# Functional Optimization for Fair Surface Design

*Henry P. Moreton*
*Carlo H. Séquin*

Computer Science Division
University of California, Berkeley

## Abstract

This paper presents a simple-to-use mechanism for the creation of complex smoothly shaped surfaces of any genus or topological type. The surfaces are specified through interpolated geometric constraints consisting of positions and, optionally, surface normals and surface curvatures. From a designer's point of view, this is a very natural way to specify a desired shape, whether free-form or technical. Nonlinear optimization techniques are then used to minimize a fairness functional based on the variation of curvature. This functional produces very high quality surfaces with predictable, intuitive behavior, while generating, where possible, simple shapes, such as cylinders, spheres, or tori, which are commonly used in geometric modeling. While easy to use, this optimization-based approach is computationally quite demanding. With more efficient optimization algorithms and with the ever increasing processing power available on every desk-top, the techniques described here will provide the basis for a new class of practical interactive geometric modeling tools.

## 1 Introduction

In this paper we present a simple-to-use mechanism for the creation of complex, smoothly shaped models of any genus or topological type. The shapes are specified using interpolated geometric constraints. The resulting models accurately reflect these specifications and are free of unwanted wrinkles, bulges, and ripples. When the given constraints indicate and/or permit, the resulting surfaces take on the desirable shapes of spheres, cylinders, cones, and tori. Specification of a desired shape is straightforward, allowing simple or complex shapes to be described easily and compactly. For example, a "suitcase corner," the blend of three quarter cylinders of differing radii, is formed by specifying just six sets of constraints (Fig. 1).

A Klein bottle is specified with equal ease; only twelve point-tangent constraints are used to model the surface shown in Figure 2.

Authors' addresses:
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720
H.P. Moreton, (510) 339-8715, moreton@cs.berkeley.edu
C.H. Séquin, (510) 642-5103, sequin@cs.berkeley.edu



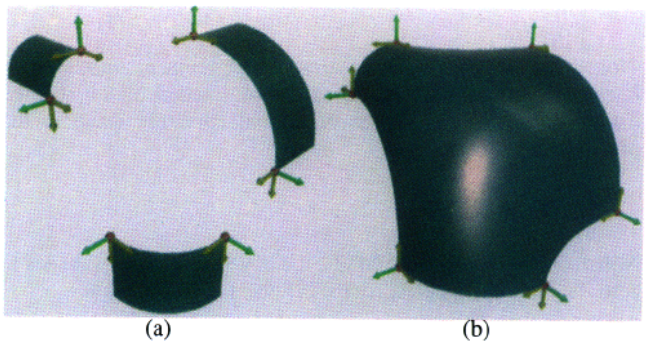



**Figure 1. A suitcase corner. (a) illustrates the specification with normal and curvature constraints. (b) illustrates the resulting blend.**

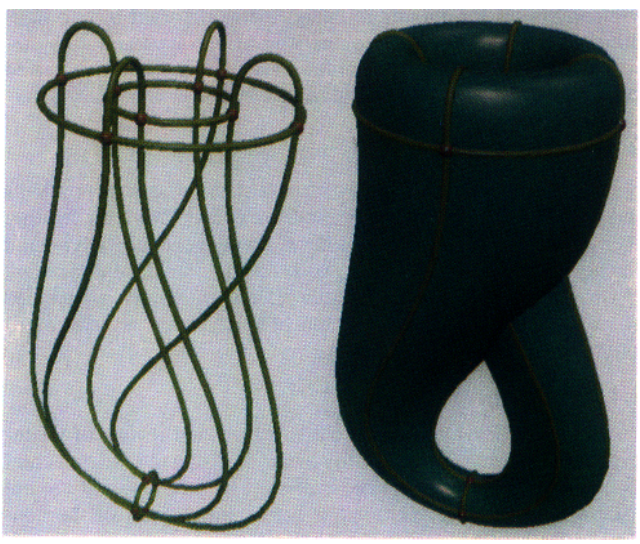


**Figure 2. A Klein bottle defined by twelve constraint sets.**

The work described here is the result of research into the fairness of curves and surfaces specified through geometric interpolatory constraints. These geometric constraints consist of points, surface normals, and surface curvatures. We use nonlinear optimization techniques to minimize a fairness functional subject to the given geometric constraints.

Once the geometric constraints are satisfied by construction, the techniques described here set the remaining surface parameters (degrees of freedom) to minimize our fairness functional while maintaining $G^1$ continuity by using a penalty function. The problem of creating surfaces with $G^1$ continuity is very difficult

 ACM-0-89791-479-1/92/007/0167 $01.50

to solve satisfactorily. Most techniques use heuristics to set extra degrees of freedom and sufficient but not necessary constructions to guarantee $G^1$ continuity; however they typically produce unnecessary and undesirable "wrinkles". An extensive survey by members of the Graphics Group at the University of Washington [17] demonstrates these common flaws.

The curve and surface functionals that we have derived minimize the *variation* of curvature; thus we refer to the curves as *minimum variation curves* ($MVC^1$) and to the surfaces as *minimum variation surfaces* (MVS). In the case of curves, the integral of the squared magnitude of the derivative of curvature is minimized

$$\int \left(\frac{d\hat{\kappa}}{ds}\right)^2 ds. \qquad (1)$$

Note that this integral evaluates to zero for circular arcs and straight lines. For surfaces, the functional is the integral of the squared magnitude of the derivatives of normal curvature taken in the principle directions[2]

$$\int \left(\frac{d\kappa_n}{d\hat{e}_1}\right)^2 + \left(\frac{d\kappa_n}{d\hat{e}_2}\right)^2 dA.$$

Note that analogous to the MVC, the MVS functional evaluates to zero for cyclides: spheres, cones, cylinders, tori, and planes.

Section 2 reviews previous related work, discussing approaches, advantages, and shortcomings. Section 3 presents an overview of our approach outlining the steps taken to produce a surface model from specified constraints. Section 4 provides details of the representation of the surfaces, and a description of the optimization techniques used to compute them. Section 5 details the computation of the minimum energy networks used in our algorithm. Section 6 presents a comparison of our approach with other methods; it exhibits examples of complex surfaces created from simple, compact specifications.

## 2 Previous Work

The work described in this paper touches on several problems and thus several areas of study. First, we discuss work on creating a $G^1$ surface out of a collection of non-degenerate polynomial patches. Second, we reference work on functional minimization, constrained optimization, and finite element analysis, all applied to surface design. The last portion of this section reviews minimum energy networks.

### 2.1 $G^1$ Continuity

Peters [22] provides a good classification and review of $G^1$ interpolation techniques. All of the methods discussed are constructive, using heuristics to set those degrees of freedom that are not fixed by continuity constraints or set as side effects of the construction method. These methods rely on the computation of a network of curves that interpolate the data, subject to various continuity and connectivity constraints. Peters has done a great deal of work on the construction of geometrically continuous surfaces. His most recent work outlines a method for creating "$C^k$" surfaces. Relevant to this discussion, Peters [23] shows that a curve network maintaining $G^2$ continuity is sufficient and in general necessary for the construction of a $G^1$ surface. This result assumes that a single quintic polynomial patch is placed in each network opening, and that the opening boundaries are fixed. In addition, there are no restrictions on the order of (i.e., the number of edges joining) the network nodes.

Our work combines a solution to the $G^1$ continuity problem with the setting of the unconstrained degrees of freedom to form a fair $G^1$ continuous surface. No explicit $G^1$ construction is used; rather a suitable penalty function is incorporated into the objective function.

