Free-Form Shape Design Using Triangulated Surfaces

William Welch and Andrew Witkin
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Keywords: fair surface design, functional minimization, polygonal models, Delaunay triangulation, adaptive meshing

Abstract

We present an approach to modeling with truly mutable yet completely controllable free-form surfaces of arbitrary topology. Surfaces may be pinned down at points and along curves, cut up and smoothly welded back together, and faired and reshaped in the large. This style of control is formulated as a constrained shape optimization, with minimization of squared principal curvatures yielding graceful shapes that are free of the parameterization worries accompanying many patch-based approaches. Triangulated point sets are used to approximate these smooth variational surfaces, bridging the gap between patch-based and particle-based representations. Automatic refinement, mesh smoothing, and re-triangulation maintain a good computational mesh as the surface shape evolves, and give sample points and surface features much of the freedom to slide around in the surface that oriented particles enjoy. The resulting surface triangulations are constructed and maintained in real time.

1 Introduction

One of the fundamental goals in computer-aided free-form shape design is to offer convenient ways to specify shape and topology. We are concerned here with a truly broad class of surfaces: smooth, doubly curved surfaces, of arbitrary topology (closed or bordered). This generality is what makes representing and controlling such shapes on a computer a difficult problem.

1.1 Functional minimization for shape design

Optimization has long been used as a way of describing fair free-form shapes (a good survey is Moreton[23]). More recently, it has come to be used in interactive modelers[4,5,41,18]. Though such approaches are computationally complex, their intent is to create an illusion of simplicity for the designer. Ideally, the designer sees a surface having no particular fixed controls or other representation-specific parameters. Instead, the surface can be directly manipulated, pinned down at points and along curves, and will behave as if made of some infinitely stretchy material. This lets us mimic a style of pen-and-paper design in which important contours of a shape are sketched out as "character lines", with the understanding that a surface passes through them in a fair way[4]. Such shapes are ultimately realized as solutions to constrained functional minimization problems — globally fair surfaces that satisfy geometric interpolation constraints. This approach allows concise descriptions of a useful class of free-form shapes.

Unfortunately, these approaches have only allowed a designer to interact with pre-fabricated families of shapes, in which the topology remains fixed and surfaces do not stray far from their initial configurations. More drastic, nonuniform deformations are not handled well by the linearized thin plate functional[30] used to fair the piecewise smooth patches making up these surfaces. Further, no real consideration has been given to the problem of creating non-trivial smooth surface topologies interactively. The one approach to fair shape design that has allowed large-scale changes in shape and topology during sculpting is the oriented particle system of Szeliski and Tonnesin[34]. The drawback of this approach is that there is no *explicit* control over surface topology — because there is no actual surface. A surface triangulation can be imposed on the particles strictly as an output, but this has no influence on the particles' subsequent behavior, and no persistence across sculpting operations.

1.2 Our approach

In this paper we continue with the agenda set forth in [41] and develop an approach to modeling truly mutable yet completely controllable free-form surfaces.

Our basic approach to shape control will use character lines and curve and surface fairing. To this we add:

- the ability to cut up surfaces and paste them back together smoothly, without topological restriction, thus building up complex topologies incrementally.
- the ability to seamlessly incorporate familiar shape control tools (e.g., generalized cylinders) into more complex faired surface models, using local shape-copying.
- a fairing functional based on geometric surface properties, which yields graceful shapes in the face of large-scale changes to surface shape and topology.

As with previous work, the shape design problem will be cast in terms of functional minimization: the desired shape will be the one that satisfies various geometric constraints while optimizing a measure of surface quality. Since explicit functional solutions to these minimization problems cannot generally be found, we will construct approximate solutions using an explicit surface representation.

In choosing a representation, we bridge the gap between the patch-based and particle-based approaches by adding just enough structure to a particle system to unambiguously fix its topology. To this end, our chosen representation is a set of sample points in three-space, triangulated to yield a 2D manifold topology. Although the resulting surface approximations will be faceted rather than smooth, we never lose sight of the fact that the implicitly defined variational[1] surface is the "real" surface. A designer interacts only with these approximate renderings, but always with the understanding that operations will be interpreted as implicitly defining an ideal variational

[1]We use "variational" in its mathematical sense as the solution of a problem in calculus of variations[7]. Unfortunately, this clashes with a different usage common in design literature.

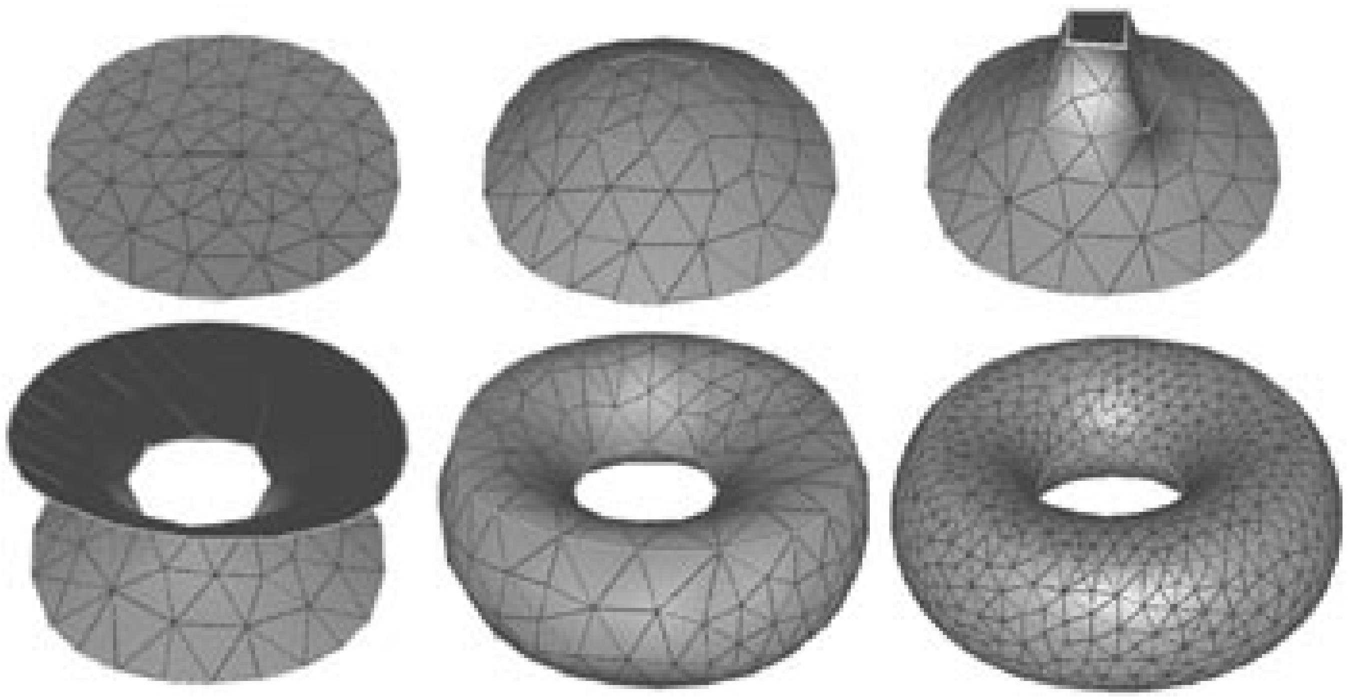

Figure 1: Making a Torus: 1. A closed curve is skinned to make a disc. 2. another closed curve is drawn on disc and elevated (this will control the inner torus shape). 3. A hole is cut in center of disc and the new boundary curve elevated. 4. The upper curve is expanded to match the lower. 5. Two boundary curves are skinned to make a single toroidal surface passing through the three control curves (200 nodes). 6. Refinement (1300 nodes. All surface meshes are rendered with Gouraud-shaded triangular facets.)

surface. Such shapes can thus be unambiguously specified and controlled through direct manipulation, regardless of the coarseness of their explicit approximations. If an explicit smooth surface is needed for any reason, it is a straightforward task to fit one to the triangulated surface in a post-processing step[24].

