6
Minimum Variation Surfaces

In this chapter we describe the calculation of the Minimum Variation Surface (MVS). A series of examples demonstrates the superior quality of MVS compared with surfaces created by other techniques.

In the computer aided design of curved surfaces there is a wide range of requirements. While it is necessary to model regular shapes such as cylinders, cones, tori, and spheres, it is also important that free-form shapes be modeled with ease. Often, it is also necessary that surfaces exactly meet a set of positional, tangent, and curvature constraints. In all cases, surface fairness is of great importance. The minimum variation surfaces (MVS) meet all these requirements.

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 [15].

Traditional work on the *fairness of surfaces* focuses on strain energy, minimizing the area integral of the sum of the principal curvatures squared [91,123,145,209]

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

Our approach minimizes the *variation* of curvature, rather than its magnitude. We minimize the area integral of the sum of the squared magnitudes of the derivatives of the normal curvatures taken in the principal directions:

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

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 [17,173] which 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. Also like MVC functional, MVS are *invariant* under rigid body transformations and uniform scaling, but similarly the MVS *changes* functional value with changes in scale. Specifically, for a change in scale by a factor of γ, the MVS functional changes value by a factor of $1/\gamma^2$. We have designed[1] a scale invariant MVS functional, similar to the SI-MVC functional (4.3), that counters the inverse square factor by multiplying the standard MVS functional by the area of the surface, which varies as the square of the scale:

$$\left(\int dA\right)\left(\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 \, dA\right) \qquad \textit{the SI-MVS functional} \qquad \textbf{(6.1)}$$

MVS 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 permit, the resulting surfaces form portions 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 Klein bottle is specified with ease; only twelve point-tangent constraints are used to model the surface shown in Figure 6.1.

6.1 MV-Surface Construction

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. 6.2); 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 expressed by a network of straight edges. These are then replaced with suitable curve segments, and an optimized network is computed composed of minimum variation curves (MVN) subject to the specified geometric constraints and the additional constraint that the curve segments meet with second order geometric continuity, G^2, at the vertices. Finally, an interpolatory minimum variation surface (MVS) is computed, interpolating the MVN 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 MVN as a starting point and modify its

[1] The SI-MVS will be implemented as part of future research.

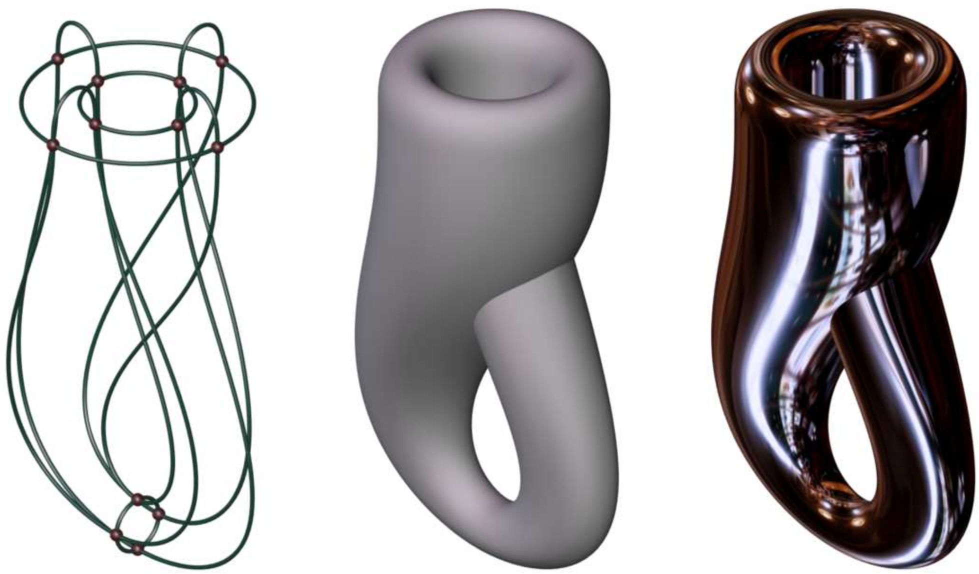


Figure 6.1. A Klein bottle.

This complex surface is defined by twelve constraint sets. ① The initial MVN used to model the bottle. ② The resulting surface with diffuse shading. ③ The surface rendered using environment mapping to "highlight" its quality.

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 MVN resulting from a given constraint set do not always lie in the MVS resulting from the same set of constraints. We examine the flexible MVN and the associated improvement in surface quality in section 6.5.

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. Currently, the user must add additional vertices and edges to the network so that it 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.

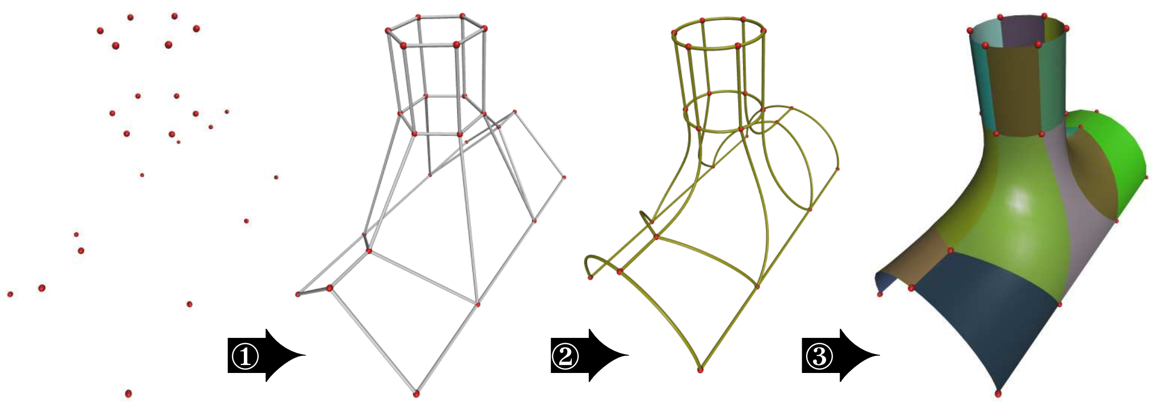


Figure 6.2. The Blend of Two Pipes.

Pipes are blended in three steps: ① The connectivity of the constraints is established. ② Smooth curves are fit to the constraints. ③ Surface patches are fit to the curve network.

Based on the MVN, the computation of the MVS interpolatory surface is accomplished using constrained optimization. The geometric constraints are imposed by constructions similar to those used in the calculation of the network. Inter-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. In addition, a similar penalty function may be applied to impose curvature continuity. The use of penalty functions alone does not guarantee perfect continuity. Exact G^1 continuity may be achieved in a subsequent phase of optimization using Lagrange multipliers [41] or using the continuum method, a continuous reduction to zero of the weight of the MVS term in the functional. This reduction of the MVS functional to zero is equivalent to increasing the G^1 penalty to infinity. In practice, it is not always necessary to resort to this second phase because the surfaces resulting from the first phase are of high quality and sufficiently close to being tangent continuous. Mann and DeRose have shown this type of *approximate* tangent continuity to be sufficient and, in fact, desirable in some applications [50].

6.2 Representation and Computation

As described in section 6.1, 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 continuous MVN. 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 6.5.4). Chapter 5 provides the details of MVN 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.

6.2.1 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 have to be at least [bi-]quintic patches. Peters [161] 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 6.2.6).

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 one of the tangent continuity conditions we use is 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®.

6.2.2 Parametric Functionals

The fairness functional for surfaces (6.2) is defined in terms of an area integral. To evaluate the functional and its gradient in the context of the parametric polynomial surface patches described in section 6.2.1, 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

$$\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 \, dA \qquad \textbf{(6.2)}$$

is converted to integrals of functions of the independent parameters 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

$$E = \vec{S}_u \cdot \vec{S}_u \quad F = \vec{S}_u \cdot \vec{S}_v \quad G = \vec{S}_v \cdot \vec{S}_v. \qquad \textbf{(6.3)}$$

The variables *E, F,* and *G*, are from the first fundamental form from differential geometry [52]. The principal curvatures κ_1 and κ_2 are the normal curvatures in the principal 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 direction of the parametric derivatives

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

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

where

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

Next we define the derivatives of κ_1, κ_2 taken in the direction of the parametric derivatives using derivatives with respect to the surface parameters u and v:

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

Finally, the parametric derivatives of κ_1 and κ_2 are computed from an expression derived from the fact that the principal 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

$$\begin{aligned} a_{11} &= \frac{fF - eG}{EG - F^2} & a_{21} &= \frac{eF - fE}{EG - F^2} \\ a_{12} &= \frac{gF - fG}{EG - F^2} & a_{22} &= \frac{fF - gE}{EG - F^2} \\ e &= \hat{n} \cdot \vec{S}_{uu} \quad f = \hat{n} \cdot \vec{S}_{uv} & g &= \hat{n} \cdot \vec{S}_{vv}. \end{aligned} \tag{6.4}$$

E, F, and *G* are defined as in equation (6.3), *e, f,* and *g* are the terms of the second fundamental form from differential geometry [52].