In related work, DeRose [10] presents the necessary and sufficient conditions for $G^1$ continuity between adjacent triangular and quadrilateral Bézier patches of equal degree. We use these results to formulate the penalty function that imposes $G^1$ continuity.

### 2.2 Optimization, Minimization, and Finite Element Analysis

In [32] Williams describes a system using finite difference methods for the computation of smooth surfaces. The system minimizes the total energy of a fictitious elastic plate. In [26] Pramila describes techniques for ship hull design that employ finite element analysis to minimize a quadratic functional approximating strain energy. Celniker and Gossard [6] present a free form design system that uses finite element analysis to simulate physical models. Interactive deformation is carried out by simulating forces applied to the subject model. Surfaces are represented by triangular patches meeting with $C^1$ continuity. Approximations are used to model deformations. As a result, surfaces converge on their theoretical shape after multiple elements are inserted between constraints. Rando and Roulier [28] propose several specialized geometrically based fairness functionals. These functionals are referred to as "flattening," "rounding," and "rolling." They apply these functionals to Bézier patches. Some of the Bézier control points are fixed in order to guarantee continuity, while others are varied to minimize the functionals. Hagen and Schulze [13] use the calculus of variations to fit generalized Coons patches to three-dimensional data. The resulting patches minimize a strain energy fairness criterion. The analysis uses simplifying approximations to limit the complexity of calculations. Most recently, Kallay and Ravani [15] discuss a method for determining "optimal" twist vectors for the surface formed by a rectangular mesh of cubic curves. In their work, twist vectors are computed to minimize a quadratic energy term.

Our work uses higher order patches and the full nonlinear expression for the functional to achieve the highest possible surface quality from the fewest underlying patches.

### 2.3 Minimum Energy Networks

Nielson [20] introduced the minimum norm network (MNN) using linear energy terms to produce a $C^1$ network and a resulting $C^1$ surface. Pottmann [25] presents a generalization of MNN to produce a $C^2$ surface. Most recently, we [19] describe an algorithm for the computation of a $G^2$ minimum energy network composed of curves minimizing (1) along the edges of the network. These MVC networks are of higher fairness and are usually closer to the corresponding minimum variation surface than networks computed heuristically or using linear energy terms.

## 3 Our Algorithm

We treat the problem of creating a surface interpolating a collection of geometric constraints as one of scattered data interpolation. The interpolation problem is broken into three steps (Fig. 3); 1) connectivity definition, 2) curve network computation, 3) patch blending. In accordance with the topological type of the desired surface, the geometric constraints are first connected into a network of straight edges. A curve is then placed at each edge of the network, and an optimized network is computed composed of minimum variation curves (MVC) subject to the specified geometric constraints and the additional constraint that the curve segments meet with second order geometric continuity, $G^2$, at the

1. Emery Jou coined this name/acronym.
2. It is our convention that a "hat", e.g. $\hat{e}$, indicates a unit vector.

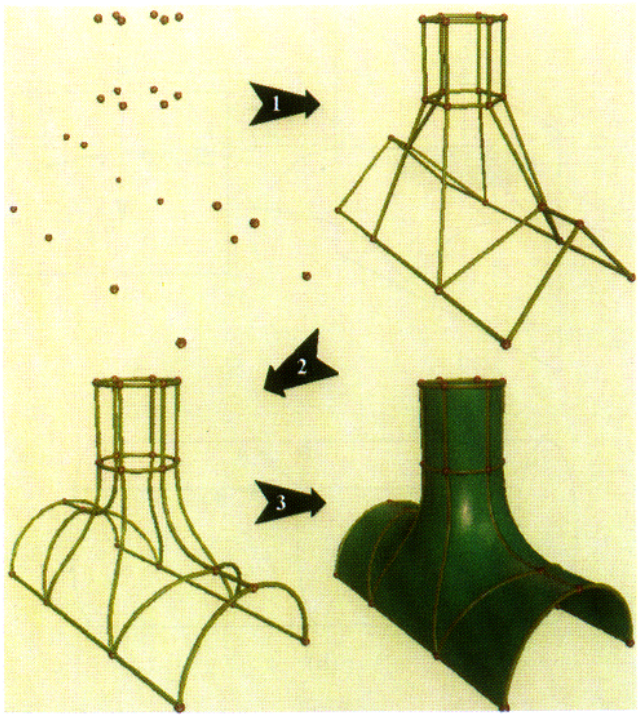



**Figure 3. The construction of a blend of two pipes.**

vertices. Finally, an interpolatory minimum variation surface (MVS) is computed, interpolating the MVC network with tangent continuity. In a first approach, the boundaries of the MVS patches are fixed, interpolating the previously constructed curve network. Alternatively, the surface calculation may use the MVC network as a starting point and modify its geometry during surface calculation. The latter approach yields even smoother surfaces, but at a substantially higher computational expense. The higher quality surfaces result because the curves of an MVC network resulting from a given constraint set do not always lie in the MVS resulting from the same set of constraints.

During the modeling process, the connectivity of the geometrical constraints is typically established as a natural outgrowth of the design process. The techniques described here are also amenable to true scattered data interpolation, in which case connectivity must first be derived with some other method, possibly based on some minimal triangulation on the data points.

Our system is based on triangular and quadrilateral patches. All constraints are located at corners of these patches. Additional vertices and edges may be added to the network so that is has only three and four sided openings. These additional vertices are not constraints and are appropriately positioned by the curve network computation and patch blending phases of the construction.

The computation of a curvature continuous ($G^2$) MVC network is cast as a nonlinear optimization or finite element problem. First an initial shape of the curve network is computed using heuristics based on the geometry of the network. The optimization then proceeds from this starting point using standard gradient descent techniques. The geometric constraints and the second order continuity of the curve network are maintained by construction. During each iteration, at each vertex of the network, the algorithm defines a surface normal, principle directions, and principle curvatures; all the curve segments are then constrained to remain consistent with them. Once the geometric and continuity constraints are satisfied, the gradient of the functional with respect to the remaining degrees of freedom is calculated, and the free parameters are iteratively adjusted to minimize the curvature variation functional overall.

Based on this MVC network, the computation of the MVS interpolatory surface is accomplished using constrained optimization. The geometric constraints are again imposed by constructions similar to those used in the calculation of the network. Patch-to-patch tangent continuity is imposed by means of a penalty function that is equal to zero when the patches composing the MVS meet with tangent continuity and proportionally greater than zero for any $G^1$ discontinuity. The use of penalty functions alone does not guarantee perfect tangent continuity. Exact tangent continuity may be achieved in a subsequent phase of optimization using Lagrange multipliers [8] or using the continuation method, a continuous reduction to zero of the weight of the curvature variation term in the functional. In practice, it is rarely necessary to resort to this second phase because the surfaces resulting from the first phase are of high quality and sufficiently close to tangent continuous. Mann and DeRose have shown this type of *approximate* tangent continuity to be sufficient and, in fact, desirable in some applications [9].

## 4 Representation and Computation

As described in section 3 the computation of an MVS satisfying a given set of constraints is broken into several steps. In this section we will focus on the last phase of the algorithm where surface patches are fit to a $G^2$ MVC network. The curves may remain fixed or they may be used simply as a starting point for optimization. The choice between fixed and variable curves is up to the designer and does not affect the algorithms described here. Section 5 provides the details of MVC network calculation.