1.3 Outline of this paper

After a brief overview of our approach to variational sculpting and its underlying machinery, the bulk of the paper (sections 3–5) develops the mathematics and algorithms needed to construct approximations to these variational surfaces. In section 6 we discuss some specific higher-level shape tools and operations formulated within this variational framework. Section 7 concludes with a discussion of the work's contributions and some suggested directions for future research.

2 Overview

2.1 Triangulated surface representation

This work develops an approach to interactively sculpting with variational free-form curves and surfaces. Because such shapes generally cannot be explicitly computed, we are concerned with approximating these surface shapes at interactive speeds.

Our representation for the topology and approximate shape of a variational surface is a set of sample points in 3D with an associated surface triangulation (we will often refer to this simply as a mesh). These meshes are represented in our modeler as collections of nodes with radially ordered neighbor relations. Curves are approximated as piecewise-linear (PWL) sequences of nodes joined by edges. We will sometimes need to operate on an embedded curve in a surface, such as a boundary or control curve. In this case the nodes and edges making up the curve must be contained in the mesh as well.

No assumptions are made about the three-dimensional shape of the mesh. In particular, there are no flatness assumptions, and we will not detect or prevent self-intersection. However, the triangulation itself must completely determine the topology of the surface that is being approximated. Regardless of the coarseness of the sampling, a triangular facet in the mesh should always correspond to a continuous triangular piece of the approximated surface, and a closed PWL boundary curve in the mesh should exist for each boundary curve in the smooth surface.

2.2 Approximating surface shapes

A fair amount of computational machinery goes towards maintaining variational shape approximations interactively (in spite of our initial minimalist intentions). We break the approximation problem into a number of reasonably simple pieces, most of them transcending our choice of a point-wise surface (as opposed to, say, smooth triangular patches) as the approximating representation. At the lowest level, local surface reconstruction lets us compute over a mesh as if it were a smooth surface, by estimating surface derivatives at sample points. On top of this is built a surface fairing scheme that minimizes squared principal curvatures. Because an even distribution of samples over a surface improves the results of such fairing computations, we apply techniques from numerical grid generation to keep samples well-dispersed. Finally, automatic refinement and retriangulation processes adapt the mesh density and connectivity as surface shape and area changes, and give sample points and local features much of the freedom to slide around within a surface that oriented particles enjoy. In order to fix some of the ideas, consider the simple construction sequence illustrated in Figure 1, which demonstrates point and curve skinning, automatic shape fairing over wide ranges of deformation, interactive changes to topology, and automatic mesh refinement. Other aspects of our approach — parameterized shaping tools and smooth surface surgery, can be seen in Figures 6 and 7.

In the following sections we detail a collection of robust procedures (some of them new) for controlling the local shape and topol-

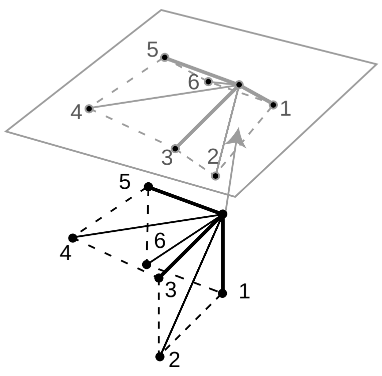

Figure 2: A failed neighborhood projection.

ogy of a mesh. Many schemes for operating on and computing over such unstructured meshes are made with reference to a separate planar domain — either a global parameterization, or a local projection of part of the mesh onto a plane. Such approaches will not be directly useful to us. On the one hand, a well-known result from differential topology implies that we cannot hope to find global parameterizations for surfaces of arbitrary genus[2]. On the other hand, local planar projection operations can fail if the mesh is not sufficiently flat (or an unfortunate choice of projection plane is made), for then the projection may scramble the radial order of neighbors about a node (Figure 2). Our modeler must behave robustly in such configurations, always honoring the given neighborhood topology at each node.

3 Local shape reconstruction

Fundamental to our approach is the ability to operate on the mesh as a sampling of a smooth surface. To do so, in addition to sample point positions we also need to be able to evaluate surface derivatives at each of these points. This will be done by temporarily fitting a smooth surface to the neighborhood around a sample point.

Because we will ultimately control sample point positions indirectly, by solving for positions that yield desired surface properties, there will be great advantage to having such derivatives be linear functions of the sample point positions. This rules out direct geometric constructions[21] as well as algebraic fitting methods[28]. Algebraic methods have the added drawback that they do not allow us to incorporate topological constraints like radial neighbor ordering into the fit.

Instead, following a standard approach to constructing finite difference stencils over a computational field[12], we will fit a truncated Taylor series expansion at a neighborhood around a point. This approach yields the desired linearity, but it comes at the expense of having to construct a separate bivariate parameterization over which to evaluate the polynomial basis functions. We'll make the most of this by constructing the parameterization in such a way that it gives the fairing functional (Section 4) a particularly simple form.

3.1 Neighborhood parameterizations

Much of the theory of curves and surfaces is developed in terms of curves parameterized by arc-length[25]. Geometric quantities, such as curvature, have simple expressions with respect an arc-length parameterization. The geometric fairing functionals of the next section will involve differentiation with respect to arc-length,

[2]the Poincaré-Hopf Index theorem on the existence of smooth vector-fields over manifolds[15], which says, informally, that you cannot comb the hair on a ball without leaving a crown somewhere.

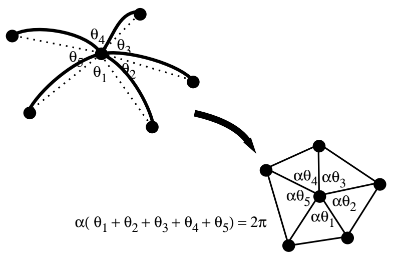

Figure 3: Surface neighborhood parameterization: angles between neighbors are measured in 3D, then scaled to sum to 2π.

so we want to use arc-length parameterizations in our local reconstructions to simplify these computations.

For surfaces, there is no corresponding notion of an arc-length parameterization; but there is a construction from differential geometry, the *geodesic polar map*[25], which serves our needs in much the same way. This is simply a generalization to geometric surfaces of polar coordinates (r, θ) in the plane. A map is constructed for a neighborhood about a given point on the surface, in such a way that a unit-speed geodesic path is associated with each tangent direction (θ). Directional derivatives computed at the point in any tangent direction are thus taken with respect to an arc-length parameterization.

In fitting a parametric curve to a sequence of scattered points, it is necessary to first assign parametric coordinates to these points. Prior to the fit, we cannot assign these coordinates based on arc-length because we have no curve. So instead, we will make a chord-length approximation to an arc-length parameterization[9,8], in which the parametric interval between two samples is taken to be the 3D Euclidean distance between them.