Since κ_1 and κ_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 $\hat{n}_v$:

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

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

6.2.3 Numerical Integration

In section 6.2.2 we discussed a method for evaluating the expression on the inside of the fairness integral. 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 as effective as Gauss-Legendre quadrature for our application, and since it samples the perimeter, its evaluations may be used to compute continuity penalty functions, see (6.2.5). As a default, we use 20 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 [210] 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. by verifying that the weights sum to 1.

6.2.4 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 [39] to approximate the partial derivatives. The standard central difference formula for computing the derivative of $f(a)$ with respect to a follows:

$$f'(a) = \frac{f(a+h) - f(a-h)}{2h}. \tag{6.5}$$

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. We initialize h to a reasonable value based on the number of significant bits in the computation of the functional and subsequently update h periodically based on the observed number of significant bits in the difference computation. An accepted rule of thumb dictates that about half the significant bits should cancel out when the difference is computed [ref].

First we find the number of significant bits in computing our functional. By computing the fairness functional, *F,* in both single and double precision, we find the number of significant bits in the single precision calculation:

$$s_{\text{single}} = -\log_2\left|\frac{F_{\text{single}} - F_{\text{double}}}{F_{\text{double}}}\right|. \tag{6.6}$$

Currently our calculations are carried out in double precision. We compute s_{single} using equation (6.6) applied to a sample of MVS calculation problems. s_{single} varies somewhat both from interpolation problem to interpolation problem and from the beginning to the end of an optimization. Averaging several sample values we compute a value of $s_{\text{single}} \approx 10.5$. Because we can only directly compute s_{single}, we compute the value of s_{double} by assuming a gain of approximately 29 bits of precision corresponding to the 29 additional fractional bits available in double precision. Using this computation we get $s_{\text{double}} \approx 39.5$.

At the start of an optimization h is initialized to $h = F \times 2^{-16}$. After each difference computation we compute the number of significant bits in the difference value d and adjust h if the number of significant bits is not within set bounds. The number of significant bits in the difference computation is computed

$$\begin{aligned} d &= s_{\text{double}} + \log_2\left(\frac{2|f(a) - f(a+h)|}{|(f(a) + f(a+h))|}\right) \\ &= s_{\text{double}} + \log_2(2|f(a) - f(a+h)|) - \log_2(|(f(a) + f(a+h))|) \end{aligned}.$$

The difference value h is adjusted as follows

$$\begin{aligned} &\text{if}\left(d < \frac{s_{\text{double}}}{3}\right) h = 1.5h \\ &\text{if}\left(d > \frac{2s_{\text{double}}}{3}\right) h = 0.75h \end{aligned}.$$

Though the specific values of 1.5 and 0.75 have been chosen empirically, it is important that they not be reciprocals of each other. If they were reciprocals, we would run the risk of h oscillating between two values, one too large and one too small. The value of a finite difference step size is usually fixed for the duration of an optimization; our approach of continually adjusting h has the advantage that h adapts to different stages in the optimization, depending on the functional and its gradient. Also note that since different degrees of freedom have differing numerical behavior, a custom finite difference value h is associated with each DOF. Finally, an inexpensive “trick” is used to preserve as much precision as possible when adding the difference value to the associated DOF to produce the perturbed DOF q

$$q = a + h \qquad h = q - a.$$

This adjusts h so that it corresponds precisely to the difference between a and the perturbed value of a, q.

6.2.5 Continuity by Penalty

In sections (3.3.2.1) and (3.3.2.2) we outline work that highlights the difficulties encountered when trying to construct networks of polynomial patches meeting with specified geometric continuity. These problems can be circumvented by incorporating the continuity problem into the optimization used to shape surfaces. We achieve this by adding to the objective function a penalty function for lack of continuity. This penalty function evaluates to zero when surfaces meet continuously and increases sharply with the magnitude of discontinuity. Because of the nature of gradient-based optimization, it is important that this penalty function have zero gradient at its minimum point. In section 6.2.5.1 we discuss the formulation of a penalty function for tangent continuity and in section 6.2.5.2 we present a similar penalty function for G^2 continuity.

6.2.5.1 Tangent Continuity

In this section we present two possible penalty functions for incorporating tangent continuity into a surface optimization. The first method is based on the work of DeRose [49]; he sets forth the necessary and sufficient conditions for G^1 continuity. The second method is based on the numerical integration of a cross-boundary discontinuity function.

DeRose’s G^1 conditions take the form of a series of formulas, eq_i, all of which must be zero for G^1 continuity to exist. Using his notation, $N_{F'}$, $N_{G'}$ 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. 6.3). For example, a pair of abutting bi-quintic patches have $N_{F'} = N_{G'} = 5$, $N_{H'} = 4$ and formulas

$$\begin{gathered} \mathrm{eq}_m = \sum_{j+k+l=m} \left| F_j^*, G_k^*, H_l^* \right| = 0 \\ m = 0 \ldots D \qquad D = N_{F'} + N_{G'} + N_{H'} \end{gathered} \tag{6.7}$$

where

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

where the F'_j, G'_k and H'_l are difference vectors as shown in Figure 6.3. The result per shared boundary, for our example of bi-quintic patches, is a set of fifteen equations, made up of one hundred distinct 3×3 determinants. The complexity of solving this system of equations (6.7) has been outlined by a number of authors [108,161,188,199]. This complexity arises because the corner interior control points 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; which is why we use penalty functions.

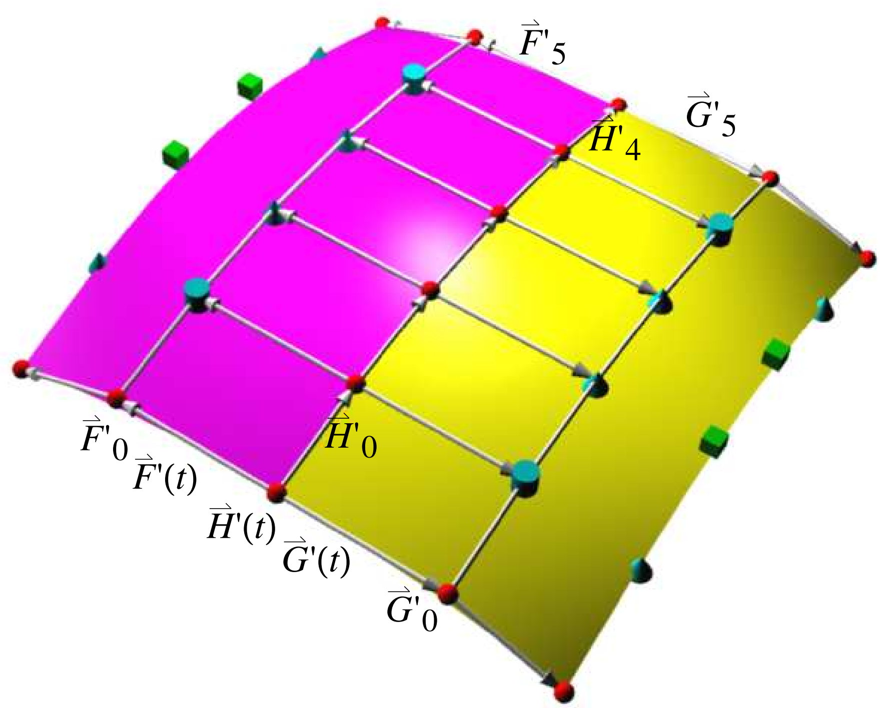


Figure 6.3. Difference vectors.

The difference vectors on a patch are formed by subtracting adjacent points, etc. for a pair of bi-quintic patches.

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 (6.7) and add yielding

$$P_{G^1} = \sum_{m=0}^{D} \Big(\sum_{j+k+l=m} \left| F_j^*, G_k^*, H_l^* \right| \Big)^2 . \qquad \textbf{(6.8)}$$

The penalty is computed for every patch-patch boundary and added to the fairness functional forming the objective function. Note that (6.8) does have the property that its gradient evaluates to zero at its minimum (zero).