The MVS is approximated by a quilt of parametric polynomial patches which interpolate the curve network, satisfy the geometric constraints, and meet with approximate tangent plane continuity. The surface functional is then minimized by varying the surface parameters that are not fixed by geometric constraints.

### 4.1 Bézier Patches

The curves of the network are represented by quintic Hermite polynomial segments; one segment replaces each edge of the network of constraints. Consequently, the patches making up the interpolatory surface are [bi-]quintic patches. Peters [23] has demonstrated that quintics are sufficient to achieve tangent continuity for all triangular/quadrilateral patch-patch combinations. One patch is used for each opening in the network. Though we have found single patches to have sufficient descriptive power, it is simple to subdivide network patches creating multiple patches per opening. The use of multiple patches improves the approximation of the theoretical MVS surface which in general has no closed form representation. Note that while Peters' construction requires that the curve network being interpolated has $G^2$ continuity, the interpolatory surface resulting from his construction is only $G^1$ across boundaries and at the vertices of the network. In contrast, our surfaces are constrained to meet with $G^2$ continuity at the vertices of the network (see section 4.8).

Even though the boundary curves are in the Hermite form, we have chosen to use Bézier patches because of their superior numerical characteristics and because the tangent continuity conditions we use are particularly concise when formulated in terms of Bézier coefficients. Also, Bézier patches are more amenable to rendering, and may be rendered directly by subroutines found in the graphics library of workstations such as the Silicon Graphics IRIS®.

## 4.2 Fairness Functionals

Our choice of functional for minimization was prompted by the need for very high quality surfaces with predictable, intuitive behavior, and the desire to capture shapes commonly used in geometric modeling. The fairness of curves and surfaces has been studied extensively and has been shown to be closely related to how little and how smoothly a curve or surface bends. For an early and interesting reference see [2].

Work on the *fairness of curves* has traditionally focused on the minimization of *strain energy* or the arc length integral of the squared magnitude of curvature [18]

$$\int \vec{\kappa}^2 \, ds .$$

We use an alternative fairness metric based on the minimization of the arc length integral of the squared magnitude of the *derivative* of curvature

$$\int \frac{d\vec{\kappa}}{ds}^2 ds .$$

This new functional results in curves with noticeably smoother curvature plots, and it has the added benefit that circular arcs are formed when constraints permit, since according to this new functional they are optimally curved.

Traditional work on the *fairness of surfaces* also focuses on strain energy, minimizing the area integral of the sum of the principle curvatures squared [13, 16, 21, 32]

$$\int \kappa_1^2 + \kappa_2^2 dA .$$

Again, our approach minimizes the *variation* of curvature, rather than its magnitude. For surfaces, we minimize the area integral of the sum of the squared magnitudes of the derivatives of the normal curvatures taken in the principle directions:

$$\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 dA . \qquad (2)$$

The *normal curvature* at a point on a surface in a direction specified by a surface tangent vector is determined from the intersection curve of the surface with the plane spanned by the surface normal and the given tangent vector. The principle directions, $\hat{e}_1$ and $\hat{e}_2$, and the principle curvatures, $\kappa_1$ and $\kappa_2$, at a point on a surface are the directions and magnitudes of the minimum and maximum of all possible normal curvatures at that point [11] (Fig. 4).

Like the MVC functional, the MVS functional has associated shapes that are optimal in the sense that the functional evaluates to zero. In the case of the MVS functional, the shapes belong to a special family of curved surfaces call cyclides [4, 27] that includes spheres, cylinders, cones, and tori. These all have *lines of principal curvature* where the associated normal curvature remains constant. Lines of principal curvature follow the paths of minimum and maximum normal curvature across a surface.

## 4.3 Parametric Functionals

The fairness functional for surfaces (2) is defined in terms of an area integral. To evaluate the functional and its gradient in the context of the parametric polynomial surfaces patches described in section 4.1, the functional must be converted to a compatible form. Here we outline the calculations necessary to evaluate the functional. The fairness functional is computed for each patch, and the value of the functional for the surface as a whole is the sum of the values for each patch. The area based definition

$$\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 dA$$

is converted to integrals of functions of the independent parameters

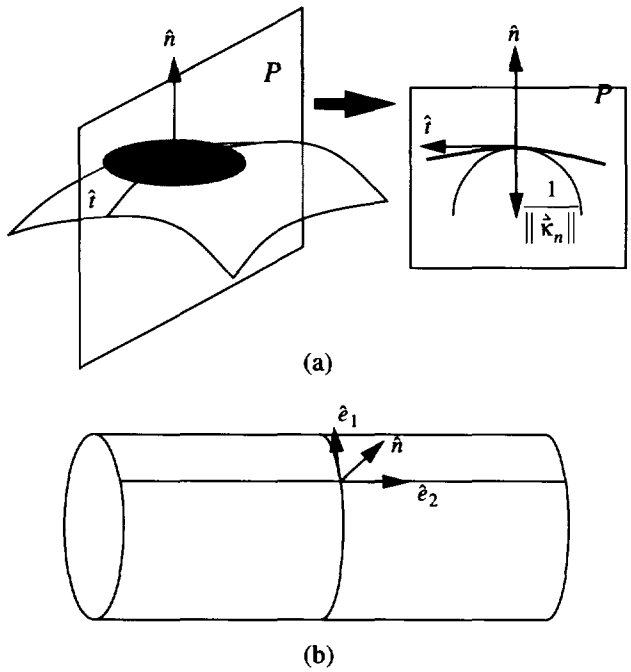



**Figure 4. (a) Normal curvature is the curvature of the curve formed by the intersection of the surface and a plane containing the normal and tangent. (b) Principle directions and principle curvatures are the directions and magnitudes of the maximum and minimum normal curvature.**

$u$ and $v$ in $\vec{S}(u, v)$. For quadrilateral patches, the bounds of the integrals are set to vary over the unit square, and the differential with respect to area is converted to differentials in $u$ and $v$

$$\int_0^1 \int_0^1 \left( \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 \right) \| S_u \times S_v \| \, du dv$$

where

$$\| S_u \times S_v \| = \sqrt{EG - F^2},$$

and

$$E = \vec{S}_u \cdot \vec{S}_u \quad F = \vec{S}_u \cdot \vec{S}_v \quad G = \vec{S}_v \cdot \vec{S}_v . \qquad (3)$$

The variables $E$, $F$, and $G$, are from the first fundamental form from differential geometry [11]. The principal curvatures $\kappa_1$ and $\kappa_2$ are the normal curvatures in the principle directions. Thus the problem of computing $d\kappa_n/d\hat{e}_1$ and $d\kappa_n/d\hat{e}_2$ becomes one of computing $d\kappa_1/d\hat{e}_1$ and $d\kappa_2/d\hat{e}_2$. First we find expressions for these in terms of derivatives taken in the parametric *directions*

$$\frac{d\kappa_1}{d\hat{e}_1} = \frac{d\kappa_1}{d\hat{u}} (\hat{e}_1 \cdot \hat{S}_u) + \frac{d\kappa_1}{d\hat{v}} (\hat{e}_1 \cdot \hat{S}_v)$$

$$\frac{d\kappa_2}{d\hat{e}_2} = \frac{d\kappa_2}{d\hat{u}} (\hat{e}_2 \cdot \hat{S}_u) + \frac{d\kappa_2}{d\hat{v}} (\hat{e}_2 \cdot \hat{S}_v)$$

where

$$\hat{S}_u = S_u / (\| S_u \|) \qquad \hat{S}_v = S_v / \| S_v \| .$$

Next we define the derivatives of $\kappa_1, \kappa_2$ taken in the parametric *directions* using parametric derivatives:

$$\frac{d\kappa_i}{d\hat{u}} = \frac{d\kappa_i}{du} \frac{1}{\| \vec{S}_u \|} \qquad \frac{d\kappa_i}{d\hat{v}} = \frac{d\kappa_i}{dv} \frac{1}{\| \vec{S}_v \|} .$$