In constructing a neighborhood parameterization for surface fitting, it is common to project neighboring vertices in 3-space onto, e.g., the node's tangent plane. As was pointed out earlier, this is a hazardous operation. The surface triangulation induces an ordering on neighbors about each node and this order should be reflected in the parameterization, but there is no guarantee that projection will do so. Instead of projecting onto a plane, we measure the angular separation between each neighbor *in 3D*, and then uniformly scale these angles so that they sum to 2π for nodes in the surface interior and π for nodes on the boundary (Figure 3). Using a chord-length approximation to arc-length, the Euclidean distance from the center node to each of its neighbors is taken as the radial parametric separation, mimicking the geodesic polar parameterization.

3.2 Local coordinate fitting

Given a suitable parameterization of the neighborhood, it is a straightforward task to fit a truncated Taylor series expansion for each of the surface *x*, *y*, and *z* coordinate functions about the neighborhood center. In the following, $\mathbf{p}_0$ will designate the position of the node at the neighborhood center, $\mathbf{p}_1 \ldots \mathbf{p}_n$ the positions of $\mathbf{p}_0$'s n neighbors, and $(u_0, v_0) \ldots (u_n, v_n)$ their parametric coordinates.

For each of the coordinate functions we seek the coefficients of a biquadratic:

$$\begin{aligned} s(u,v) &= c_0 + c_1\,u + c_2\,v + \frac{c_3}{2}u^2 + c_4\,uv + \frac{c_5}{2}v^2 \qquad (1) \\ &= \mathbf{b}(u,v)\mathbf{c}, \qquad (2) \end{aligned}$$

where $\mathbf{b}(u,v)$ is the basis row vector $[1, u, v, \frac{1}{2}u^2, uv, \frac{1}{2}v^2]$, and $\mathbf{c}$ a column vector of coefficients. We want $s(u_0,v_0) \equiv s(0,0) = p_0$; this requires $c_0 = p_0$ (here we take p_i to mean one of p_{ix}, p_{iy}, p_{iz}, since the same fitting procedure applies to each).The remaining coefficients will be determined such that the $s(u_i, v_i)$ are a least-squares fit to the p_i.

Shifting the origin to p_0 yields the vector of shifted neighbor positions $\mathbf{q} = [p_1 - p_0, ..., p_n - p_0]^T$. The sample matrix $\mathbf{S}$ for this shifted, center-constrained system is built by evaluating the basis vector $\mathbf{b}(u_i, v_i)$ for each of the neighbors and collecting these rows into a matrix, then deleting its first column. Then $\mathbf{S}[c_1, ..., c_5]^T = \mathbf{q}$, and the least-squares solution for $c_1...c_5$ is

$$\begin{aligned} [c_1, ..., c_5]^T &= [\mathbf{S}^T\mathbf{S}]^{-1}\mathbf{S}^T\mathbf{q} && (3) \\ &= \mathbf{Zq}. && (4) \end{aligned}$$

To put this in a more convenient form as a linear function of the $\mathbf{p}_i$, let $\mathbf{z}$ represent the vector sum of the columns of $\mathbf{Z}$ and $\mathbf{P}$ represent the $n \times 3$ matrix of the $\mathbf{p}_i$'s x, y, z coordinates. The vector form for the reconstructed surface positions $\mathbf{s}(u, v)$ is then

$$\begin{aligned} \mathbf{s}(u, v) &= \mathbf{b}(u, v) \begin{bmatrix} 1 & 0 \\ -\mathbf{z} & \mathbf{Z} \end{bmatrix} \mathbf{P} && (5) \\ &= \mathbf{b}(u, v)\mathbf{BP}. && (6) \end{aligned}$$

This fitting procedure is somewhat complicated by the irregular degrees of the triangulated nodes since it requires that every node have at least five neighbors (a radial ordering for nodes beyond the immediate neighbors is not determined by the triangulation, so we do not search beyond the neighborhood to bring in additional samples as is common in least-squares schemes[13]). Worse, even though a node may have five neighbors they may be positioned parametrically so as to make the full biquadratic fit ill-conditioned.

In this case, a reduced polynomial basis function is used. For each node (of sufficient degree), an initial fit of the full basis $[1, u, v, \frac{1}{2}u^2, uv, \frac{1}{2}v^2]$ is attempted. The condition number of this fit, $c = \|\mathbf{S}^T\mathbf{S}\| \cdot \|(\mathbf{S}^T\mathbf{S})^{-1}\|$, is then computed (where the matrix norms are Frobenius norms[14]).

If the fit was ill-conditioned (say, $c > 1000$), or if there were too few neighbors, a fit is attempted with the reduced basis functions $[1, u, v, \frac{1}{2}(u^2+v^2)]$ for interior nodes[3]. Boundary nodes, which will rarely have enough neighbors for a full fit, are treated specially: the parameterization is constructed so that the two boundary nodes lie on the $\pm u$ axis, and the basis functions $[1, u, v, \frac{1}{2}u^2]$ are used. This lets a surface curve along its boundary while remaining flat in the infield direction. As before, the condition number is evaluated and an ill-conditioned fit rejected. As a last resort, a planar fit (for boundary or infield nodes) is made with the basis functions $[1, u, v]$.

A shortcoming of this approach is that the somewhat arbitrary choice of basis functions could lead to instability over time (though we have not observed this, a neighborhood that is nearly well-conditioned might switch back and forth between different sets of basis functions). Better would be to consistently use the same set of basis functions and optimize some auxiliary norm in the underdetermined case, as in Barth [1]. This requires an orthogonal decomposition of $\mathbf{S}^T\mathbf{S}$, an added computational expense that degrades the overall interactivity of the modeler.

3.3 Curve reconstruction

Smooth neighborhoods for PWL curve approximations are computed analogously to the fitting procedure for surfaces. A chord-length parameterization is used, along with the basis functions $[1, u, \frac{1}{2}u^2]$.

4 Surface fairing

In this section we formulate constrained fairing for triangulated surfaces. Surfaces will seek shapes that minimize a global measure of curvature, subject to the requirement that they interpolate specified control points and curves.

4.1 Smooth surface objective

We take as the surface fairing objective function the integral of the squared principal curvatures over a smooth surface[25]:

$$E = \int_S (\kappa_1^2 + \kappa_2^2) dA, \qquad (7)$$

where dA is the differential area form. Lott and Pullin [20] used this for surface fairing because of its relationship to the strain energy of a thin elastic plate. Unlike the more commonly used linearized thin plate approximation[4,41,35,18], this formulation does not create shape artifacts related to an underlying fixed surface parameterization. E is a geometric quantity whose definition is independent of parameterization.

Recall from the differential geometry of surfaces that information about the curvature of a surface at a point is given by the second fundamental form[32]. The normal section curvature of a surface $\mathbf{s}(u, v)$ in the direction of a parametric unit tangent $\mathbf{t} = [t_u, t_v]$ is given by $\kappa = II(\mathbf{t}, \mathbf{t})$, where

$$II(\mathbf{t}, \mathbf{t}) = \mathbf{t}^T \begin{bmatrix} \mathbf{s}_{uu} \cdot \mathbf{n} & \mathbf{s}_{uv} \cdot \mathbf{n} \\ \mathbf{s}_{vu} \cdot \mathbf{n} & \mathbf{s}_{vv} \cdot \mathbf{n} \end{bmatrix} \mathbf{t}, \qquad (8)$$

$\mathbf{n}$ is the surface unit normal, and subscripts indicate partial differentiation with respect to arc-length. It is straightforward to show that the squared Frobenius norm of the matrix is equivalent to $\kappa_1^2 + \kappa_2^2$. This lets us reformulate the objective function E in terms of the surface derivatives and normals appearing in these matrix elements:

$$E = \int_S ((\mathbf{s}_{uu} \cdot \mathbf{n})^2 + 2(\mathbf{s}_{uv} \cdot \mathbf{n})^2 + (\mathbf{s}_{vv} \cdot \mathbf{n})^2) dA. \qquad (9)$$