An attractive alternative to a penalty based on equation (6.7) is the integration of a cross-boundary discontinuity function. We may form such a penalty function from parameterization independent measures by computing the surface normal along either side of the patch-patch boundary. One such function integrates a function of the cosine of the angle formed by the normals

$$P_{G^1} = (1 - (\hat{n}_a \cdot \hat{n}_b))^2 \qquad (\hat{n}_a \cdot \hat{n}_b) \to 1 \qquad \nabla P_{G^1} = 0. \qquad \textbf{(6.9)}$$

This equation is satisfactory except that it is not strongly positive for large discontinuities, in fact its magnitude is limited to 2.0. Equation (6.10) computes a function of (6.9) such that its values range from 0 to ∞.

$$\begin{aligned} & P_{G^1} = \left(\frac{1 - (\hat{n}_a \cdot \hat{n}_b)}{1 + (\hat{n}_a \cdot \hat{n}_b)} \right)^2 \\ & (\hat{n}_a \cdot \hat{n}_b) \to 1 \qquad \nabla P_{G^1} = 0 \\ & (\hat{n}_a \cdot \hat{n}_b) \to -1 \qquad P_{G^1} = \infty \end{aligned} \qquad \textbf{(6.10)}$$

Since the normal vector is computed as part of evaluating the fairness functional, and the fairness functional is integrated using Lobatto quadrature, we use the same quadrature rule to integrate (6.10). In order to show that a polynomial of order n is identically zero, it is only necessary to show that it is equal to zero at n distinct locations. The roots of equation (6.10) are identical to the roots of the determinant of tangent functions F', G' and H', a polynomial of degree 14 in the biquintic-biquintic case. Thus if (6.10) is sampled at least 15 times, we are guaranteed that if the result is zero then the function itself is also zero. This latter G^1 penalty function P_{G^1} is more efficient to compute, and it is independent of parameterization.

6.2.5.2 Curvature Continuity

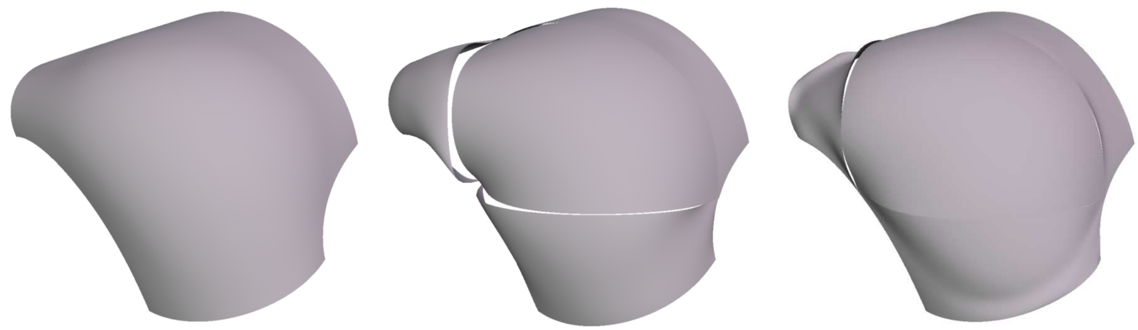
The problem of constructing a G^2 continuous network of polynomial patches is extremely difficult to solve exactly due to the second order analogue of the twist compatibility problem (3.3.2.2). Our penalty based method avoids this problem by using methods similar to the G^1 penalty function to push the optimization toward a G^2 continuous solution. We construct the penalty function from the geometric measures of curvature, $\kappa_1, \kappa_2, \hat{e}_1, \hat{e}_2$. Pegna and Wolter [155] show that, given a pair of patches that meet with G^1 continuity, patches need only have equal normal curvatures in a single transverse direction in order to assure G^2 continuity. From this observation we derive a penalty function measuring the difference in normal curvature in the direction perpendicular to the patch-patch boundary. The normal curvature κ_n in the direction $\hat{t}$ is $\kappa_n = \kappa_1 (\hat{e}_1 \cdot \hat{t})^2 + \kappa_2 (\hat{e}_2 \cdot \hat{t})^2$, therefore, because $\hat{e}_1 \perp \hat{e}_2$, the normal curvature in the direction $\perp\hat{t}$ is $\perp\kappa_n = \kappa_2 (\hat{e}_1 \cdot \hat{t})^2 + \kappa_1 (\hat{e}_2 \cdot \hat{t})^2$. Referring to $\perp\kappa_n$ on adjacent patches as κ_a, κ_b, our G^2 penalty function is $P_{G^2} = (\kappa_a - \kappa_b)^2$. Note that this function has the necessary property that the gradient is zero when $\kappa_a - \kappa_b = 0$. In order to determine the number of samples required to accurately integrate this function, we could use a similar argument based on the order of the polynomials involved in its evaluation. Since the polynomial order is prohibitively high, we sample this function with the same density as the G^1 penalty and the MVS functional. We have found this to significantly improve the quality of problematic surfaces. For example, the suitcase corner blend in Figure 1.1 has large curvature discontinuities as shown in Figure 6.4②. In Figure 6.4③ we see the results of applying our penalty for curvature discontinuity; the resulting surface is nearly curvature continuous; Fig. 6.4① is a shaded rendering of the G^2 surface.

6.2.5.3 Continuum Methods

The G^1 (6.2.5.1) and G^2 (6.2.5.2) penalty functions are combined with the MVS functional to form the objective function:

$$\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 dA + \int P_{G^1} ds + \int P_{G^2} ds \qquad \textbf{(6.11)}$$

Minimizing this objective function alone does not guarantee G^2 or even G^1 continuity. This is because using a single quintic patch in each network opening may not provide sufficient degrees of freedom to accurately represent the theoretical minimum variation surface. Two solutions to this problem are possible. First, each patch of the network may



① ② ③

Figure 6.4. A Curvature Continuous Suitcase Corner

① a shaded rendering of the G^2 surface. ② and ③ are displayed as functional offset surfaces, with the offset proportional to Gaussian curvature (see section 6.4). ② illustrates a G^1 suitcase corner with curvature discontinuities appearing as large gaps in the offset surface. ③ illustrates a nearly G^2 suitcase corner achieved using P_{G^2} as a penalty function.

be subdivided into multiple patches adding flexible arcs to the MVN, and augmented with the resulting additional degrees of freedom, the optimization may proceed. Second, we may use continuum methods to achieve final continuity. A continuum method computes a series of solutions to a problem by varying a parameter of the problem. In our case we modify (6.11) to allow the influence of the MVS functional to be slowly phased out:

$$w\int \frac{d\kappa_n}{d\hat{e}_1}^2 + \frac{d\kappa_n}{d\hat{e}_2}^2 dA + \int P_{G^1} ds + \int P_{G^2} ds \qquad w \rightarrow 0. \qquad \textbf{(6.12)}$$

Our approach iteratively minimizes (6.12) reducing w by a factor of two before each iteration. Iteration continues until the desired continuity is achieved. Peters has shown that quintic polynomial patches are sufficient for forming a G^1 continuous surface out of a network of polynomial patches. The degree requirements for patches forming a G^2 continuous surface are as yet unknown. If, after reducing w to zero, we find that G^1 continuity has not been achieved, we may either, reapply the continuum method without any penalty for G^2 discontinuity, or subdivide the network of patches and reapply the method retaining the penalty for G^2 discontinuity.

6.2.6 G^2 Vertices

The order of the derivatives in the surface functional indicate a requirement for G^2 continuity. In addition to the G^2 penalty function, we may construct the network of patches to meet with G^2 continuity at their shared vertices only, and we maintain this continuity by construction during the minimization process. 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 5.1). An additional step is carried out after the principal 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 $\vec{S}_{uv}$ 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 (6.3) and (6.4). The twist vector must be adjusted to satisfy $f = \hat{n} \cdot \vec{S}_{uv}$. This is accomplished by forcing the tip of $\vec{S}_{uv}$ to lie in the plane perpendicular to $\hat{n}$, offset by distance f from the vertex

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

f can be computed from the values of κ_1, κ_2 and from the first and second order derivatives of $\vec{S}(u, v)$.

6.2.7 Symmetry—a time saving constraint

Many of the objects that we have designed using MVS possess some degree of symmetry. For the purposes of our discussion we will examine a surface fit to a tetrahedral frame. The surface was constructed by first fitting four-sided prisms to the edges of a tetrahedron. Then, using the topology of the prismatic frame, a smooth surface was computed that interpolates the resulting 20 vertices with 24 patches (Fig. 6.6).

