions to the manifold by using the α_c functions where they are defined and 0 where they are not. Define $\hat{B}_s : M \to \Re$ by

$$\hat{B}_s(p) = \begin{cases} \hat{b}_s(\alpha_c(p)) & \text{if } p \in \alpha_c^{-1}(c) \\ 0 & \text{otherwise} \end{cases}$$

These functions are C^k and local. For the surface Q to behave like a spline surface, we require basis functions B_s that meet three properties that traditional basis functions satisfy:

- B_s is C^k for some k.
- B_s is local (its support lies in a single chart).
- $\sum_{s \in S} B_s(p) = 1$ for all $p \in M$.

The last step is to normalize the $\hat{B}_s$ functions to ensure that they sum to one. If for every point $p \in M$ we have $\sum_{s \in S} \hat{B}_s(p) \neq 0$ then the definition $B_s : M \to \Re$,

$$B_s(p) = \frac{\hat{B}_s(p)}{\sum_{s' \in S} \hat{B}_{s'}(p)}$$

is valid. Figure 11 shows how the center parts of edge and face charts overlap a vertex chart. Recall that the vertex charts "nearly" cover every chart and hence the manifold; to ensure that the above definition is valid, we make certain that the supports of the proto-basis functions cover the center area of the chart in which they are defined. As shown in Figure 11, this ensures that the vertex charts are covered by the supports of the $\{\hat{B}_s\}_{s \in S}$ functions and hence that the manifold is covered by them as well.

8.1.1 The proto-basis functions

We now show how to build the proto-basis functions using a tensor-product B-spline and a projective transformation. For each proto-basis function $\hat{b}_{c,i}(r)$ we start with a quadrilateral Q_i in its chart c

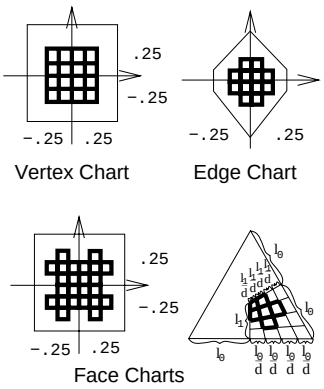

Figure 12: The quadrilaterals Q_i for the proto-basis functions with grid division 4.

(see Figure 12) and construct a projective transformation ϕ_{Q_i} (see Appendix A) from the quadrilateral to the unit square. Let $\beta : \Re^2 \rightarrow \Re$ be a C^k tensor-product B-spline whose support is from $0-(k/2)$ to $1+k/2$. Then the ith proto-basis function is

$$\hat{b}_i(r) = \beta(\phi_{Q_i}(r))$$

The quadrilaterals for the face chart are formed by mapping a subdivided unit square into U_{FV} via a projective transform. If ϕ_{FV} is a projective transform from U_{FV} to the unit square then the four corners of Q_0 are ϕ_{FV}^{-1} of $(0,0)$, $(0,1/d)$, $(1/d,1/d)$, and $(1/d,0)$.

The choice of d depends on k. The size and number of the quadrilaterals (and the size of the support of the tensor-product B-spline) are chosen so that the supports of the proto-basis functions cover as much of the chart as possible without falling out of it. For the C^2 pictures we used a grid division of 4.

8.2 The control points

The location of the control points is completely unconstrained; however, the user has already provided a rough sketch of the shape of the surface (the polyhedron). We provide an initial placement of the control points based on the subdivision surface of the polyhedron.

We describe how to assign values to the control points using the Catmull-Clark subdivision surface ($\mathcal{L}$) of the polyhedron. This produces a surface with the "feel" of a B-spline surface (note, though, that the choice of values does not affect the continuity). We define a mapping from the manifold to the subdivision surface $\mathcal{H} : M \rightarrow \mathcal{L}$ and assign the function G_s the value $\mathcal{H}(p_s)$, where p_s is the center of support for the basis function B_s.