Finally, the parametric derivatives of $\kappa_1$ and $\kappa_2$ are computed from an expression derived from the fact that the principle curvatures are the eigenvalues of the curvature tensor. The expression for the curvature tensor is

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix},$$

where

$$a_{11} = \frac{fF - eG}{EG - F^2} \qquad a_{21} = \frac{eF - fE}{EG - F^2}$$

$$a_{12} = \frac{gF - fG}{EG - F^2} \qquad a_{22} = \frac{fF - gE}{EG - F^2}$$

$$e = \hat{n} \cdot \hat{S}_{uu} \quad f = \hat{n} \cdot \hat{S}_{uv} \quad g = \hat{n} \cdot \hat{S}_{vv}. \qquad (4)$$

$E$, $F$, and $G$ are defined as in equation (3), $e$, $f$, and $g$ are the terms of the second fundamental form from differential geometry [11]. Since $\kappa_1$ and $\kappa_2$ are the eigenvalues of the curvature tensor, we get the following expression:

$$\kappa_i = \frac{a_{11} + a_{22} \pm \sqrt{a_{11}^2 + 4a_{12}a_{21} - 2a_{11}a_{22} + a_{22}^2}}{2}.$$

This expression is in terms of the surface parameters $u$ and $v$. Using the chain rule, it is simple to compute the required parametric derivatives, $d\kappa_i/du, d\kappa_i/dv$. Note that in computing the parametric derivatives of $e, f$, and $g$, it is helpful to have a simple way of computing $\hat{n}_u$ and $\hat{n}_v$:

$$\hat{n}_u = \kappa_1 (\hat{S}_u \cdot \hat{e}_1)\hat{e}_1 + \kappa_2 (\hat{S}_u \cdot \hat{e}_2)\hat{e}_2$$

$$\hat{n}_v = \kappa_1 (\hat{S}_v \cdot \hat{e}_1)\hat{e}_1 + \kappa_2 (\hat{S}_v \cdot \hat{e}_2)\hat{e}_2.$$

## 4.4 Numerical Integration

In section 4.3 we discussed a method for evaluating the quantity on the inside of the fairness integral (2). Because it is impractical to compute the integral analytically, we use numerical integration to evaluate the integral. Instead of using standard Gauss-Legendre quadrature, we use Lobatto quadrature [1]. Lobatto quadrature has approximately the same convergence and samples the *perimeter* of the integration domain:

$$\int_0^1 f(x)dx \approx w_1 f(0.0) + \sum_{i=2}^{n-1} w_i f(x_i) + w_n f(1.0)$$

We have found Lobatto's integration formula to be more effective than Gauss-Legendre quadrature for our application. As a default, we use ten integration points in each parametric direction, a satisfactory number for the modeling problems we have encountered so far. If the number of sample points is reduced, the surface might form a cusp or crease between sample points where the integrator will not "see" it.

The first ten sets of abscissas and weight factors for Lobatto's integration formula are tabulated in [1]. The computation of other sets of weights and abscissas requires finding the roots of the first derivative of a Legendre polynomial. Mathematica [33] may be used to generate larger tables. Because finding the roots of high order polynomials is difficult and prone to numerical errors, the results calculated for a new table should be checked for accuracy, e.g. verifying that the weights sum to 1. An alternative to computing higher order sets of weights and abscissas is to subdivide the domain and integrate over the subdomains.

## 4.5 Differentiation

During the optimization process, it is necessary to compute the gradient of the functional with respect to all the available degrees of freedom. When computing the curve network, analytical partial derivatives are used in conjunction with numerical integration to compute the gradient. In the case of surfaces, the functional is of such complexity that it is impractical to compute the gradient in this fashion. Instead we use central differences [7] to approximate the partial derivatives. The standard central difference formula for computing the derivative of $f(a)$ with respect to $a$ follows:

$$f'(a) = \frac{f(a+h) - f(a-h)}{2h}. \qquad (5)$$

In order to get accurate derivative estimates, it is necessary to choose the difference value $h$ carefully. An optimum value of $h$ balances the trade-off between the discretization error resulting from a large $h$ and an increasing relative roundoff error resulting from too small a value for $h$. The analysis used to compute $h$ is taken from [7]. First we find the approximate roundoff error in computing our functional. By computing the fairness functional, $FF$, in both single and double precision, we find the number of significant digits in the single precision calculation:

$$s_{\text{single}} = -\log_{10}\left|\frac{FF_{\text{single}} - FF_{\text{double}}}{FF_{\text{double}}}\right|.$$

The roundoff error $R$ in (5) is approximately

$$R = \pm\frac{2FF \times 10^{-s_{\text{single}}}}{2h}.$$

The discretization error $T$ is approximately

$$T = -\frac{1}{6}h^2.$$

To find the optimum $h$ we must minimize

$$\frac{FF \times 10^{-s_{\text{single}}}}{h} + \frac{1}{6}h^2. \qquad (6)$$

To find the value of $h$ for which (6) is a minimum, we differentiate with respect to $h$ and find the positive root.

$$-\frac{FF \times 10^{-s_{\text{single}}}}{h^2} + \frac{h}{3} = 0 \qquad h = \sqrt[3]{3FF \times 10^{-s_{\text{single}}}}.$$

Currently our calculations are carried out in double precision. Because we can only directly compute $s_{\text{single}} \approx 5.1$, we presume a value of $s_{\text{double}} = 9$ by extrapolation. We recalculate $h$ for each new value of the functional. For example, $FF = 10.0$ yields $h = 0.006$. Note that because the functional ($FF$) and its derivatives are computed on a patch by patch basis, the value of $h$ is also set on a patch by patch basis.

## 4.6 Tangent Continuity

In [10] DeRose sets forth the necessary and sufficient conditions for $G^1$ continuity. The $G^1$ conditions take the form of a series of formulas, $\text{eq}_i$, all of which must be zero for $G^1$ continuity to exist. Using the notation of DeRose, $N_{F'}, N_{G'}$ and $N_{H'}$ refer to the degree of the cross-boundary tangent functions $(F', G')$ and the degree of the tangent function $(H')$ along the boundary (Fig. 5). For example, a pair of abutting bi-quintic patches have $N_{F'} = N_{G'} = 5, N_{H'} = 4$ and formulas

$$\text{eq}_m = \sum_{j+k+l=m} \left|F_j^*, G_k^*, H_l^*\right| = 0$$
$$m = 0 \ldots D \qquad D = N_{F'} + N_{G'} + N_{H'} \qquad (7)$$

where

$$F_j^* = \binom{N_{F'}}{j} F'_j \qquad G_k^* = \binom{N_{G'}}{k} G'_k \qquad H_l^* = \binom{N_{H'}}{l} H'_l$$

where the $F'_j, G'_k$ and $H'_l$ are difference vectors as shown in Figure 5. The result per shared boundary, for our example of bi-quintic patches, is a set of fifteen equations, made up of one hundred distinct $3 \times 3$ determinants. The complexity of solving this

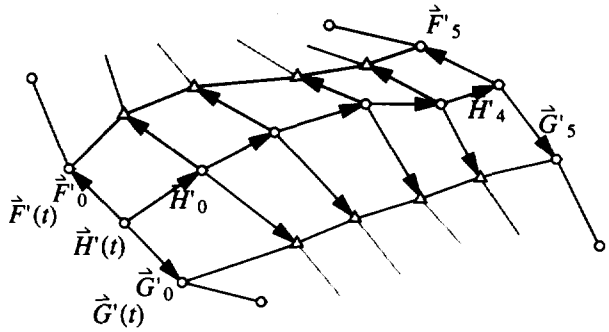



**Figure 5. Difference vectors for a pair of bi-quintic patches.**

system of equations has been outlined by a number of authors [14, 23, 29, 30]. This complexity arises because the corner interior control points (▲ Fig. 6) appear in the cross-boundary equations for multiple sides. This multiple appearance couples different patch-patch continuity equations and thereby creates a global system of equations with a very large number of variables. In the context of the optimization described here, it is impractical and unproductive to solve this explicitly.