On first examination, this surface appears to be quite complex, but on closer inspection we discover that there are actually only two distinct patches, the “inner” and “outer” patches (Fig. 6.7). This is because this object has full tetrahedral symmetry. If the symmetry is specified before the surface is calculated and enforced as an additional constraint, great computational savings may be reaped. In this case we may compute the shape of the object at a cost comparable to computing the shape of an object composed of only two patches. This translates into a twelve-fold savings in computation. If we look closer, we find that

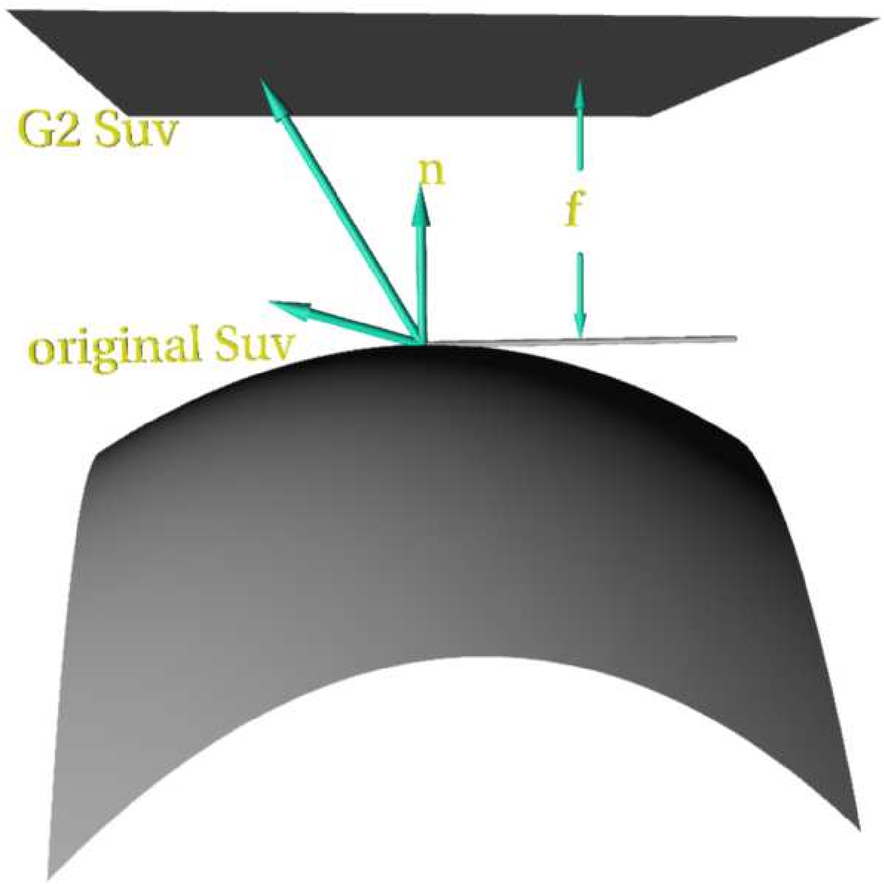


Figure 6.5. Construction of a G^2 Vertex

even these two patches have mirror symmetries and further efficiency may be gained by constraining these patches to be self-symmetric. In all, by specifying object symmetries as an optimization constraint, we see a 24-fold decrease in the time to compute this complex shape. Note that the arrangement of patches is an artifact of how the surface was constructed and that it is also possible to use 24 copies of a single patch.

To assess the benefit from imposing symmetry, we examine a simpler case. Here we compute the MVS interpolating the vertices of a cube; the surface is composed of six quintic tensor product patches, each with 48 degrees of freedom. For comparison we ran four optimizations each with successively greater exploitation of symmetry. We ran cases with no symmetry, 2, 6, and 24-fold symmetry corresponding to 288, 144, 48, and 12 degrees of freedom. Clearly we expect the case with the fewest degrees of freedom to complete more quickly, but we also notice that the improvement per *iteration* of the

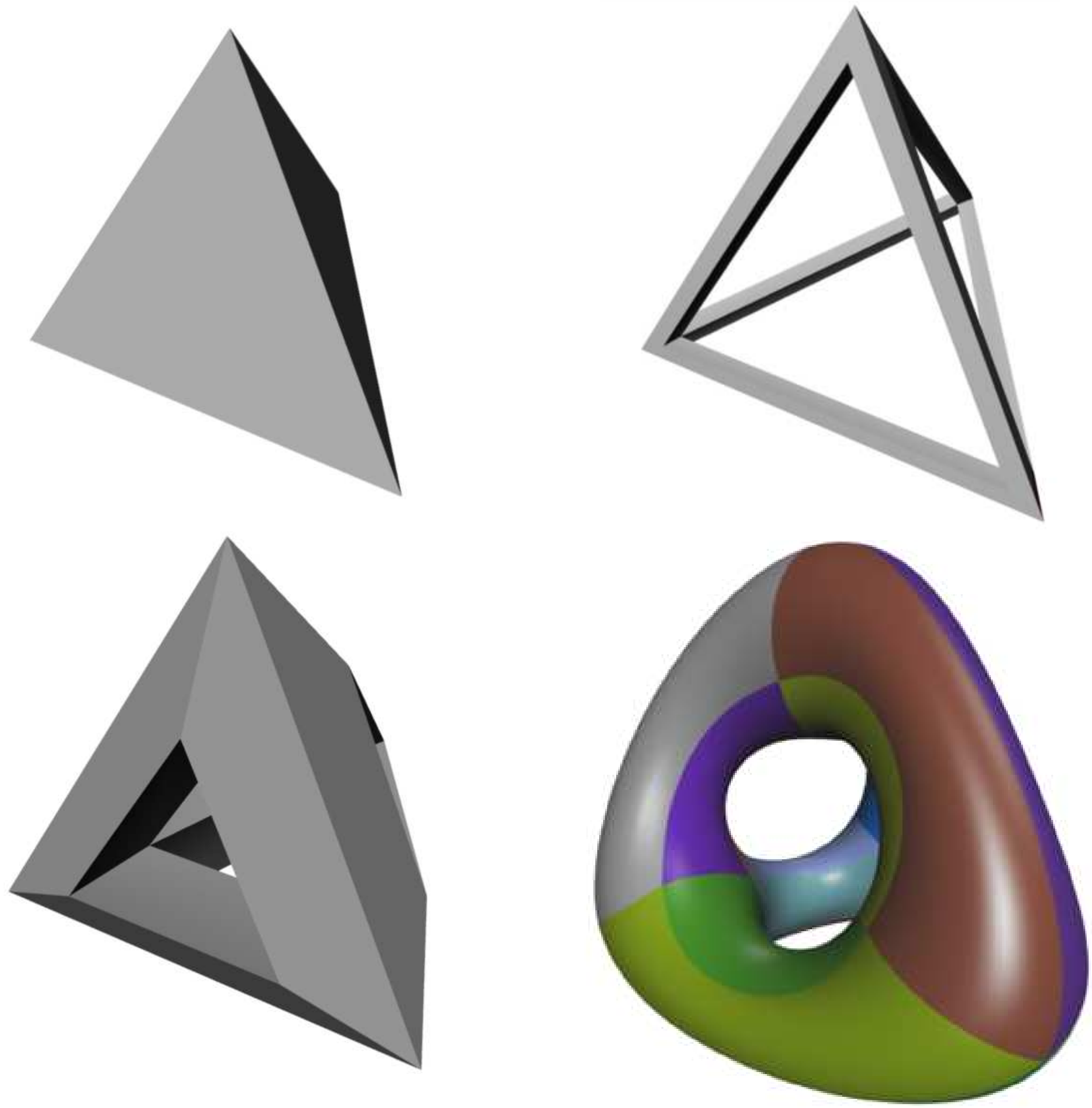


Figure 6.6. The Construction of a tetraThing

optimization is greater for those cases with fewer degrees of freedom. We see these results in Figure 6.8. The increase is due to the fact that there are fewer patches to minimize, but also that we have greatly reduced the dimensionality of the space in which the optimization is taking place.

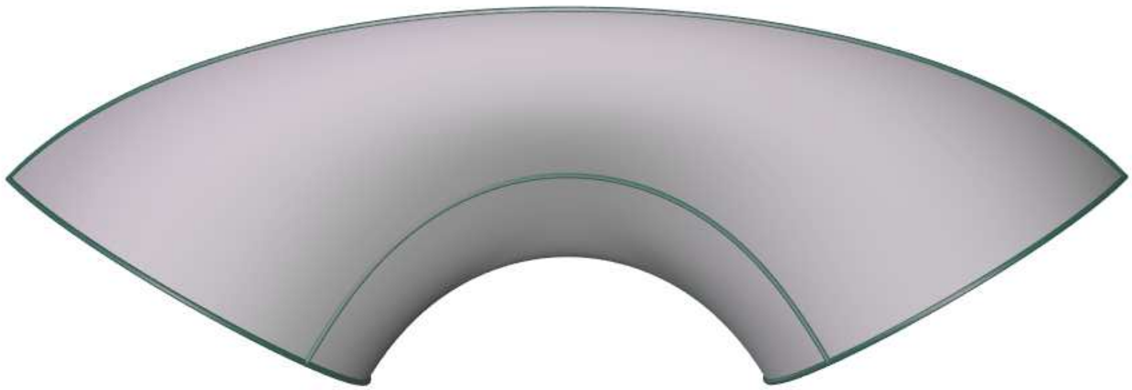


Figure 6.7. Unique Patches Composing tetraThing.
The tetraThing is composed multiple copies of these two patches. This particular patch-wise breakdown of the model is an artifact of the construction used to create it.