4.2 Triangulated surface objective

For a triangulated surface this integral is approximated as an area-weighted sum of integrands over sample points:

$$E = \sum_{\text{nodes}} ((\mathbf{s}_{uu}(0,0) \cdot \mathbf{n})^2 + 2(\mathbf{s}_{uv}(0,0) \cdot \mathbf{n})^2 + (\mathbf{s}_{vv}(0,0) \cdot \mathbf{n})^2) a, \qquad (10)$$

where $\mathbf{s}(0, 0)$ is the local surface function (6) evaluated at its node, and a the node's associated area (nominally, $1/3$ the area of each of the triangles in its parametric neighborhood). Because the neighborhood parameterization was constructed so that directional derivatives are computed with respect to an approximate arc-length parameterization, the partial derivatives above are simply taken with respect to the local u and v. The assumption here is that these parameterizations are being continually updated as the surface shape changes; whether or not this is a good way to approximate differentiation with respect to arc-length (to be characterized in[40]), the resulting objective function is geometric in nature, so that our surface fairing does not exhibit parameterization artifacts.

Substituting (6) and evaluating the derivatives of the basis functions $\mathbf{b}(0, 0)$ leads to a particularly simple form for E:

$$E = \sum_{\text{nodes}} ((\mathbf{B}_3\mathbf{P} \cdot \mathbf{n})^2 + 2(\mathbf{B}_4\mathbf{P} \cdot \mathbf{n})^2 + (\mathbf{B}_3\mathbf{P} \cdot \mathbf{n})^2) a \qquad (11)$$

where $\mathbf{B}_j$ is the jth row of the ith neighborhood basis matrix.

The dependency of E on the node positions is given by its gradient with respect to $\mathbf{P}$. To make the minimization of E tractable, we take the $\mathbf{n}$ and a to be constant when computing this gradient, and will refer to this modified objective as $\hat{E}$. This makes $\hat{E}$ quadratic and positive definite in $\mathbf{s}$'s parametric derivatives, and similarly in

[3]It is tempting to damp the second-order terms in the system matrix $\mathbf{S}^T\mathbf{S}$ to insure well-conditioning without these repeated fitting attempts, but this noticeably degrades the fairing in Section 4.

the node positions, thus guaranteeing a unique minimum. Note that the E itself is *not* a quadratic function of the node positions, and so a single minimization of $\hat{E}$ with fixed $\mathbf{n}$ and a will not in general minimize E. We will return to this point shortly.

In manipulating gradients of $\hat{E}$ it will be notationally convenient to consider a "flattened" $\mathbf{P}$, formed by concatenating its x, y, z components into a single long vector $\bar{\mathbf{P}}$. Because $\hat{E}$ is purely quadratic in $\mathbf{P}$, it is possible to write it in the form $\hat{E} = \bar{\mathbf{P}}^T\mathbf{H}\bar{\mathbf{P}}$, where $\mathbf{H}$ is a constant, $3n \times 3n$ matrix. In the constrained minimization scheme below, $\mathbf{H}$ will never actually be computed and stored as a monolithic matrix; instead, vector products with $\mathbf{H}$ (and sub-matrices of $\mathbf{H}$) will be computed. This is done by looping over the nodes, accumulating each neighborhood's contribution to the product. Some rearrangement of (11) yields a convenient form for this matrix-vector product:

$$\mathbf{H}\bar{\mathbf{P}} = \sum_{\text{nodes}} \mathbf{n}(\mathbf{P}\,\mathbf{n})^T(\mathbf{B}^T{}_3\mathbf{B}_3 + 2\mathbf{B}^T{}_4\mathbf{B}_4 + \mathbf{B}^T{}_5\mathbf{B}_5)a \qquad (12)$$

(note that the $\mathbf{B}_j$ are row vectors). In addition to saving on the work of constructing $\mathbf{H}$, this multiplication scheme also exploits $\mathbf{H}$'s inherent sparsity without any additional effort on our part.

4.3 Curves

Before taking up the constrained minimization of E, we briefly mention fairing for point-sampled curves. This is well-trodden ground ([19]), but to keep our presentation self-contained we point out that the geometric curve fairing objective

$$E = \int \kappa^2 ds, \qquad (13)$$

can be formulated analogously to the surface objective above. The shapes of space curves in our modeler are controlled this way, subject to point interpolation constraints, below. They in turn control the shapes of embedded surface curves, with corresponding sequences of surface nodes constrained to track the nodes of free-standing curves.

4.4 Geometric constraints

Point and curve interpolation constraints on a surface are enforced by simply freezing the positions of their associated nodes during fairing. A frozen node is no longer considered an independent variable, but it contributes linear terms to $\hat{E}$. Splitting $\bar{\mathbf{P}}$ into unconstrained and constrained parts $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$ and partitioning $\mathbf{H}$ accordingly,

$$\hat{E} = [\bar{\mathbf{Q}}^T\bar{\mathbf{R}}^T] \begin{bmatrix} \mathbf{H}^{QQ} & \mathbf{H}^{QR} \\ \mathbf{H}^{RQ} & \mathbf{H}^{RR} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{Q}} \\ \bar{\mathbf{R}} \end{bmatrix} \qquad (14)$$

The gradient of $\hat{E}$ with respect to the active nodes is then

$$\nabla\hat{E} = \mathbf{H}^{QQ}\bar{\mathbf{Q}} + 2\mathbf{H}^{QR}\bar{\mathbf{R}}. \qquad (15)$$

We mention in passing another useful constraint, the *hinge*[4], which allows cross-boundary tangents along a surface boundary to be controlled. Under our scheme, with boundaries aligned parametrically in the u direction, this amounts to constraining $\mathbf{s}_v(0, 0)$ (which is simply $\mathbf{B}_2\mathbf{P}$) at each point on the boundary. This constraint cannot be directly enforced by freezing independent variables, as was done with point constraints, so a penalty[27] or Lagrange multiplier[33] technique should be used.

4.5 Minimizing the objective

To minimize $\hat{E}$ subject to the point constraints, we solve for the $\bar{\mathbf{Q}}$ yielding $\nabla\hat{E} = 0$. Rather than form $\mathbf{H}$ explicitly, the system is solved using a conjugate-gradient method[33], which only requires matrix-vector products with $\mathbf{H}$, not an explicit representation $\mathbf{H}$ itself[4].

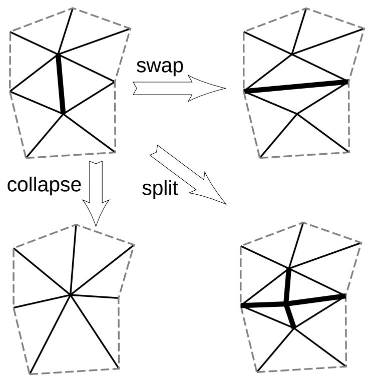

Figure 4: Mesh transformation operators: edge split, edge collapse, and edge flip. These change the mesh without altering its manifold topology.

As was mentioned, a single minimization of $\hat{E}$ does not in general minimize E. This minimization will be iterated as the user interacts with the surface, each time reparameterizing neighborhoods then using the current surface normals and areas in evaluating $\hat{E}$. It is tempting to claim that when (if) the surface reaches equilibrium, it will have approached the minimum for E; but this disregards the possibility that the ignored gradient terms are nonzero. In any event, the resulting shapes are visually pleasing, and an accurate minimization of E thus seems less important here than the added speed and robustness that have been gained through the linearization.