We have described how the fairness functional is evaluated. We complete the objective function to be minimized by adding a penalty for lack of $G^1$ continuity. In formulating the penalty function, we square the terms from equation (7) yielding

$$\sum_{m=0}^{D} \Big( \sum_{j+k+l=m} \left| F_j^*, G_k^*, H_l^* \right| \Big)^2 . \qquad (8)$$

The penalty is computed for every patch-patch boundary and added to the fairness functional forming the objective function.

## 4.7 Initialization

The gradient descent scheme described in section 3 starts with an initial surface and iteratively refines that surface until the surface functional reaches a (local) minimum and an optimal surface is reached. In this section we discuss a method for finding a suitable initial surface. In terms of the desired optimization, the goal is to find an initial point in the proper "valley" of the solution space such that the desired surface is found as the minimal point in that valley. The optimization requires that initial values be provided for any parameters not explicitly set. The use of an optimized curve network initializes the control points on the perimeter of each patch. The interior control points are positioned

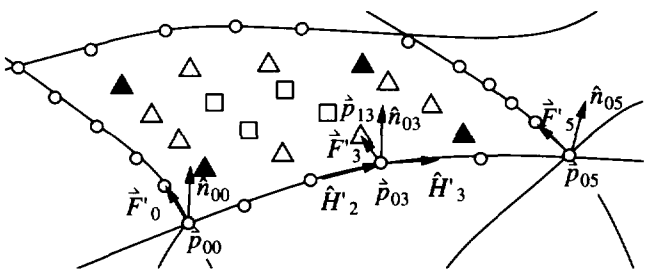



**Figure 6. The control points of a Bézier patch are grouped as the 20 perimeter control points (○), the 12 adjacent control points (△, ▲), and the 4 central control points (□). As an example of initialization, $\hat{p}_{13}$ is computed by linearly interpolating the surface normal vectors at the corners and the magnitudes of the corresponding difference vectors.**

so as to: 1) achieve approximate $G^1$ continuity and 2) set the high order derivatives at the patch corners equal to zero. The first step initializes the twelve control points adjacent to the perimeter (△, ▲), and the second step initializes the four points in the center of the patch (□) (Fig. 6). The heuristic used to position the control points adjacent to the perimeter linearly interpolates the normal vectors and the magnitudes of difference vectors. Figure 6 and equation (9) demonstrate the approach, with the calculation of $\hat{p}_{13}$:

$$\hat{p}_{13} = \hat{p}_{03} + \vec{F}'_3 \qquad \vec{F}'_3 = \left\| \vec{F}'_3 \right\| \hat{F}'_3$$

$$\hat{F}_3 = \frac{\hat{H}'_2 \times \hat{n}_{03} + \hat{H}'_3 \times \hat{n}_{03}}{\left\| \hat{H}'_2 \times \hat{n}_{03} + \hat{H}'_3 \times \hat{n}_{03} \right\|}$$

$$\left\| \vec{F}'_3 \right\| = \frac{2}{5} \left\| \vec{F}'_0 \right\| + \frac{3}{5} \left\| \vec{F}'_5 \right\| \qquad (9)$$

$$\hat{n}_{03} = \frac{\frac{2}{5}\hat{n}_{00} + \frac{3}{5}\hat{n}_{05}}{\left\| \frac{2}{5}\hat{n}_{00} + \frac{3}{5}\hat{n}_{05} \right\|} .$$

Alternatively, one could also use the construction due to Peters [23].

## 4.8 $G^2$ Vertices

The order of the derivatives in the surface functional indicate a requirement for $G^2$ continuity. Rather than imposing $G^2$ cross boundary continuity at the cost of many more equations to evaluate, we construct the network of patches to meet with $G^2$ continuity at the vertices only, and we maintain this continuity by construction during the minimization process. Because of this, the elements used in these optimizations are classified as nonconforming elements, and the "patch test" [34] must be used to guarantee convergence. Intuitively, because of the nature of the fairness functional, $G^2$ continuity tends to "propagate" along the patch-patch boundaries. If the surface elements are subdivided into smaller and more numerous elements and G2 vertex continuity is maintained, the overall surface converges on G2 continuity. A comparison of surfaces with and without $G^2$ vertex continuity shows those with $G^2$ continuity to have superior overall curvature distribution.

The construction used to maintain $G^2$ vertex continuity of the surface is a simple extension of the construction used to maintain $G^2$ compatibility of the MVC network (section 5). An additional step is carried out after the principle directions and curvatures at the vertices of the network have been established. This extra step of the construction requires that the twist vector of each incident patch corner be compatible with the established curvature. The restriction on $\hat{S}_{uv}$ is derived from the formulas for mean and Gaussian curvature:

$$\text{Gaussian} = \kappa_1 \kappa_2 = \frac{eg - f^2}{EG - F^2}$$

and

$$\text{mean} = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2} \frac{gE - 2fF + eG}{EG - F^2}$$

where $e, f, g, E, F, G$ are defined as in equations (3) and (4). The twist vector must be adjusted to satisfy $f = \hat{n} \cdot \hat{S}_{uv}$. This is accomplished by forcing the tip of $\hat{S}_{uv}$ to lie in the plane perpendicular to $\hat{n}$, offset by distance $f$ from the vertex

$$\hat{S}'_{uv} = \hat{S}_{uv} + (f - \hat{n} \cdot \hat{S}_{uv})\,\hat{n} .$$

$f$ can be computed from the values of $\kappa_1$, $\kappa_2$ and the first and second order derivatives of $\hat{S}(u, v)$.

## 5 MVC Network Computation

MVC networks are used to initialize the boundaries of the patches from which the surface is composed. Many of the techniques used in computing the network are used in the computation of an MVS where the initial shape of the curve network is allowed to change. In this section we outline the methods used to compute the $G^2$ curve network.