We take advantage of symmetry in an indirect fashion. Rather then precisely defining the symmetry relationship between a pair of patches, we define the transformation that maps one patch (the master) onto another (the instance); during optimization the control points of the instance are computed as transformed versions of the master's control points. Rotational symmetry is specified by defining an axis of rotation and an angle of rotation (Fig. 6.9). Mirror symmetry involves the specification of a plane through which the master is reflected to produce the instance. It is also possible to specify arbitrary transformations relating one patch to another. Currently we restrict intrapatch symmetries to a subset of the possible symmetries (Fig. 6.10). Triangular patches may be defined with three-fold rotational symmetry. Quadrilateral patches may have two and four-fold mirror (D_1,D_2) or rotational symmetry (C_2,C_4).

6.3 Initialization

The gradient descent scheme described in section 6.1 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 methods 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. We use an MVN to initialize the control points on the

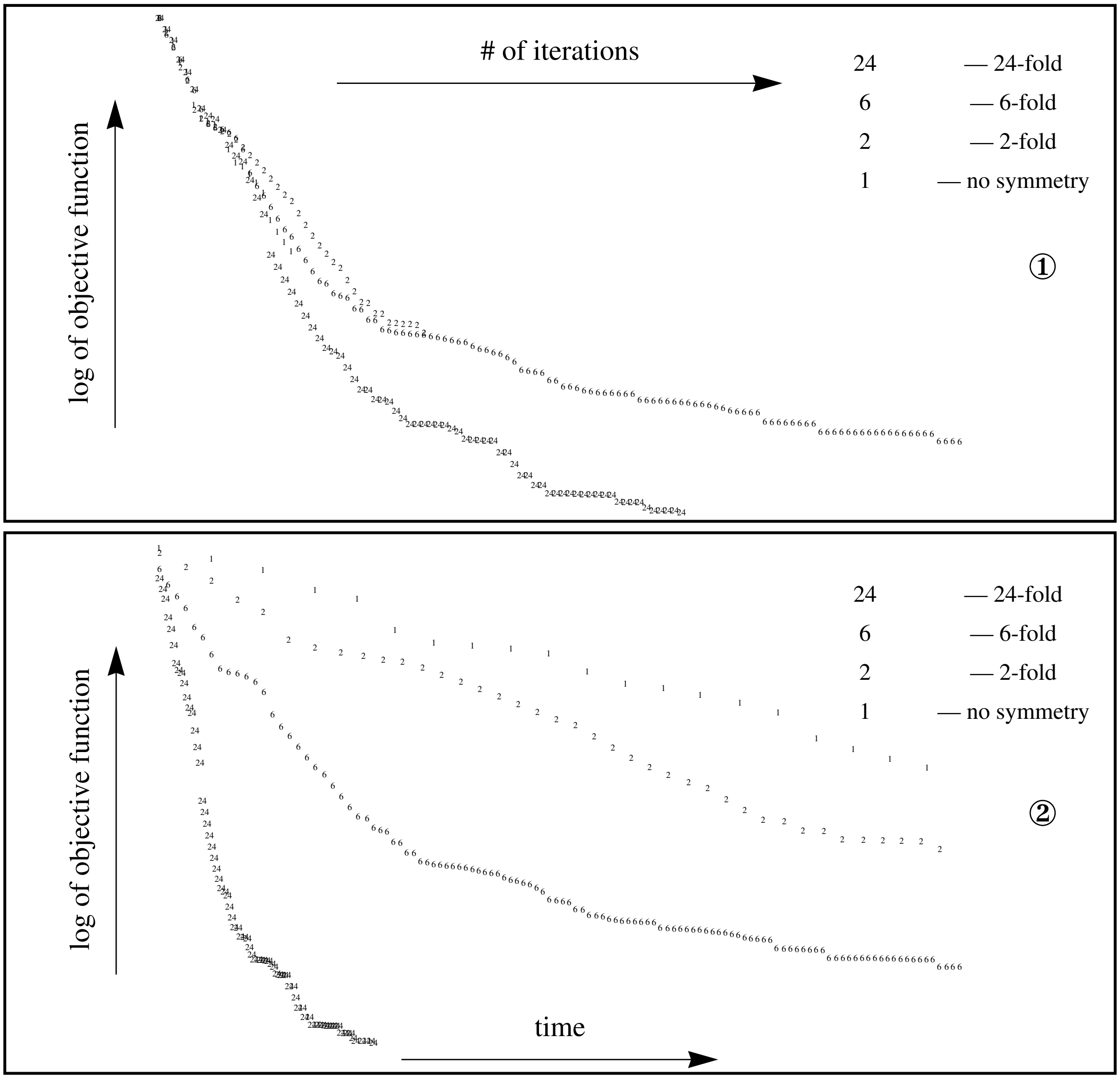


Figure 6.8. Symmetry—Convergence vs. Iterations/Time

① Initially, the various levels of symmetry have similar convergence per iteration, but once the gross shaping of the surface is complete, the 24-fold symmetric optimization converges more efficiently. ② The greater the symmetry, the faster the convergence.

perimeter of each patch. We discuss three methods for the initialization of the patch's interior control points: Peters' construction [161], approximate G^1 continuity, and zero corner derivatives.

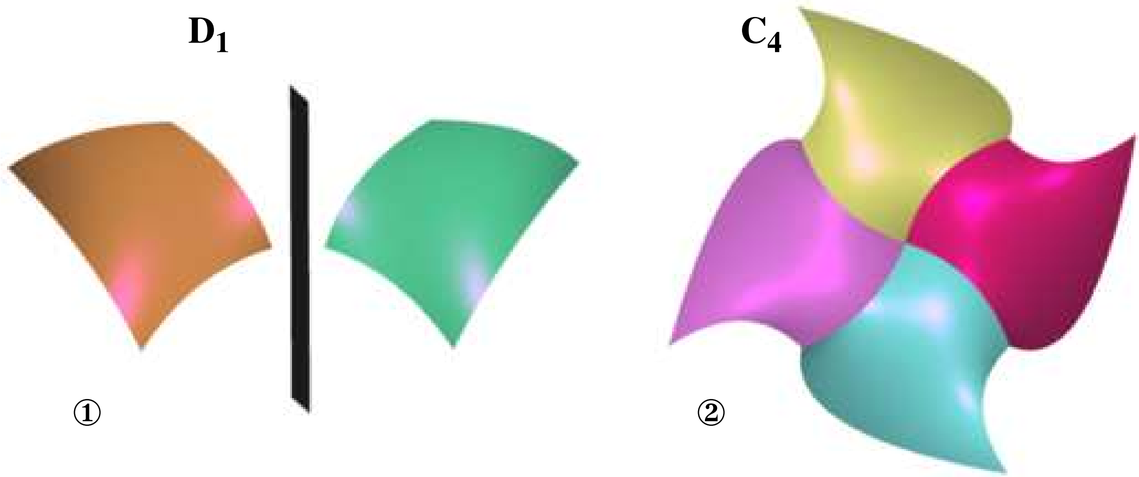


Figure 6.9. Examples of Interpatch Symmetries
① mirror symmetry (D_1), ② 4-fold rotational symmetry (C_4)

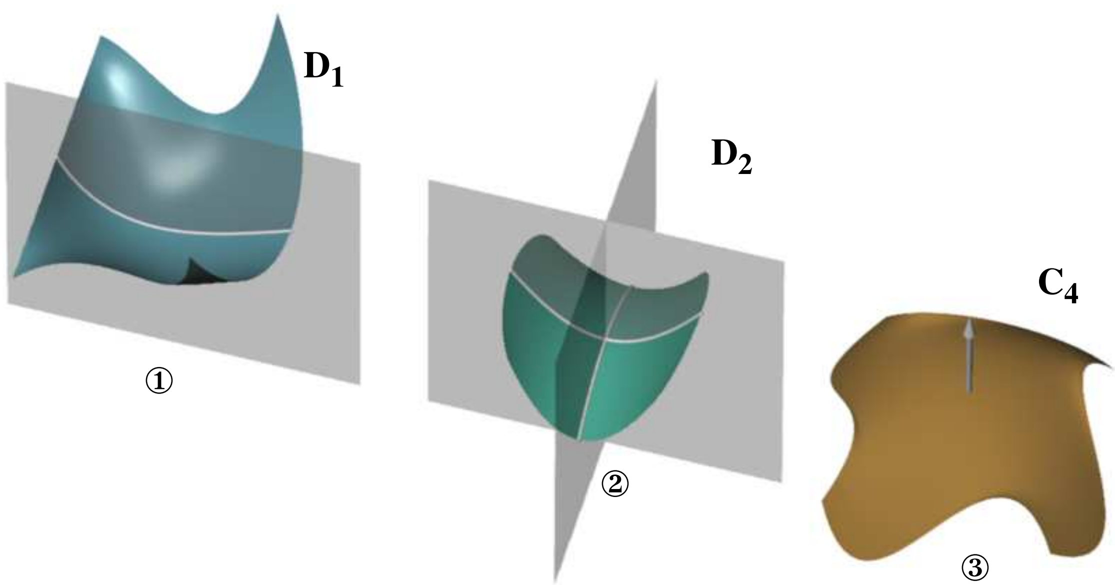


Figure 6.10. Examples of Intrapatch Symmetries
① two fold mirror symmetry (D_1), ② four fold mirror symmetry (D_2), ③ four fold rotational symmetry (C_4).