To define $\mathcal{H}$, we first note that after one level of subdivision we have one subdivision point l_c for each chart c in M. We relate the origins of the charts to the subdivision points by $\mathcal{H}(\alpha_c(0,0)) = l_c$. This places the subdivision points in a chart V as shown on the left of Figure 13.

After one level of subdivision every face in the subdivision surface is 4-sided; these faces are mapped to the quadrants of the vertex charts. Further subdivisions "grids" the vertex chart as shown in the right of Figure 13. We define $\mathcal{H}$ by assigning the grid points $(\alpha_V^{-1}(\text{grid point}))$ to their corresponding points in the subdivision surface. Eventually, this relates a set of points that are dense in M to the subdivision points.[1] The function for $\mathcal{H}$ on this dense set can be extended to M in a natural way.

[1] We assign the points along the boundary of the vertex charts $\alpha_V(V)$ to their adjacent points in M.

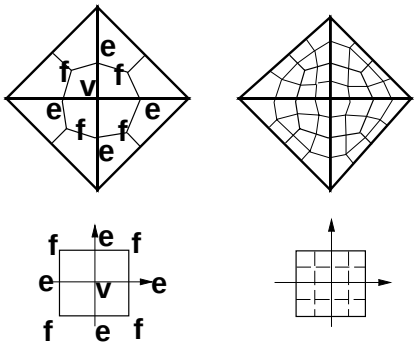

Figure 13: An abstraction of the subdivision process on a vertex chart. The polyhedron has been subdivided twice.

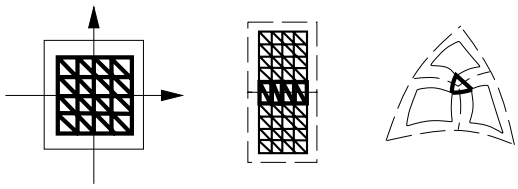

Figure 14: Left: Triangulating the interior of a vertex chart. Middle: Adjoining edges. Right: Filling in the corners.

8.3 Triangulating the embedding

There is a tradeoff between the number of triangles in the triangulation and how closely it approximates the surface. The triangulation presented here produces approximately $(2r)(2r)N_v$ triangles for a given resolution r. If the control points are evenly spaced then the resulting triangulation is also evenly spaced.

We triangulate the domain by first triangulating the vertex charts as shown on the left of Figure 14, where r determines the number of squares. To fill in the remaining gaps, we adjoin the triangles along the boundaries to the triangles of neighboring vertex charts using a strip of triangles. The corners are filled in with an appropriate n-sided triangulation (see Figure 14).

9 Remarks and future work

Images 17–22 show some example surfaces, most of which were created using an interactive editor. The coloring of Image 18 was determined by running a reaction-diffusion simulation on the manifold and using the resulting chemical concentrations to create the stripes [Tur91]. Because the embedding defines a metric on the manifold, we can use either that global metric or the local (chart) metric with the reaction-diffusion equations.

This surface technique produces aesthetically pleasing models fairly efficiently and easily. It is also suitable for data fitting because the topology of the surface can be made to fit the topology of the data, bringing the surface fairly close to the data initially. Additionally, continuity constraints need not be maintained while fitting the surface to the data.

Although this technique shares many of the properties of traditional splines, some techniques have yet to be developed, such as the equivalent of knot insertion and the Oslo algorithm. Although we can use subdivision to produce a more finely controllable surface similar in shape to the original surface, this surface is not necessarily identical to the original. Note that if the polyhedral mesh is rectangular then the resulting surface reduces to a B-spline surface.

10 Implementation

We have implemented a simple interactive polyhedral editor to create the surfaces shown here. The user builds an arbitrary polyhedron P by creating vertices and connecting them together into faces. The system automatically creates a second polyhedral model C which is the dual of the first subdivision surface of P. (Figure 16 shows both P and C for the flower model.) The user is free to move the vertices of C. The editor runs in real time on an HP-735 with surfaces of continuity C^2 and a triangulation level of 4.