There is little we can say formally about the conditions under which this iterative scheme converges to an equilibrium; but in our experience the minimization has been well-behaved over a wide range of configurations. As with almost any nonlinear optimization, a caveat is that "reasonable" initial surface shapes must be used. This has not been a problem in our interactive system because changes to shape are generally incremental. There are configurations in which curvature minimization is undesirable as an objective function, as with narrow cylinders (which collapse). A minimum curvature variation functional, as in Moreton and Séquin[22], would remedy this problem; but it is not clear how to compute the curvature derivatives given our local quadratic reconstructions, and we leave this as future work.

5 Surface sampling

Taking the view that a mesh is a discrete sampling of some smooth underlying surface, it is important that the surface be sampled sensibly. In our case, "sensibly" means samples are distributed and triangulated so as to give good neighborhood shapes, so that local polynomial fitting is well-conditioned. It also means samples are distributed for speed and accuracy in the global approximation. While there is no numerical harm in having many samples clumped together in an uninteresting place on the surface, their presence needlessly slows down the computation. On the other hand, a large gap in sampling over an area will poorly resolve the shape. Since we generally do not know surface shapes or topologies *a priori*, the surface sampling and triangulation must be controlled dynamically as points move about and neighborhoods change shape.

[4]To maintain interactivity as this system increases in size, we allow the conjugate gradient solver only a fixed number of iterations (10–20) per solve/redraw cycle. A (not unpleasant) side-effect of this is an illusion of viscous drag as the solution converges over time.

5.1 Sample point distribution

We begin with the problem of maintaining a uniform sampling density over a surface. One approach that has been used successfully for both smooth and polyhedral surfaces is point repulsion[29, 38,39,42]: points move under the influence of mutual repulsive (or attractive) forces between points, constrained to act within the surface. In these schemes, a pair of points' influence on each other falls off as an inverse power of their separation. Ideally, this separation is measured in terms of geodesic distance on the surface, since in a highly curved surface, two points that are nearby geometrically may be far apart geodesically. As a practical matter, these schemes project a 3D neighborhood (there being no triangulation to deliver up a surface neighborhood) onto each sample point's tangent plane and measure these projected distances. Forces are accumulated to produce a velocity for each point, and as points are moved in these directions they must be repeatedly pushed back onto the surface to counter this integration drift.

These schemes work best when sampling is dense enough that repulsion forces are acting over small distances and neighborhoods are not highly curved (these assumptions are implicit in making the above-mentioned projections). Interestingly, these methods do not break down when the assumptions fail. So long as there is a robust procedure for returning points to the surface after integration error has moved them off, it doesn't really matter if some point accidentally lands in a geodesically distant location (assuming this behavior doesn't prevent the method from reaching equilibrium). However, this is behavior is clearly disastrous if we are dragging a triangulation along with the points, as in our application.

We consider instead a parametric repulsion scheme that uses the neighborhood structure given by the triangulation to avoid making wild leaps off the surface. In place of a neighborhood projection onto a node's tangent plane, we use the geodesic polar parameterization of Section 3.1. Since this includes only the neighboring points in the triangulation (whereas a repulsion scheme relies potential interactions between all points) we will optimize the placement of the node within this polygonal neighborhood. Imagining a uniform continuous distribution of samples within the neighborhood, the equilibrium position for the node under an r^2 repulsion/attraction scheme would be the centroid of the polygon.

We move the point parametrically towards the neighborhood centroid, then the local surface function is evaluated to determine the new node position in 3D. This avoids the need to project the point back onto the surface after integrating a 3D velocity vector. Thus the scheme will work properly even in situations where a neighborhood's shape is folded over or collapsed. Note that this treats the surface as if it were a (non-smooth) union of quadratic bowls, over which samples are sliding.

As it happens, this method is closely related to a "mesh improvement" scheme called *Laplacian smoothing*[11], so named because its fixed point is an approximate solution to Laplace's equation over the mesh. In fact, Laplace's equation is ubiquitous in computational mesh generation[36,37], arising naturally from a variational formulation of a uniform-density objective. We experimented with forming the Laplacian over a surface mesh, in terms of locally reconstructed neighborhoods[17], and solving the resulting global error minimization for point positions. In practice, we found that this was more robust than the purely local scheme above, but was more expensive to compute, and in the end we reverted to the local scheme.

5.2 Sample point density

We use a simple spatial refinement/un-refinement scheme to control the local density of sample points. Node separations are measured in 3D, and an edge-splitting refinement (see Figure 4) is triggered if any two neighbors are too far apart. Similarly, if any node is too close to each of its neighbors, the node is destroyed using an edge-collapse operation. Both of these operations preserve the surface topology, and Hoppe, et. al.[16] showed that they are sufficient to transform any surface triangulation into any other of the same topological type.

5.3 Surface triangulation

The sample distribution scheme just presented moves nodes around within their respective triangulated neighborhoods. In addition to this neighborhood smoothing, we will dynamically maintain a quality triangulation over the nodes as they change position. In addition to the numerical conditioning benefits that result from a good triangulation, this will also have the effect of allowing nodes to migrate across neighborhoods. Surface features like bounded subregions and embedded curves will be free to slide around relative to each other within the surface triangulation (e.g., the final frame in Figure 5).

Field[11] has shown that combining a Laplacian neighborhood smoothing scheme and Delaunay triangulation (DT) tends to produce nice triangulations. For our meshes we will work with a generalization of the planar DT due to Chew[6]. He generalizes the "empty circumcircle" characterization of the planar DT to one of empty circumspheres on a surface, and proves that it retains many of the desirable properties of the planar DT. The surface DT maximizes the minimum angle in the triangulation (measured in 3D), and thus eliminates skinny triangles. Further, this globally optimal triangulation can be constructed from any other surface triangulation by a series of local edge-flip operations(Figure 4), each of which increases the minimum angle within some quadrilateral in the mesh. These edge-flips preserve the topological type of the surface, so there is always a valid mesh as retriangulation progresses.

5.4 Constrained triangulation

In constructing the surface DT given an initial surface triangulation, there will be edges that must not be disturbed, such as those that are part of embedded control curves. A scheme that incorporates these *source edges* is referred to as a *constrained Delaunay triangulation*, or CDT [2]. It enjoys the same minimum-angle property as the DT (over all triangulations which include the source edges), and this leads directly to the *flip algorithm* for restoring a CDT given an initial triangulation:

> **Algorithm Restore-CDT[2]:** For an edge e not a source edge, Q_e is the quadrilateral formed by taking the two triangles on either side of e. We say that Q_e is *reversed* if e forms a smaller minimum angle with the outside edges than the other diagonal does. Initially, put all non-source edges into a queue. Repeatedly remove the first edge e, each time checking to see if Q_e is reversed. If so, e is flipped in the triangulation, replacing it with Q_e's other diagonal. The non-source edges of Q_e are added to the queue (if not already present). When the queue is empty, the CDT has been restored.

This terminates in at most $O(edges^2)$ flips, but in practice we see only a few flips at any one time. We can introduce a bit of hysteresis by only flipping edges if they increase their local minimum-angle by some small minimum. This produces an approximate CDT by making edges somewhat more reluctant to flip.

A technicality of this definition of the surface DT is that the generalization of consistent circumcircles to circumspheres on surfaces depends on a local flatness assumption: that no dihedral angle exceeds $\pi/2$. Rather than enforce this requirement by refining the triangulation in highly curved neighborhoods, we have found that it works well in practice to relax the requirement by maintaining only an approximate DT. Edges with sharp dihedral angles are treated as temporary source edges and are not allowed to be flipped, thus preserving the algorithm's termination guarantee.