Peters' construction finds an exact solution to the G^1 continuity problem, using a

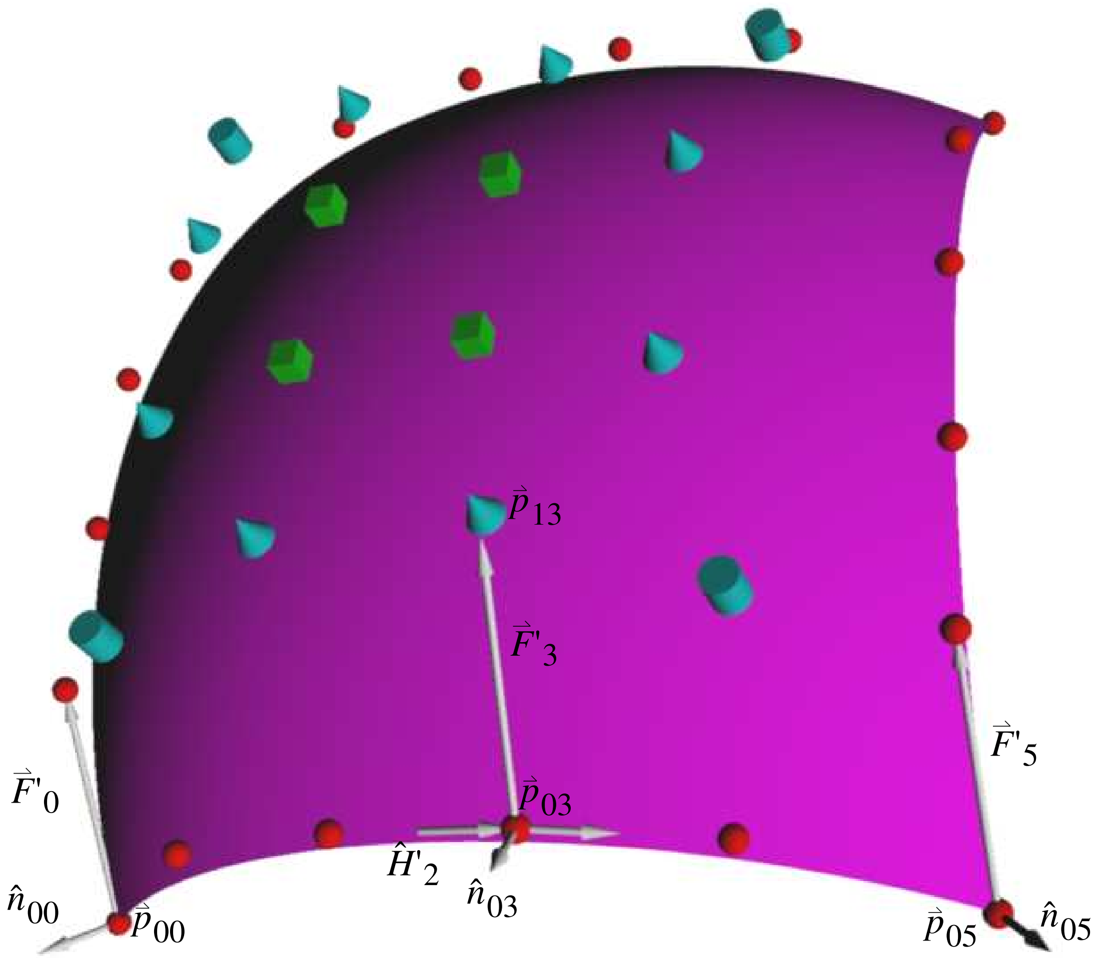


Figure 6.11. The Control Points of a Bézier Patch.

The control points are grouped as the 20 perimeter control points (●), the 12 adjacent control points (▲, ■), and the 4 central control points (■). As an example of initialization, $\vec{p}_{13}$ is computed by linearly interpolating the surface normal vectors at the corners and the magnitudes of the corresponding difference vectors.

sufficient construction and heuristics to set those degrees of freedom not assigned by the construction. Experience has shown that starting with a G^1 continuous surface has no particular advantage, in fact the heuristics used to compute such a surface often result in a starting point further from the optimal surface.

To achieve approximate G^1 continuity, we position the interior control points so that: 1) an approximate boundary normal is interpolated and 2) the fourth order derivatives at the patch corners are 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. 6.11). The heuristic used to position the control points adjacent to the perimeter linearly interpolates the normal vectors and the magnitudes of difference vectors. Figure 6.11 and eq. (6.13) illustrate the approach, with the calculation of $\vec{p}_{13}$:

$$\vec{p}_{13} = \vec{p}_{03} + \vec{F}'_3 \qquad \vec{F}'_3 = \left\|\vec{F}'_3\right\|\hat{F}'_3 \qquad \hat{F}_3 = \frac{\hat{n}_{03} \times \hat{H}'_2 + \hat{n}_{03} \times \hat{H}'_3}{\left\|\hat{n}_{03} \times \hat{H}'_2 + \hat{n}_{03} \times \hat{H}'_3\right\|}$$

$$\left\|\vec{F}'_3\right\| = \frac{2}{5}\left\|\vec{F}'_0\right\| + \frac{3}{5}\left\|\vec{F}'_5\right\| \qquad \hat{n}_{03} = \frac{\frac{2}{5}\hat{n}_{00} + \frac{3}{5}\hat{n}_{05}}{\left\|\frac{2}{5}\hat{n}_{00} + \frac{3}{5}\hat{n}_{05}\right\|}. \tag{6.13}$$

This approach fails when modeling highly curved surfaces; it produces an usable starting shape. For example if the normal vectors differ by more than 180°, linearly interpolating them will yield nonsensical results.

Finally, we may also position all the interior control points so that the twist vector, and third and fourth order derivatives are zero at the corners of the patch:

$$\vec{S}_{uv}(u, v) = 0 \qquad \vec{S}_{uuv}(u, v) = 0$$
$$\vec{S}_{uvv}(u, v) = 0 \qquad \vec{S}_{uuvv}(u, v) = 0$$
$$u, v = \{0, 1\}$$

We have found this final technique to be simple and robust for a large variety of problems. There are cases for which even this approach fails; for example the surface in Figure 6.26 had to be manually initialized in order to start the optimization in the correct energy valley.

6.4 Surface Analysis Methods

To assess the quality of curved surfaces and of different design techniques, it is necessary to develop rendering and analysis methods that expose the subtler aspects of curved surface character. Forrest [73] discusses a variety of techniques for increasing the information content and the understandability of renderings of curved surfaces. He discusses realistic renderings, hedgehog surfaces, contouring, isoparametric lines, pseudo-coloring by Gaussian and mean curvature, and various combinations of these techniques. Klass [117] evaluates and fairs (modifies) curved surfaces with *lines of reflection*, i.e. the images of parallel lines reflected in a smoothly curved surface. Dill [51] discusses the assessment of curved surfaces based on pseudo-coloring that represents mean and Gaussian curvature. He presents various schemes designed to expose different aspects of surface character. To assess the results of his fairing scheme, Kjellander [116] uses isoparametric lines, curvature profiles of surface sections, and renderings of lines of principal curvature. Schweitzer [194] presents a technique based on synthetic textures designed to enhance the understandability of curved surface renderings. He renders patches with dots scattered across the surface in roughly even patterns. These renderings are designed to take advantage of three aspects of texture that affect our perception of shape: size, shape, and density. These correspond to three types of gradient. Size is a function of depth. Shape is a function of normal to the surface. Density is a function of both compression and convergence. Compression is due to perspective, and convergence is due to slant. Hoschek [103] describes the use of *polarity* to expose inflections in curves, and areas of zero Gaussian curvature in surfaces. The polar complement of a point relative to a circle is a line perpendicular to the line connecting the point and circle center. The line is positioned at a distance inversely proportional to the distance between the point and circle center. As a curve or surface is traversed, an envelope of lines or planes is formed: the polar curve or polar surface. Both polar forms exhibit cusps corresponding to points of inflection and zero Gaussian curvature. Poeschl [165] discusses surface analysis techniques using isophotes, contours of $\hat{l} \cdot \hat{n}$. He also illustrates how shadow outlines, or *terminators*, can expose G^1 and G^2 discontinuities; note that terminators *are* isophotes. This may be seen in a cylinder topped with a cone and a hemisphere (Fig. 6.12). Beck et al. [10] have compared analysis techniques including shaded renderings, contours of