### 5.1 Network Representation and Continuity

The network of curves is defined via the second order parameters of a surface description at each vertex of the network and via a description of how each curve segment emerges from within the surfaces specified at its endpoints. Each surface is defined by the vertex position $\hat{p}$, a pair of conjugate directions, $\hat{w}_1, \hat{w}_2$, and the normal curvatures in those directions $\kappa_{w_1}, \kappa_{w_2}$. Conjugate directions are equivalent to principle directions in that, coupled with the associated curvatures, they fully characterize the curvature of a surface at a point [11]. Conjugate directions are more amenable to optimization because they do not have to be constrained to mutual orthogonality. The network is represented by quintic Hermite curves. These Hermite curves are defined by the positions and first two parametric derivatives at their endpoints. Each curve in the network is defined by the position $\hat{p}$, tangent direction $\hat{t}$, and three scalar parameters, $m, \alpha, c$, at each endpoint. The mapping from these values to the parameters defining the corresponding Hermite curve is

$$\hat{P} = \hat{p} \qquad \hat{P}' = m^2\hat{t} \qquad \hat{P}'' = m^4\hat{\kappa} + \alpha m^2\hat{t}$$

$$\hat{\kappa} = \kappa_n\hat{n} + c\hat{b} \qquad \hat{b} = \hat{n} \times \hat{t}$$

$$\kappa_n = \hat{t} \cdot [K] \cdot \hat{t}$$

$$[K] = \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \hat{n} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \kappa_{w_1} & 0 & 0 \\ 0 & \kappa_{w_2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \left( \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \hat{n} \end{bmatrix}^{-1} \right)^T \qquad (10)$$

Note that the curvature of the curve is the sum of two orthogonal components; $\kappa_n$, the component in the normal direction is a function of the surface at the vertex and the tangent direction of the curve at its end point; $c$, the component in the binormal direction is independent of the surface curvature at the vertex and represents the curvature of the curve "within the surface."

During the optimization process, those variables not fixed by constraints are iteratively adjusted to minimize the MVC functional (1). At each iteration step, $\hat{w}_1$ and $\hat{w}_2$ are renormalized, and $\hat{t}$ is projected onto the plane spanned by $\hat{w}_1, \hat{w}_2$ and also renormalized. It is this normalization step in combination with the construction outlined in equation (10) that guarantees $G^2$ continuity is maintained.

### 5.2 Network Initialization

The curve network must be initialized to some reasonable values before optimization can proceed. First a vertex normal vector is initialized, then the tangent vectors of the incident curves are computed, next the principle directions and curvatures are defined, and finally each curve's scalar coefficients are initialized.

The vertex normal is initialized as an average of the incident face normals weighted inversely proportional to the area of the incident face, i.e. the smaller the face the greater its influence on the vertex normal [5]. The tangent vectors of curves incident to a vertex are set to the direction of the incident chords projected onto the plane defined by the vertex position $\hat{p}_i$ and the normal $\hat{n}$ (Fig. 7).

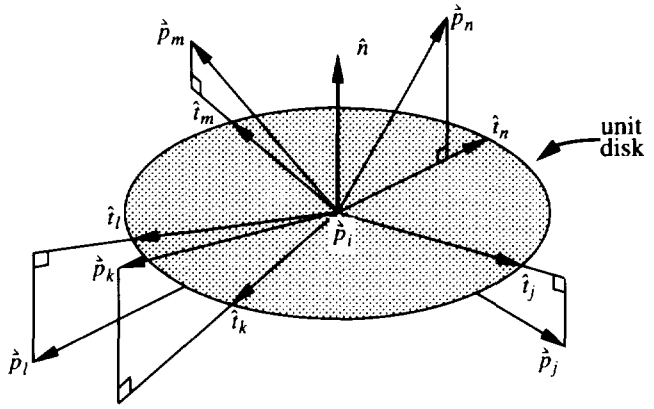



**Figure 7. Tangent initialization. The projection of incident chords onto the plane defined by the normal.**

Once vertex normal vectors and incident tangent directions have been computed, the principle curvatures and principle directions at a vertex are calculated. Both Calladine [5] and Todd and McLeod [31] describe approaches for estimating the curvature of polyhedral surfaces. Calladine's method only estimates Gaussian curvature. Todd and McLeod require that a pairing be established among the points neighboring a point; this is not possible at vertices of odd order. At even order vertices, it remains problematic since the results vary greatly depending on the pairing chosen; logically opposite curves are not always appropriate partners.

Our approach uses a least squares fit of sample tangent directions and normal curvatures to compute the principle directions and curvatures. The initialization of these values is very important to the speed of convergence. First consider the situation shown in Figure 7. A vertex is shown with a number of incident edges. For each edge we calculate the curvature implied by that edge emanating from the vertex. Starting with edge $\hat{p}_i\hat{p}_n$ we reflect $\hat{p}_n$ through the normal and fit a circle through $\hat{p}'_n, \hat{p}_i$ and $\hat{p}_n$. The radius of the resulting circle is the radius of curvature (Fig. 8). Repeating this procedure for each of the incident edges

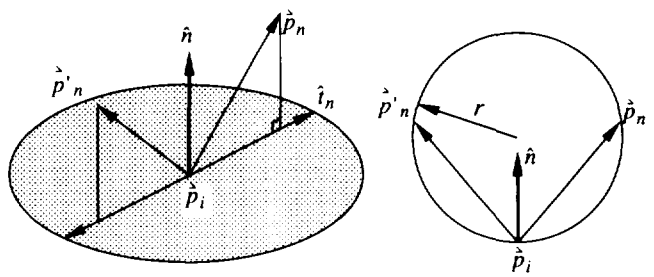



**Figure 8. Calculating an approximate radius of curvature in the direction of $\hat{t}_n$.**

provides a set of sample tangent directions and normal curvatures. This set is used to compute a least squares fit for the principle directions and principle curvatures of the surface at the vertex as follows.

We start with the expression for normal curvature expressed with respect to any convenient basis in the plane defined by the normal,

$$\kappa_n = \hat{t} \cdot [K] \cdot \hat{t}$$

$$[K] = \begin{bmatrix} \hat{e}_{1,x} & \hat{e}_{1,y} \\ -\hat{e}_{1,y} & \hat{e}_{1,x} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \cdot \left( \begin{bmatrix} \hat{e}_{1,x} & \hat{e}_{1,y} \\ -\hat{e}_{1,y} & \hat{e}_{1,x} \end{bmatrix}^{-1} \right)^{T}$$

and extract the tangent components, to produce an over determined set of linear equations:

$$\begin{bmatrix} \hat{t}_{0,x}^2 & \hat{t}_{0,x}\hat{t}_{0,y} & \hat{t}_{0,y}^2 \\ \hat{t}_{1,x}^2 & \hat{t}_{1,x}\hat{t}_{1,y} & \hat{t}_{1,y}^2 \\ \cdot & \cdot & \cdot \\ \hat{t}_{m,x}^2 & \hat{t}_{m,x}\hat{t}_{m,y} & \hat{t}_{m,y}^2 \end{bmatrix} \cdot \begin{bmatrix} \hat{e}_{1,x}^2 \kappa_{\hat{e}_1} + \hat{e}_{1,y}^2 \kappa_{\hat{e}_2} \\ 2\hat{e}_{1,x}\hat{e}_{1,y}(\kappa_{\hat{e}_1} - \kappa_{\hat{e}_2}) \\ \hat{e}_{1,x}^2 \kappa_{\hat{e}_2} + \hat{e}_{1,y}^2 \kappa_{\hat{e}_1} \end{bmatrix} = \begin{bmatrix} \kappa_{n,0} \\ \kappa_{n,1} \\ \cdot \\ \kappa_{n,m} \end{bmatrix}$$

$$Ax = b. \qquad (11)$$

The general formula for computing the least squares solution to this type of system is $A^T A\bar{x} = A^T b$, where $\bar{x}$ is the least squares solution for "$x$" in equation (11). Having solved for $\bar{x}$ we have three equations and four unknowns

$$\begin{bmatrix} \hat{e}_{1,x}^2 \kappa_1 + \hat{e}_{1,y}^2 \kappa_2 \\ 2\hat{e}_{1,x}\hat{e}_{1,y}(\kappa_1 - \kappa_2) \\ \hat{e}_{1,x}^2 \kappa_2 + \hat{e}_{1,y}^2 \kappa_1 \end{bmatrix} = \begin{bmatrix} \bar{x}_0 \\ \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}.$$