11 Acknowledgments

We would like to thank our sponsors for providing the support for this research, Dan Robbins for help with the video, Michael Kowalski for the model of Alexander's two-holed torus, and Tony DeRose, Jean Schweitzer, Marc Levoy, and Jules Bloomenthal for their helpful comments.

A Projective Transformations

Let $B_1 = (1,0,0)^t$, $B_2 = (0,1,0)^t$, $B_3 = (0,0,1)^t$, and $B_4 = (1,1,1)^t$ and let $P_1 \ldots P_4$ be the (column vectors of) homogeneous coordinates of four points of the plane, no three colinear. There is a matrix T_P such that $T_P(B_i)$ is a non-zero multiple of P_i. Here is the construction of T_P: Let M_P be the 3×3 matrix whose columns are P_1, P_2, and P_3. Let $\Lambda = M_P^{-1} P_4$. Letting λ_i ($i = 1, 2, 3$) denote the entries of Λ, T_P is the matrix whose columns are $\lambda_1 P_1$, $\lambda_2 P_2$ and $\lambda_3 P_3$.

To find a projective transformation on the plane taking any set of four points $\{P_i\}$, no three colinear, to any other such set $\{Q_i\}$, compute the matrix $K = T_Q T_P^{-1}$. Multiplying K by the vector $(x, y, 1)^t$ gives a vector $(X(x,y), Y(x,y), W(x,y))$. The projective transformation we seek is just $(x, y) \mapsto (\frac{X(x,y)}{W(x,y)}, \frac{Y(x,y)}{W(x,y)})$.

B Charts

We describe the exact shape of each chart (a subset of $\Re^2$) and the overlap regions $U_{cc'}$.

Each vertex chart is a unit square centered at the origin (see Figure 5). A vertex chart overlaps four face charts (corresponding to the four faces having v as a vertex) and four edge charts (corresponding to the four edges having v as a vertex). The U_{Vc} are defined to be $\varphi_{cV}(U_{cV})$. If U_{VF_i} and U_{VE_j} overlap then e must be an edge of f.

The face chart for an n-sided face is an n-sided regular polygon centered at the origin. The size of the polygon is chosen so that the perpendicular distance from the edge of the face chart to the edge of the unit polygon containing it is a constant h (see Figure 7). Typically, h is small. For the figures in this paper a value of .1 was used. A *wedge* of a polygon is the triangle whose vertices are the center of the polygon and the two ends of one side of the polygon.

An n-sided face chart overlaps with n vertex charts (the n corners of F) and n edge charts (the n edges of F). The U_{FV_i} are bounded by lines drawn from the centroid of the polygon to the midpoints of the polygon edges. The U_{FE_i} are the parts of the chopped-off "wedges" mentioned above that actually lie in the chart F. If $U_{FV} \bigcap U_{FE} \neq \emptyset$ then v is a vertex of e.

The edge chart is a diamond with its left and right ends chopped off. The diamond consists of two triangles, one congruent to a wedge of a unit polygon with the same number of sides as each of the overlapping face charts. The triangles are joined along the sides that correspond to the edge, and that side is placed along the x-axis.

An edge chart overlaps two face charts and two vertex charts (the left and right sides of E). If $f_0 = \{\ldots, v, v', \ldots\}$ and $f_1 = \{\ldots, v', v, \ldots\}$ then U_{EF_0} is in the upper half plane, U_{EV} is on the left, and $U_{EV'}$ is on the right.

C Transition functions

The face-to-edge function is a rigid motion (see Figure 8). Let d_n be the height of the wedge and θ be the amount of rotation:

$$\varphi_{FE}(s,t) = \{s\cos\theta - t\sin\theta, t\cos\theta + s\sin\theta + d_n\}$$

The face-to-vertex transition function uses the unique projective transformation taking any four points in the plane, no three of which are collinear, to any other four such points (see Appendix A). Let ϕ_{PQ} denote the projective map that takes a corner of a unit polygon P containing a face chart F to a quadrant Q of a vertex chart V (see Figure 9). The function φ_{FV} is simply the restriction of ϕ_{PQ} to U_{FV} and φ_{VF} is ϕ_{PQ}^{-1} restricted to U_{VF}.