In our modeler, Restore-CDT runs continually, interleaved with steps of the shape optimization, so that a quality mesh is always present for computation. An important aspect of this is approach is that we always begin with valid surface triangulation. Improving the quality of a surface triangulation is much simpler than attempting to impose a surface triangulation on a collection of scattered points (hence inferring a surface topology) — and it is guaranteed never to fail. In keeping with this approach, modeling operations which change the mesh structure or surface topology (Section 6.3) must do so in controlled ways that always leave a valid triangulation.

6 Operations on surfaces

The previous three sections develop sufficient machinery to approximate faired surfaces that interpolate shape control curves. In this section we consider using this machinery as a basic computational substrate for free-form modeling. Recall that in this approach a designer will interact only with approximate renderings of surfaces, but that anything done to a surface model must be interpreted in terms that define an ideal variational shape.

We consider here some basic modeling operations cast in terms of our variational substrate. The intent is to illustrate important considerations in using variational surfaces as a basic shape representation, rather than offer an exhaustive set of sculpting operations. A common theme is that of carving a surface up into disjoint regions and designating a shape controller for the nodes in each region. Mixing explicit shape control for some regions and functional minimization for others will allow us to construct structured models for parameterized free-form shapes.

6.1 Embedding control curves

A surface control curve is specified by designating a series of (not necessarily neighboring) points on the surface that the control curve should pass through. These points may be specified by a designer "drawing" a curve on a triangulated surface, or may be created automatically by the modeler as part of a composition operation discussed in this section. *Face splitting* is used to allow such points to be placed at arbitrary locations on the surface. This operation adds a new node in the middle of a triangular face and connects it to each vertex of the triangle (equivalent to an edge-split followed by an edge-flip, with suitable repositioning of the new node). Once the desired anchor nodes have been inserted into the mesh, a sequence of nodes and edges is inserted to join the designated nodes in a PWL curve. One way to insert a curve connecting two nodes is to first find a sequence of abutting faces that connect them; their union then forms a polygonal channel into which a curve connecting the two nodes can be inserted by splitting each edge that crosses the channel. Once the surface curve has been created, a matching space curve is created (with point constraints at the original anchor positions) and the surface curve is constrained to follow it. Note that our procedure needn't create a particularly straight surface curve, or leave a particularly nice triangulation (it does neither), since fairing and Delaunay triangulation will subsequently neaten things up.

6.2 Bounded surface regions

Many computations are meant to be performed only on some subset of the surface (e.g., splitting operations, or the application of shaping tools to surface regions). This will require that surface nodes and edges be classified as being inside, outside, or on the boundary of a surface region delineated by an embedded closed curve. The classification is straightforward if performed edge-by-edge:

Algorithm Find-Interior-Edges: Given an edge e_{seed} in the interior of the region to be collected, a closed series of edges representing the region boundary, and a list of surface edges, gather all interior edges into a list. Begin by adding edge e_{seed} to an (initially empty) check queue. Then, while the check queue is not empty, remove the first edge $e_{interior}$ and add it to the list of interior edges. Let Q_e be the quadrilateral formed by taking the two triangles on either side of $e_{interior}$. For each edge e_{test} of Q_e, check to see if it is in the list of boundary edges or interior edges. If not, add e_{test} to the check queue. When the check queue is empty, all interior edges have been found.

6.3 Changing the topology

We must be able to make controlled changes to surface topology: splitting along an embedded curve to create a new boundary, or stitching two surfaces together along boundary edges. Introducing a crease in a surface along a curve or smoothing such a crease is also implemented in terms of splitting and merging, since such a discontinuity is actually represented using two independent surfaces whose boundary curves are constrained to coincide.

Curve correspondence: To merge two surfaces along a pair of boundaries, or skin two boundary curves with a single sheet, the nodes on the two curves must first be put into correspondence. In general, something like Sederberg's scheme [31] might be used to robustly determine this correspondence. A simpler (though by no means fail-safe) procedure is to iteratively refine the curve with fewer edges by splitting its longest edge until both curves have the same number of nodes. Then choose the alignment that minimizes the sum of squared distances between nodes. This works well in our modeler in the common situation where the user brings curves into proximity with each other before triggering a merge. The resampling and Delaunay triangulation processes quickly iron out artifacts that may result from the simpler matching procedure once the merge has been completed.

Splitting and merging surfaces: Recall that a triangulated surface is represented as a collection of nodes, each with a radially ordered list of neighbors. To merge two boundary curves that have been put into correspondence, for each pair of nodes move the interior edges of one node to the other, preserving their radial order. This will convert the latter nodes from boundary nodes to infield nodes, and leave the former boundary curve completely disconnected from the merged surface, whence its nodes and edges may be destroyed. Splitting a surface along a closed infield curve is the inverse of this: classify the node edges as being part of the region's interior, exterior, or boundary. Clone the boundary nodes and edges. Then delete the interior edge connections from one boundary copy, and the exterior edge connections from the other.

6.4 Surface intersections

A limitation of this modeling approach is that we cannot consider operations that depend on approximated shapes to tell us something about ideal shapes. For example, we shouldn't look for points of intersection between two approximate surfaces in order to answer the question, "do the variational surfaces intersect?" Because of discretization error, whether or how two approximate surfaces intersect says nothing about the true intersection topology.

This would seem to rule out an important style of design in which intersecting surfaces are trimmed against each other and joined along their intersection curves. But a version of trim-and-stitch surface composition that is eminently suited for our approach treats such intersection curves as free-standing boundary or character lines in the composite surface. To "intersect" two variational surfaces, a curve is first constructed that approximates the shape of the intersection curve of the explicit surfaces[5]. Then the parent surfaces are *re-*

[5]In our implementation, the approximate curve is coarsely sampled, and these fixed samples are used as the anchors of a free-standing curve.

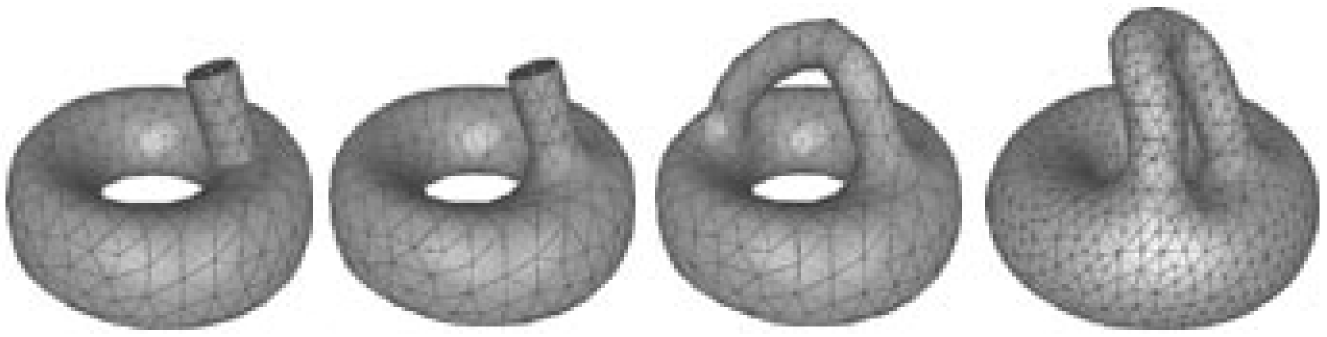

Figure 5: Adding a handle: 1. a cylinder and a torus. The red glow indicates they are close enough for an automated join operation. 2. Join complete: a hole has been cut in the torus and a blend region added connecting it to the cylinder. Boundary curves for the blend region control the tightness of the blend. 3. The cylinder is extended to the opposide wall and joined (500 nodes). 4. The handle's far attachment point is dragged towards the initial attachment point (surface refined to 1500 nodes).

defined to interpolate this independent curve as their new boundary. This somewhat backwards notion of trimming has the advantage over passive surface intersection that the trim curve can be directly reshaped by the designer.