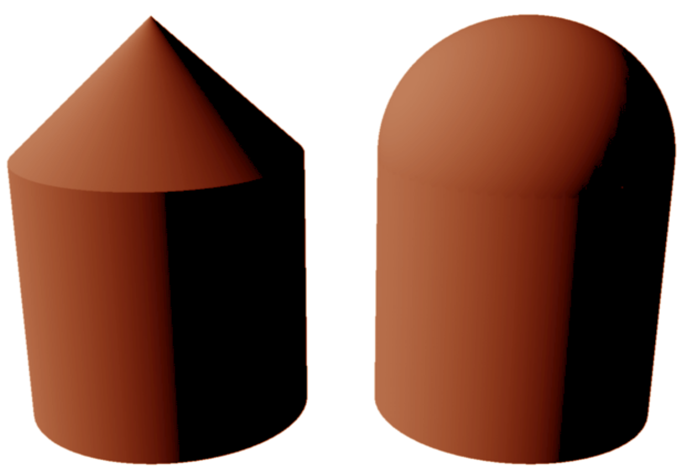


Figure 6.12. Terminators Exposing Discontinuities.
G^1 and G^2 surface discontinuities cause respectively G^0 and G^1 discontinuities in isophotes and terminators.

intensity, curvature, etc., lines of principal curvature, geodesic paths, and curvature pseudo-coloring. Munchmeyer [137] presents a case study of surface analysis techniques concerning the display of contours of Gaussian curvature, mean curvature, maximum principal curvature, principal directions, asymptotic directions, and level contours. The primary problem with these methods is that while some give false negative indications, others miss surface irregularities. In [170] Pottmann provides simple examples where lines of reflection fail to expose discontinuities of curvature. He describes a new approach that is designed to maximize the apparent curvature discontinuity at patch-patch boundaries. His method relies on results from differential geometry; for any direction of travel on a surface, there is a conjugate direction determined by the curvature of the surface. As the curvature of the surface is varied the conjugate direction changes. Pottmann finds the surface tangent direction at a point on the patch-patch boundary that maximizes the angular difference between the conjugate directions corresponding to the two abutting patches; a difference in conjugate direction will only result if the patches have differing curvatures. The result is displayed as short vectors originating at the patch-patch boundary and oriented in the derived conjugate directions. This approach illustrates the maximum visual discontinuity a viewer could experience. However, this approach fails to detect curvature discontinuities in some cases; this is because on a parabolic surface one of a pair of conjugate directions is always in the direction of zero curvature. For example, Pottmann's approach fails to detect a discontinuity when half cylinders of differing radii meet along their lines of zero curvature (Fig. 6.13).

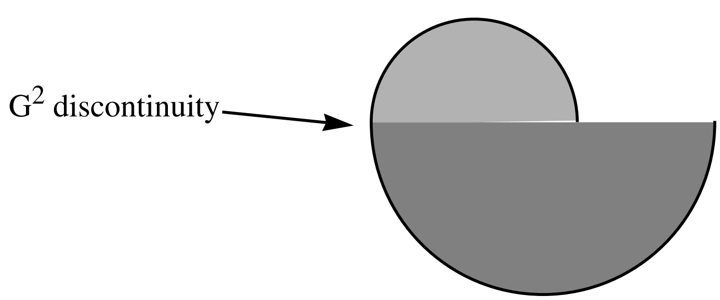


Figure 6.13. Cross Section of Half Cylinders of Differing Radii

Pottman's G^2 discontinuity detection scheme fails to indicate a discontinuity for this joint because, for any cross boundary tangent the conjugate direction is along the boundary.

We have reviewed work on rendering techniques designed to aid in the evaluation of surface quality. Here we describe an additional technique based on rendering the surface resulting from a functionally controlled normal offset from the original surface:

$$\vec{S}'(u, v) = \vec{S}(u, v) + f(u, v)\hat{n}_{\vec{S}}(u, v) \qquad \textbf{(6.14)}$$

where $f(u, v)$ is a scalar function such as mean or Gaussian curvature. Positions on the surface $\vec{S}'(u, v)$ are straightforward to compute using equation (6.14). The normal $\hat{n}_{\vec{S}'}$ to $\vec{S}'(u, v)$ may be computed by finding the partial derivatives of $\vec{S}'(u, v)$ with respect to u and v and taking their cross product. This is straightforward since the partial derivatives of $\hat{n}(u, v)$ were computed as part of the optimization process. Functional offset surfaces provide an excellent means for analyzing and comparing the curvature characteristics of surfaces (Fig. 6.15⑤). Discontinuities of curvature appear as gaps in the surface, and the distribution of curvature is conveyed by the smoothness of the offset surface.

Figures 6.14 and 6.15 illustrate some of the techniques we used in the analysis of our results. We classify techniques involving surface position and orientation as *first order*, and those techniques involving surface curvature are classified as *second order*. From among the first order approaches we used primarily lines of reflection (Fig. 6.14⑤) to expose G^1 discontinuities, and environment mapping (Fig. 6.14⑥) in the subjective evaluation of surfaces. In combination with these techniques, real-time motion produces many additional visual cues highlighting discontinuities and enhancing a sense of shape quality. The second order approaches are specifically designed to help expose the character of a surface's curvature distribution. In Figure 6.15①, a method analogous to those used in the analysis of MVN and MVC space curves is shown. The method does not carry over very effectively to surfaces, other techniques are more useful. In ② of the figure, the shaded surface is overlaid with contours of constant Gaussian curvature. We have experimented with contouring other quantities, such as mean curvature, and minimum and maximum curvature, and found the contours of Gaussian curvature most useful. Such contours make G^2 discontinuities obvious, there are breaks in the contours. However, they do not provide a sense of the magnitude of curvature or of apparent discontinuities. Figure 6.15③ is a rendering of the surface pseudo-colored based on Gaussian curvature. Areas of positive curvature are pink to red, areas of negative curvature are light to deep blue, and black regions have approximately zero Gaussian curvature. This approach provides a sense of curvature sign and magnitude, but discontinuities are difficult to identify. In Figure 6.15④ the surface is pseudo-colored to provide a stronger sense of curvature distribution coloring with hues that vary quickly with curvature. ⑥ combines the pseudo-coloring of ④ with the functional offset surface shown in Figure 6.15⑤.