The edge-to-vertex transition is defined in terms of the other transition functions. We define φ_{EV} to be $\varphi_{F_0V} \circ \varphi_{EF_0}$ on U_{EF_0} and $\varphi_{F_1V} \circ \varphi_{EF_1}$ on U_{EF_1} so that the cocycle condition is satisfied (see Figure 10). To complete our definition, we need only define φ_{EV} on the no-man's land, i.e., we must produce a smooth blend between the two composite functions on the region $U_{EV} - (U_{EF_0} \bigcup U_{EF_1})$. Recall that the regions U_{EF_i} are each at a distance h from the x-axis (this displacement by h was included precisely to permit us do this blend). The choice of h will affect the embedding function defined in Section 8, but does not affect the discussion here.

To define φ_{EV} we first extend the domains of the functions φ_{EF_i} and φ_{F_iV}. φ_{EF_0} is linear, so it extends to all of $\Re^2$; similarly for φ_{EF_1}. Now we extend the domains of the functions φ_{F_iV} to the region $\varphi_{EF_0}(U_{EV} - U_{EF_1})$, i.e., the domain of the φ_{EF_0} plus the no-man's land. The singularities of φ_{VF_0} lie on a line that does not intersect $\varphi_{EF_0}(U_{EV} - U_{EF_1})$ [Gri]. Therefore the composite function can be extended to the region $U_{EV} - U_{EF_1}$. A similar argument holds for φ_{EF_1}. Using these extended functions, we define φ_{EV} by

$$\begin{aligned}\varphi_{EV}(x,y) &= \eta(y)\varphi_{EF_0} \circ \varphi_{F_0V}(x,y) \\ &\quad +(1-\eta(y))\varphi_{EF_1} \circ \varphi_{F_1V}(x,y)\end{aligned}$$

where $\eta : \Re \to \Re$ is a blend function which is C^k, and is 1 to the right of h and 0 to the left of $-h$. In [Gri] we show that φ_{EV} is invertible and 1–1 on U_{VE}.

D Dual of first subdivision surface

Given a polyhedron P, the first subdivision surface P' of P contains a vertex for every vertex, edge, and face of P (see Figure 15). All of the faces of P' have exactly 4 sides [Kin77]. Taking the dual of this produces a polyhedron C with vertices of valence 4. Figure 16 shows the original polyhedron P in light blue, and the dual surface C in green. The vertices of C are initially placed at the centroids of the faces of P'.

REFERENCES

[BBB87] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[CC78] E. Catmull and J. Clark. Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design*, 10(6):350–355, November 1978.

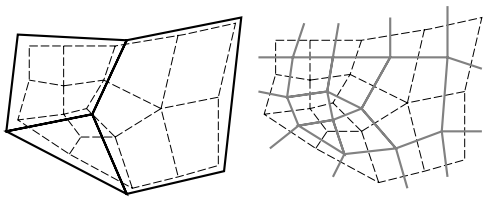

Figure 15: An example of taking the dual of the first subdivsion surface of an arbitrary polygon. The original polygon is shown in black, the first subdivision surface in dashed lines, and the dual in light grey.

[Far88] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1988.

[Gri] C. Grimm. *Surfaces of Arbitrary Topology using Manifolds*. PhD thesis, Brown University (in progress).

[HDK93] M. Halstead, T. DeRose, and M. Kass. Efficient, Fair Interpolation Using Catmull-Clark Surfaces. *Computer Graphics*, 27(2):35–44, July 1993. Proceedings of SIGGRAPH '93 (Los Angeles).