This style of surface composition simplifies the implementation of familiar intersection-based operations. For example, consider applying a parameterized embossing tool (a "branding iron" with control knobs) to a curved surface. Implementing such a tool becomes simple if we observe that, since the tool will be given control over the intersection shape, we don't really need to perform the initial surface intersection and trimming. To apply the tool, we can simply specify that a given surface region contains a hole matching the tool outline. The tool itself then supplies the fitted piece of surface that fills the outline. In this example, the explicit surface is used only for topological information — allowing the user to indicate (e.g., by proximity to the tool) which surface region is to be affected. The result is unambiguous and independent of the resolution of the current surface approximation.

6.5 External shape controllers

Because it would be cumbersome to specify all shapes in terms of functional minimization, we also allow externally represented shapes to control the shapes of bounded surface regions. These can be thought of as shape-control tools which may be applied to variational surfaces. For example, in our sculpting system we define a cylindrical offset tool in terms of a space curve backbone and a radius function (Figure 6). A controlled region may contain subregions that are in turn controlled by other shaping tools. This lets us, e.g., attach handles or drill holes in externally controlled regions, so that ultimately their local topology may be very different from that of the shape controller.

Nodes along the boundary between an externally controlled region and a faired region will have a mixture of free and constrained neighbors. This allows the shape of the surface on the controlled side to influence the shape on the faired side up to second-order, so that tangent and curvature information are propagated across the boundary. Since only one shape control tool can drive a given surface point at a time, it is not generally possible to have two tool-controlled regions meet along a shared control curve. In these situations a faired blend region must be installed between them (see, e.g., the body/handle attachment on Figure 7).

In implementing such controllers, there are any number of ways to triangulate and track a tool shape, depending on how the tool surface is represented, and we will not pretend to a complete discussion. As an example from our system, a number of tool shapes are defined as implicit surfaces, and we use the surface-tracking technique from Witkin and Heckbert's point-sampling scheme [42] to keep sample points glued to moving tool surfaces. The sample distribution and triangulation over the tool surface itself is handled just as it was for faired surfaces. Since the the topologies of the implicit surface tools in our system are known *a priori* and remain fixed during sculpting, we avoid the difficult general problem of triangulating an implicit surface by generating a valid initial triangulation when the tool is applied. Nonetheless, it is still possible for points to behave badly by bunching up when a tool surface is moved quickly, and this approach can certainly be improved on. If tool surfaces have an associated parameterization, the sampling and tracking may be done using parametric coordinates and these difficulties do not arise.

6.6 Building structured models

The composition of parameterized shapes via blending regions leads naturally to structured free-form models. As in Bonner, *et al*'s work with tubular structures [3], these shape control tools may be organized in a variety of ways through the use of deformation hierarchies (note that the deformations are not applied to the surface itself). In our system the resulting collections of shape control tools and character lines serve as "skeletons" supporting a triangulated surface skin. As the underlying control shapes are changed, the surface tracks them, automatically adjusting blend shapes, refining, unrefining, and re-triangulating to maintain a good approximation to the composite shape.

7 Conclusion

We have presented an approach to designing fair, free-form shapes using triangulated surfaces. Our initial implementation of these ideas is a modeler that runs at interactive speeds on a Silicon Graphics Indigo class workstation, for surface models of several hundred nodes (in the illustrations in this paper, any surfaces containing more than 500 nodes were given a final refinement after all interactive shaping was complete).

7.1 Contributions

The principal contribution of this work is a scheme for interactively designing fair free-form shapes of arbitrary, mutable topology. Little work has appeared regarding topological design for free-form shapes (though see [10]). Our approach uses a triangulated mesh to represent a surface model's topology, and interactive modeling operations alter the mesh to change this topology in controlled ways.

Geometric fairness functional: The positions of the triangulated sample points in 3D approximate the shape of an underlying smooth surface, whose shape is defined as the solution of a functional minimization. To this end, we use a geometric fairness functional based

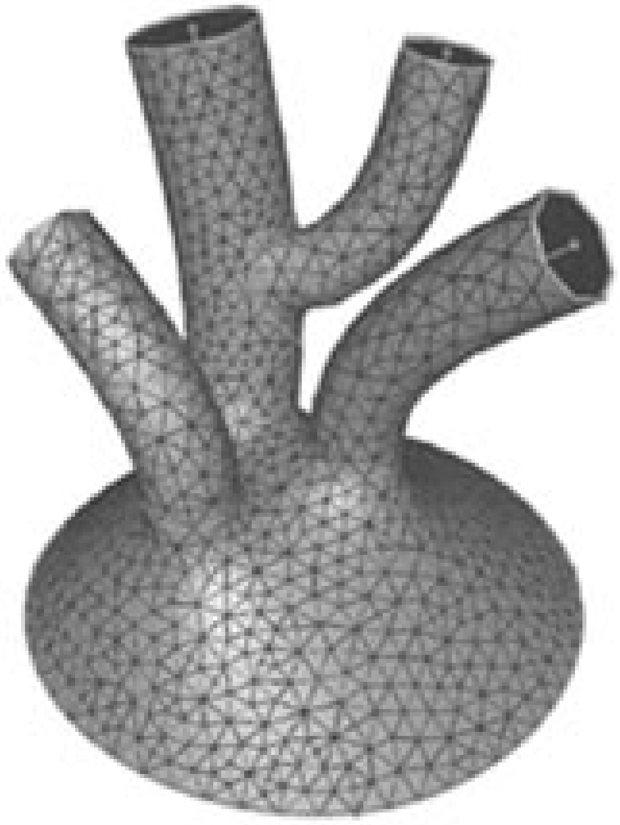

Figure 6: A branching surface. Three cylinders, controlled by offset cylinder tools, meet in a minimum curvature surface. A hole has been cut in one of the cylinders, and a fourth cylinder attached there using a faired skirt (800 nodes).

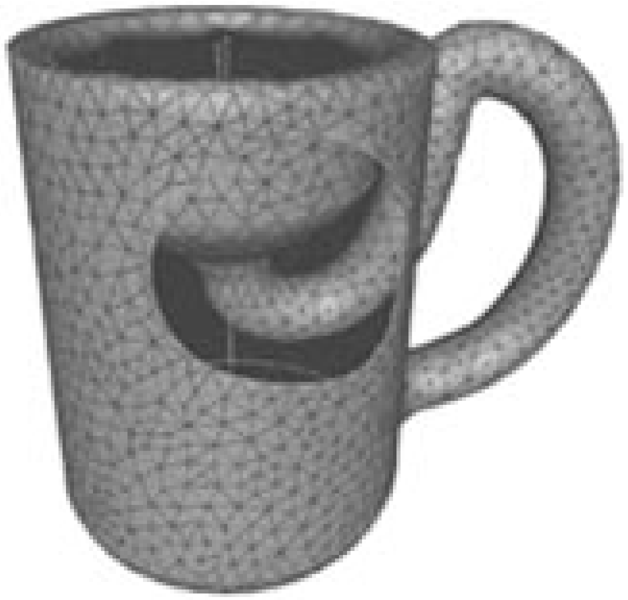

Figure 7: A Klein mug. A single, self-intersecting surface whose handle and outside and inside walls are each controlled by cylinder tools (800 nodes).

on extrinsic surface curvature, and thus avoid shape artifacts related to surface parameterization. This is a highly non-linear functional, yet we have developed an approach to its optimization that is fast and robust enough to withstand interactive re-shaping.