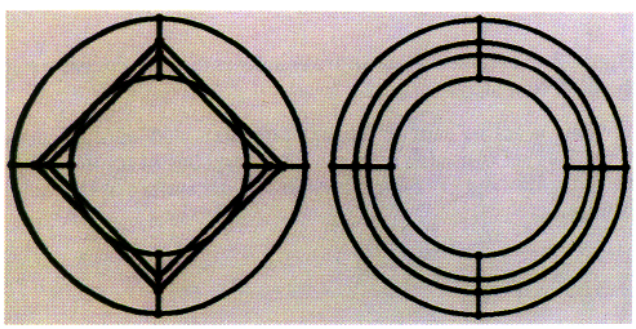
Adding the fact that $\hat{e}_{1,x}^2 + \hat{e}_{1,y}^2 = 1$, allows us to solve for the principle directions and principle curvatures.

To complete the initialization of the network, the scalars associated with each curve are set as follows: $m$ is set to the chord length, and $\alpha$, $c$ are set to zero.

### 5.3 Optional Network Constraints

Since the quality of the network directly impacts the quality of the resulting surface, we present an optional heuristic constraint. A very successful method for improving network quality is to force pairs of curve segments incident to a common vertex of the network to join with $G^2$ continuity. Pairs of curves are made $G^1$ continuous by forcing them to share tangent vectors. $G^2$ continuity is imposed by forcing the curves to also share the bi-normal component, $c$, from (10).

During initialization, shared tangents are set to the average of the individual tangents computed by chord projection. Figure 9 illustrates curve continuity applied to 16 regularly spaced points on the surface of a torus. Figure 9b illustrates the improvement to the network when $G^2$ continuity is imposed.



(a) (b)

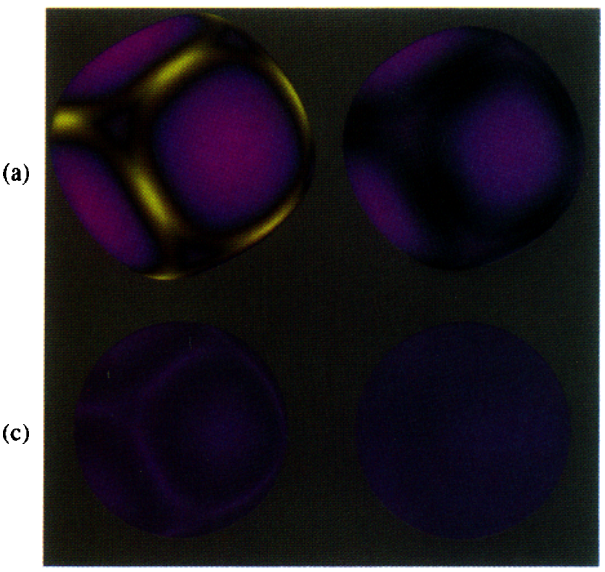
**Figure 9. A network through points on a torus (a) with $G^0$ and (b) with $G^2$ continuous curves through vertices.**

## 6 Examples: A Comparison of Functionals

In order to evaluate the quality and usefulness of MVS, we examine a few interpolation and design problems. Special rendering techniques are used to assist in the evaluation of the quality of these surfaces. Functional shading is used to examine the distribution of curvature. In this case, mean and Gaussian curvature are used to index into a color map. Lines of reflection are used to demonstrate G1 and G2 continuity [24]. They are generated by assuming that the surface is highly reflective and is place inside a large box with vertical lines drawn on its walls. The surface is then rendered using environment mapping [3, 12]. Generally, smooth surfaces have smooth lines of reflection. G2 (surface curvature) discontinuities appear as kinks or G1 discontinuities in the lines of reflection. G1 (surface tangent) discontinuities cause $G^0$ discontinuities in the lines of reflection.

### 6.1 Spheres

In Figure 10 we use functional shading to compare the MVS



(a) (b) (c) (d)

**Figure 10. Surfaces are computed interpolating the 8 corners of a cube. These illustrate the differences among different methods. Pseudo-coloring according to mean and Gaussian curvature exposes the differences between objective functions. (a) - $G^1$ penalty, (b) - linearized strain energy, (c) - strain energy, (d) - MVS.**

functional with three other functionals. In (a) only the $G^1$ penalty function was minimized when fitting a surface to the points of a cube. (b) illustrates the result of using a linearized approximation to strain energy; curvature distribution is improved. Next, (c) true strain energy is minimized producing a surface with fairly uniform curvature. Finally, in Fig. 10d an MVS surface fitted to the corners of a cube produces a very close approximation to a sphere.

### 6.2 Three Handles

Figure 11 illustrates the application of MVC to a more complicated example. Fig. 11a is the MVC network interpolated to create the $G^1$ MVS. (b) illustrates the surface rendered with lines of reflection. In the lower half of Figure 11, strain energy (c) and the

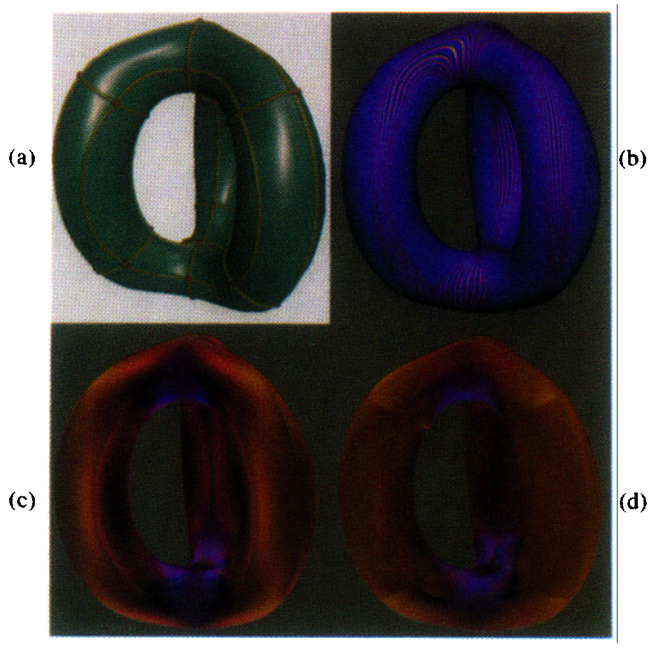



**Figure 11. Three Handles. Minimization of strain energy is compared with the MVS starting with the same MVC network. (a) - the MVC network. (b) - the MVS rendered with lines of reflection indicating $G^1$, ~$G^2$ continuity. (c) - the surface minimizing strain energy. (d) - the MVS interpolating the same network of curves. (c) and (d) are rendered using functional shading to expose the differences in curvature distribution.**

MVS functional (d) are compared. The differences are subtle, curvature varies more smoothly and is distributed more evenly over the MVS surface.

### 6.3 Tetrahedral Frame

As our last example, an MVS surface is fit to a tetrahedral frame, (Fig. 12). (a) illustrates the fitted surface with its MVC network. (b) shows the parameterization of individual patches demarcated by black borders. In (c) the surface is rendered with lines of reflection. Finally, (d) is a simple lighted rendering of the surface. This last example demonstrates the versatility and power of MVS to solve a very difficult interpolation problem.