[HM90] K. Hollig and H. Mogerle. G-splines. *Computer Aided Geometric Design*, 7:197–207, 1990.

[Kin77] P. King. On Local Combinatorial Pontrjagin Numbers (I). *Topology*, 16:99–106, 1977.

[LD89] C. Loop and T. DeRose. A Multisided Generalization of Bézier Surfaces. *ACM TOG*, 8(3):204–234, July 1989.

[Loo87] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, 1987.

[Loo94] C. Loop. Smooth Spline Surfaces Over Irregular Meshes. *Computer Graphics*, 28(2):303–310, July 1994. Proceedings of SIGGRAPH '94.

[MS74] J. Milnor and J. Stasheff. *Annals of Mathematical Studies*. Princeton University Press, Princeton, New Jersey, 1974.

[MYV93] J. Maillot, H. Yahia, and A. Verroust. Interactive Texture Mapping. *Computer Graphics*, 27(4):27–35, July 1993. Proceedings of SIGGRAPH '93.

[Spa66] E. H. Spanier. *Algebraic Topology*. McGraw-Hill Inc., New York, New York, 1966.

[Spi70] M. Spivak. *Differential Geometry Volume 1*. Publish or Perish Inc., 1970.

[ST67] I.M. Singer and J. A. Thorpe. *Lecture Notes on Elementary Topology and Geometry*. Scott, Foresman and Company, Glenview, Illinois, 1967.

[Ste74] N. Steenrod. *The Topology of Fibre Bundles*. Princeton University Press, Princeton, New Jersey, 1974.

[Tur91] G. Turk. Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. *Computer Graphics*, 25(2):289–298, July 1991. Proceedings of SIGGRAPH '91 (Las Vegas).

[WW92] W. Welch and A. Witkin. Variational Surface Modeling. *Computer Graphics*, 22(2):157–166, July 1992. Proceedings of SIGGRAPH '92.

[WW94] W. Welch and A. Witkin. Free Form Shape Design Using Triangulated Surfaces. *Computer Graphics*, 28(2):247–256, July 1994. Proceedings of SIGGRAPH '94.

Figure 16: The polyhedral sketch for the flower. The light blue polygon (drawn in wire-frame) is built by the user. The green polygon is the dual of the first subdivision surface of the light blue polygon and is constructed automatically. The locations of the vertices of the green polygon have been adjusted to produce finer detail.

Figure 17: A two-holed torus in the shape of a flower.

Figure 18: The coloring is achieved by running a reaction-diffusion system on the domain of the surface (the process was halted when partially finished for aesthetic reasons).

Figure 19: Alexander's two-holed torus (in the shape of a mug).

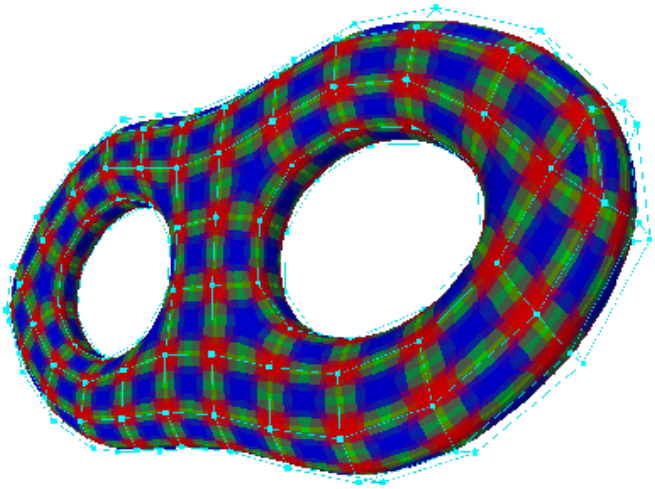

Figure 20: A two-holed torus colored by atlas page type (vertex, edge, or face).

Figure 21: A laser-scanned image of a ceramic bunny, courtesy of Stanford University.

Figure 22: An approximation of the laser-scanned bunny.