Neighborhood parameterization: In order to perform this optimization over a triangulated surface mesh, we perform local smooth surface reconstruction that estimates surface derivatives while honoring the neighborhood topologies induced by the mesh. Our construction of neighborhood parameterizations uses a projection scheme we have not seen elsewhere.

Interactive adaptive meshing: Finally, the fairing computation is interleaved with adaptive refinement, adaptive sample distribution, and re-triangulation. The approach used in this work, while not fundamentally new, is used here to novel effect: features are free to slide around in a surface so as to help minimize the fairness functional, rather than being bound by parametric surface coordinates to a fixed place in the surface.

7.2 Future Work

Control nets: Though we formulate surface shape control using interpolated control curves, our scheme does not yet accommodate intersecting control curves. A compatibility condition[26] demands that when control curves meet at a point, they must all fit a common quadratic surface form; otherwise, no there can be no smooth interpolating surface in the neighborhood of the intersection. What is needed is a special intersection node that enforces this compatibility constraint on curves meeting there (like the hub of an umbrella).

Smooth surfaces: It would be interesting to develop a version of this approach using smooth triangular patches. We expect it to be computationally expensive (this was our motivation for using point-sampled approximations), but it may be that a very coarse surface refinement in terms of patches could be made to perform comparably to a more highly refined point-sampled surface, thus offsetting this cost.

Curvature-adaptive sampling: Finally, it may be worthwhile to consider a curvature-sensitive scheme for distributing sample points across the surface. The error of our objective function integration in a neighborhood is related to the neighborhood's total curvature, and an adaptive scheme would would tend to distribute this error more evenly across nodes.

Acknowledgements

The authors would like to thank the Siggraph reviewers for many helpful comments on the early draft of this paper (and to Reviewer #5 — you're welcome). Thanks also to Hans Pedersen and Zoran Popović for their help with the figures. This work was supported in part by the Engineering Design Research Center, an NSF Engineering Research Center at Carnegie Mellon University, by a Science and Technology Center Grant from the National Science Foundation, #BIR-8920118, by a High Performance Computing and Communications Grant from the National Science Foundation, #BIR-9217091, by Apple Computer, Inc., and by an equipment grant from Silicon Graphics, Inc.

References

[1] Timothy Barth. Higher order solution of the euler equations on unstructured grids using quadratic reconstruction. In *28th Aerospace Sciences Meeting.* AIAA-90-0013, 1990.

[2] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. Technical Report CSL-92-1, XEROX Palo Alto Research Center, March 1992.

[3] D.L. Bonner, M.J. Jakiela, M. Watanabe, and N. Kishi. Pseudoedge: nonintersected parametric quilt modeling of multiply connected objects. *Computer Aided Design*, 25(7):438–452, July 1993.

[4] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics*, 25(4), July 1991. (Proceedings Siggraph '91).

[5] George Celniker and William Welch. Linear constraints for nonuniform B-spline surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, 1992.

[6] Paul Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, 1993.

[7] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume volume I. Wiley, 1937.

[8] M. Eppstein. On the influence of parameterization in parametric interpolation. *SIAM J. Numer. Anal.*, 13:261–268, 1976.

[9] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1990.

[10] Helaman Ferguson, Alyn Rockwood, and Jordan Cox. Topological design of sculptured surfaces. *Computer Graphics*, 26(2), July 1992. (Proceedings Siggraph '92).

[11] D. A. Field. Laplacian smoothing and delaunay triangulations. *Comm. Appl. Numer. Methods*, 4:709–712, 1984.

[12] G. Forsythe and W. Wasow. *Finite Difference Methods for Partial Differential Equations*, chapter 19, pages 179–182. John Wiley and Sons, 1960.

[13] Richard Franke and Gregory Nielson. Scattered data interpolation and applications: a tutorial and survey. In Hans Hagen and Dieter Roller, editors, *Geometric Modeling.* Springer-Verlag, 1991.

[14] Gene Golub and Charles Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1989.

[15] Victor Guillemin and Alan Pollack. *Differential Topology*. Prentice-Hall, 1974.

[16] Huges Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of Siggraph 93*, 1993.

[17] Geertjan Huiskamp. Difference formulas for the surface Laplacian on a triangulated surface. *Journal of Computational Physics*, 95:477–496, 1991.

[18] Michael Kallay. Constrained optimization in surface design. In *Modeling in Computer Graphics.* Springer Verlag, 1993.

[19] Michael Kass, Andrew Witkin, and Dimetri Terzopoulos. Snakes: Active contour models. *International Journal Computer Vision*, 1(4), 1987.

[20] N. J. Lott and D. I. Pullin. Method for fairing b-spline surfaces. *Computer-Aided Design*, 20(10), 1988.

[21] Steve Mann, Charles Loop, Michael Lounsbery, D. Meyers, J. Painter, Tony Derose, and K. Sloan. A survey of parametric scattered data fitting using triangular interpolants. In *Curve and Surface Modeling.* SIAM.

[22] Henry Moreton and Carlo Séquin. Functional minimization for fair surface design. *Computer Graphics*, 26(2), July 1992. (Proceedings Siggraph '92).

[23] Henry P. Moreton. *Minimum Curvature Variation Curves, Networks, and Surfaces for Fair Free-form Shape Design.* PhD thesis, University of California, Berkeley, 1993.

[24] Gregory Nielson. A transfinite, visually continuous, triangular interpolant. In Gerald Farin, editor, *Geometric Modelling*, pages 235–246. SIAM, 1987.

[25] Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[26] Jörg Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7:221–246, 1991.

[27] John Platt. *Constraint Methods for Neural Networks and Computer Graphics.* PhD thesis, California Institute of Technology, 1989.

[28] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):145–152, July 1987.

[29] M. M. Rai and D. A. Anderson. Application of adaptive grids to fluid-flow problems with asymptotic solutions. *AIAA J.*, 20:496–502, 1982.

[30] D.G. Schweikert. An interpolation curve using a spline in tension. *Journal of Math and Phys.*, 45:312–317, 1966.

[31] Thomas Sederberg and Eugene Greenwood. A physically based approach to 2D shape blending. *Computer Graphics*, 26(2), July 1992. (Proceedings Siggraph '92).

[32] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish, Inc., 1979.

[33] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, 1986.

[34] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2), July 1992. (Proceedings Siggraph '92).

[35] D. Terzopoulos. Multi-level reconstruction of visual surfaces. *MIT Artificial Intelligence Memo Number 671*, April 1981.

[36] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation: Foundations and Applications.* North-Holland, 1985.

[37] Joe F. Thompson. A survey of dynamically-adaptive grids in the numerical solution of partial differential equations. *Applied Numerical Mathematics*, 1:3–27, 1985.

[38] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):289–298, July 1991.

[39] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, July 1992. (Proceedings Siggraph '92).

[40] William Welch. *Free-Form shape design using triangulated surfaces.* PhD thesis, Carnegie Mellon University, (in preparation) 1994.

[41] William Welch and Andrew Witkin. Variational surface modeling. *Computer Graphics*, 26(2), July 1992. (Proceedings Siggraph '92).

[42] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *In these proceedings*, July 1994.