## 7 Conclusions

Constructing a network of $G^1$ continuous surface patches is known to be a difficult task, and it is even harder to shape such a network into a satisfactory surface. The use of a general optimization procedure with suitable penalty functions greatly simplifies both tasks.

In this paper we have described a conceptually simple yet powerful technique for surface modeling. Nonlinear optimization is used to minimize a fairness functional while maintaining geometric and continuity constraints. The functional of choice is the variation of curvature. This choice has the advantage that it leads to regular shapes commonly used in geometric modeling; spheres, cylinders, and tori are formed in response to a compatible set of constraints. The minimization of our fairness functional also produces very fair free-form surfaces. This allows the designer to specify technical and artistic shapes in a very natural way for a given design problem.


**Figure 12. TetraThing. (a) - the interpolated MVC network. (b) - individual patches demarcated by black boundaries. (c) - lines of reflection indicating $G^1$, ~$G^2$ continuity. (d) - a simple shaded rendering.**

Surfaces are represented by a patchwork of quintic polynomial Bézier patches. The use of quintic patches makes the satisfaction of geometric constraints simple, direct, and exact. The use of a quintic patch also tends to minimize the *number* of patches needed to solve a given problem. The use of Bézier patches eases numerical problems and simplifies communication with other modeling systems.

The techniques described here are computationally very expensive. They have only become practical because of the wide availability of very fast workstations. As computers increase in speed, these techniques will become even more attractive.

## Acknowledgments

The authors thank Silicon Graphics Inc. for its generous support. We would also like to thank Tony DeRose and the reviewers for their many helpful comments on this paper.

## References

1. Abramowitz, M. and Stegun, I.A. (editors), *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, Inc., New York, N.Y. 1972.
2. Birkhoff, G.D., *Aesthetic Measure*, Harvard University, Cambridge, Mass., 1933.
3. Blinn, J.F., Newell, M.E., Texture and Reflection in Computer Generated Images, *Communications of the ACM* 19, 10 (Oct. 1976), 542-547.
4. Böhm, W., On Cyclides in Geometric Modeling, *Computer Aided Geometric Design* 7 (1990), 243-255.
5. Calladine, C.R., Gaussian Curvature and Shell Structures. In *The Mathematics of Surfaces*, Clarendon, Oxford, England, 1986, pp. 179-196.
6. Celniker, G. and Gossard, D., Deformable Curve and Surface Finite-Elements for Free-Form Shape Design, Proceedings of SIGGRAPH '91 (Las Vegas, Nevada, July 29-August 2, 1991). In *Computer Graphics* 25, 4 (July 1991), 257-266.

7. Conte, S.D. and de Boor, C. *Elementary Numerical Analysis, An Algorithmic Approach*, McGraw-Hill, New York, N.Y., 1980.
8. Courant, R. and Hilbert, D., *Methods of Mathematical Physics, Volume 1*, Wiley-InterScience, New York, N.Y., 1953.
9. DeRose, T.D. and Mann, S., An Approximately G1 Cubic Surface Interpolation. To appear in the Proceedings of the 1991 Biri Conference.
10. DeRose, T.D., Necessary and Sufficient Conditions for Tangent Plane Continuity of Bézier Surfaces, *Computer Aided Geometric Design* 7 (1990), 165-179.
11. Do Carmo, M.P., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.
12. Greene, N., Environment Mapping and Other Applications of World Projections, *IEEE Computer Graphics and Applications* 6, 11 (Nov. 1986), 21-29.
13. Hagen, H. and Schulze, G., Automatic Smoothing with Geometric Surface Patches, *Computer Aided Geometric Design* 4, (1987), 231-236.
14. Jones, A.K., Nonrectangular Surface Patches with Curvature Continuity, *Computer Aided Design* 20, 6 (Jul./Aug. 1988), 325-335.
15. Kallay, M. and Ravani, B., Optimal Twist Vectors as a Tool for Interpolating a Network of Curves with a Minimum Energy Surface, *Computer Aided Geometric Design* 7, 6 (1990), 465-473.
16. Lott, N.J. and Pullin, D.L., Method for Fairing B-spline Surfaces, *Computer Aided Design* 20, 10 (Dec. 1988), 597-604.
17. Mann, S., Loop, C., Lounsbery, M., Meyers, D., Painter, J., DeRose, T., and Sloan, K., A Survey of Parametric Scattered Data Fitting Using Triangular Interpolants. In *Curve and Surface Modeling*, Hagen, H. (editor), SIAM.
18. Mehlum, E., Nonlinear Splines. In *Computer Aided Geometric Design*, R.E. Barnhill and R.F. Riesenfeld (editors), Academic, Orlando, Florida, 1974, pp. 173-207.
19. Moreton, H.P. and Séquin, C.H., Surface Design with Minimum Energy Networks. In *Proceeding of the Symposium on Solid Modeling Foundations and CAD/CAM Applications* (Austin, Tex., June 5-7). ACM, New York, N.Y., 1991, pp. 291-301.
20. Nielson, G.M., A Method for Interpolating Scattered Data Based Upon a Minimum Norm Network, *Mathematics of Computation* 40, 161 (1983), 253-271.
21. Nowacki, H. and Reese, D., Design and Fairing of Ship Surfaces. In *Surfaces in Computer Aided Geometric Design*, Barnhill, R.E. and Boehm, W. (editors), North-Holland, Amsterdam, The Netherlands 1983, pp. 121-134.
22. Peters, J., Local Smooth Surface Interpolation: A Classification, *Computer Aided Geometric Design* 7 (1990), 191-195.
23. Peters, J., Smooth Interpolation of a Mesh of Curves, *Constructive Approximation* 7 (1991), 221-247.
24. Poeschl, T., Detecting Surface Irregularities Using Isophotes, *Computer Aided Geometric Design*, 1 (1984), 163-168.
25. Pottmann, H., Scattered Data Interpolation Based upon Generalized Minimum Norm Networks, Preprint Nr. 1232, Technische Hochschule, Darmstadt, May 1989.
26. Pramila, A., Ship Hull Surface Using Finite Elements, *International Shipbuilding Progress* 25, 284 (1978), 97-107.
27. Pratt, M.J., Cyclides in Computer Aided Geometric Design, *Computer Aided Geometric Design* 7 (1990), 221-242.
28. Rando, T. and Roulier, J.A., Designing Faired Parametric Surfaces, *Computer Aided Design* 23, 7 (Sept. 1991), 492-497.
29. Sarraga, R.F., $G^1$ Interpolation of Generally Unrestricted Cubic Bézier Curves, *Computer Aided Geometric Design* 6, 2 (1987), 23-40.
30. Shirman, L. and Séquin, C.H., Local Surface Interpolation with Bézier Patches, *Computer Aided Geometric Design* 4 (1987), 279-295.
31. Todd P.H. and McLeod, R.J.Y., Numerical Estimation of the Curvature of Surfaces, *Computer Aided Design* 18, 1 (Jan./Feb. 1986), 33-37.
32. Williams, C.J.K., Use of Structural Analogy in Generation of Smooth Surfaces for Engineering Purposes, *Computer Aided Design* 19, 6 (Jul./Aug. 1987), 310-322.
33. Wolfram, S., *Mathematica, A System for Doing Mathematics by Computer, 2nd ed.*, Addison-Wesley, Redwood City, California, 1991.
34. Zienkiewicz, O.C., *The Finite Element Method*, McGraw-Hill, London, England, 1977.