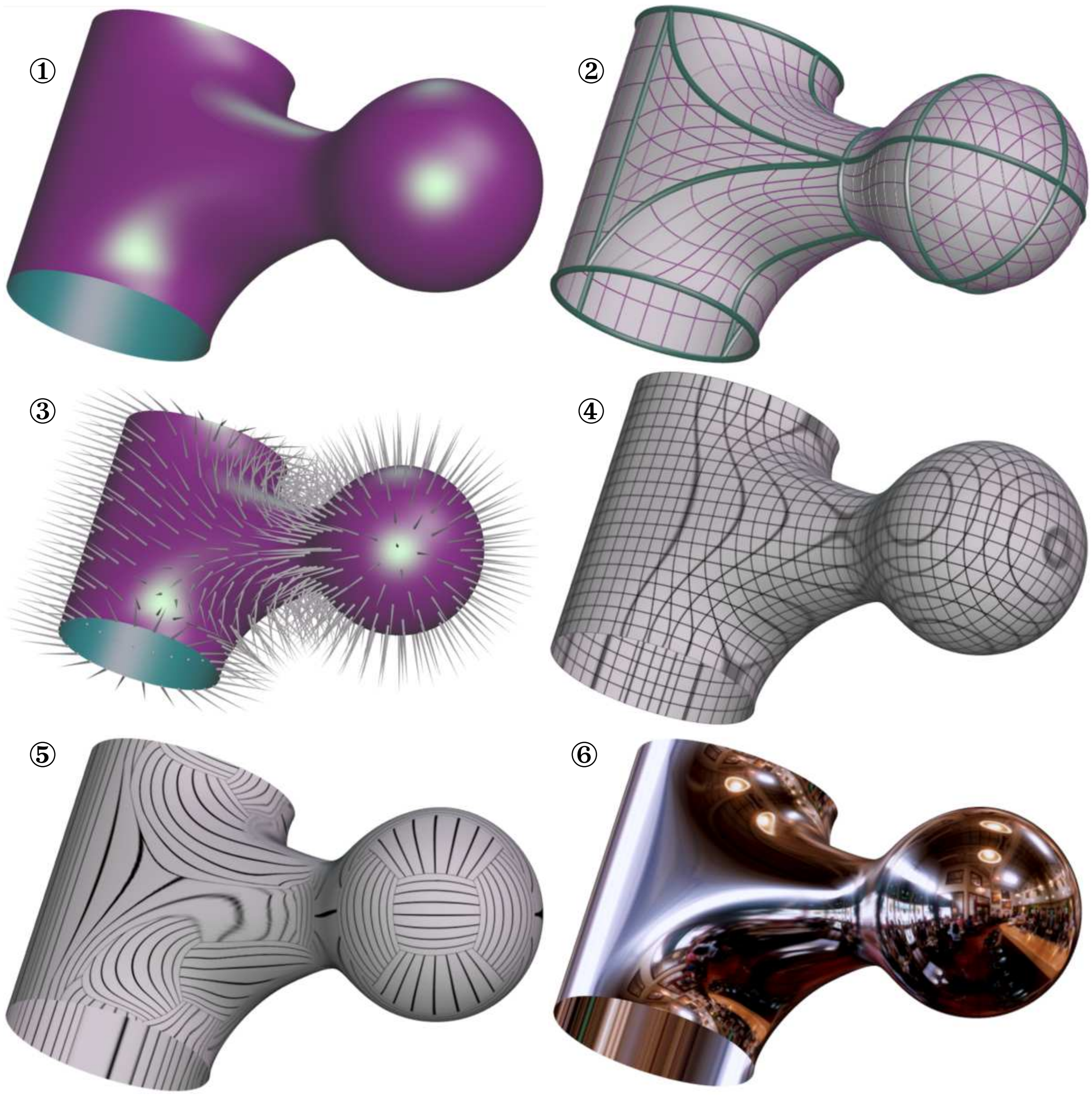


Figure 6.14. Surface Analysis—First Order.

① Simple shaded rendering with specular highlights. ② Patch boundaries with isoparametric lines indicating parameterization. ③ A "hedgehog" surface; spines indicate surface orientation and tangent discontinuities. ④ Surface contours. ⑤ Lines of reflection; G^0 reflection line discontinuity $\rightarrow$ G^1 surface discontinuity, G^1 line discontinuity $\rightarrow$ G^2 discontinuity. ⑥ Environment mapping, similar to lines of reflection, but provides for a subjective evaluation of aesthetics.

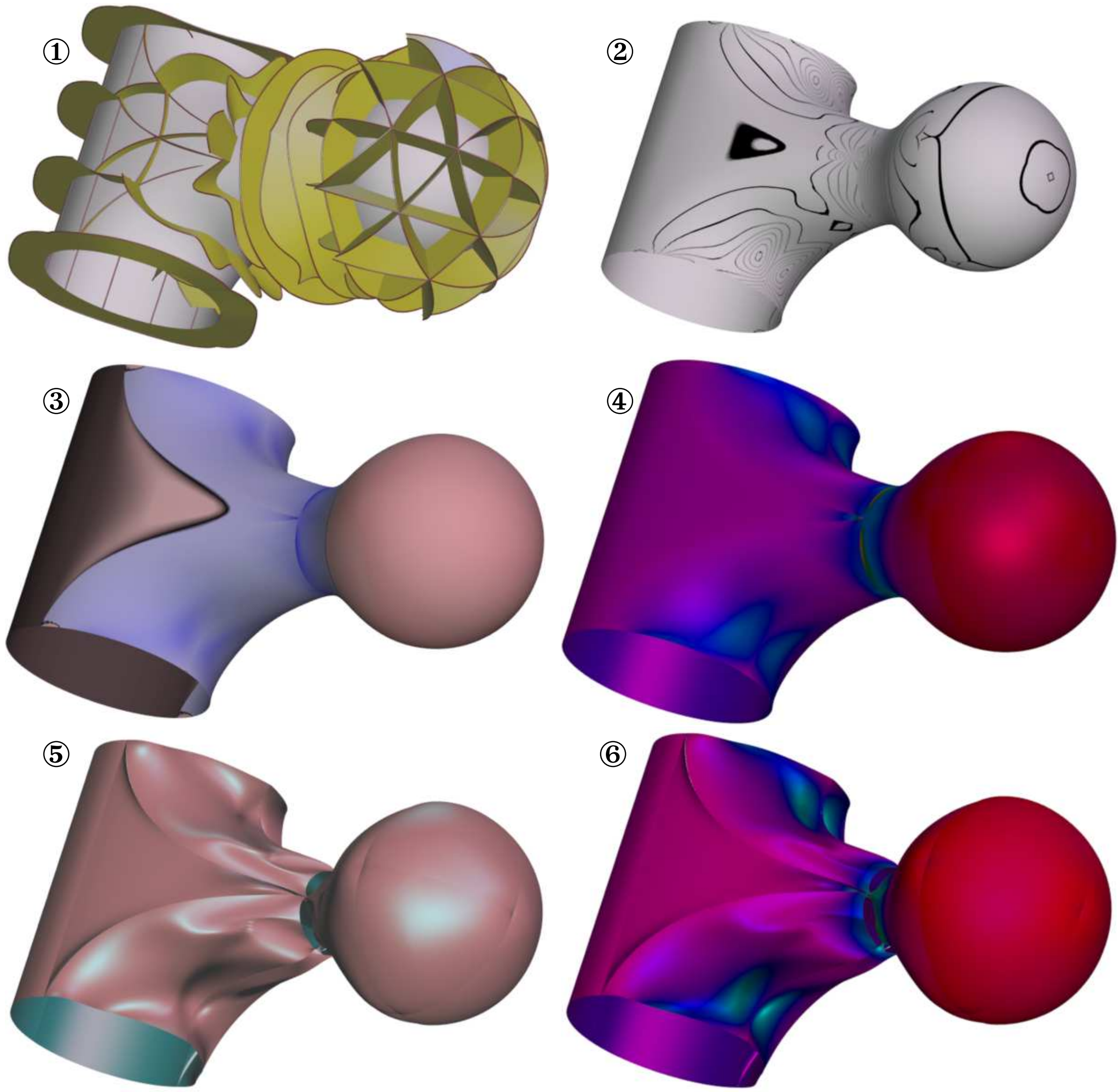


Figure 6.15. Surface Analysis—Second Order.

① isoparametric normal curvature profiles. ② Contours of Gaussian curvature, $(\kappa_1 \kappa_2)$. ③, ④ Gaussian curvature pseudo-coloring. ③ Pink → red ≡ ε → positive, lt. blue →blue ≡ -ε→ negative, black ≡ zero. ④ The full range of hues is mapped to the full range of Gaussian curvatures. ⑤ A functional offset surface. This surface is offset from the original in proportion to Gaussian curvature. The surface has expanded in areas of positive (elliptic) curvature, and contracted in areas of negative (hyperbolic) curvature. ⑥ A pseudo-colored offset surface; ④ and ⑤ are combined to produce this rendering of curvature distribution.

6.5 Examples & a Comparison of Functionals

In order to evaluate the quality and usefulness of MVS, we examine a few interpolation and design problems. The special rendering techniques described in section 6.4 are used to assist in the evaluation of the quality of these surfaces. First we examine the problem of interpolating the corners of a cube. Next, we revisit the data sets from the University of Washington, introduced during the evaluation of MVN. The remainder of the section is devoted to the modeling of more complex shapes, and an assessment of the effect of flexible MVN on the quality of MVS.

6.5.1 Spheres

In Figure 6.16 we use functional offset surfaces and pseudo-coloring to compare the MVS functional with four other functionals. In ② and ③ only the G^1 and G^2 penalty functions were minimized when fitting a surface to the points of a cube.④ illustrates the result of using a linearized approximation to strain energy; curvature distribution is improved. Next, ⑤ true strain energy is minimized producing a surface with fairly uniform curvature. Finally, in Fig. 6.16⑥ an MVS surface fitted to the corners of a cube produces a very close approximation to a sphere.

6.5.2 University of Washington Data Sets

In section 5.4 we compared MVN with the networks produced by the techniques studied by the Mann et al. [126]. In this section we revisit these data sets continuing the comparison with MVS. Both of the methods we make comparisons with use a Clough-Tocher scheme (Fig. 3.22), placing three quartic triangular patches in each network opening. We place a single quintic triangular patch per network opening in the construction of an MVS. In all cases the MVS interpolants are clearly superior to those of both Shirman-Séquin and DBHS, (Fig. 6.17-6.21). In each figure, the surface computed using each technique is displayed as a row of three renderings. The first image in each row is a simple shaded rendering. The second rendering is pseudo-colored according to Gaussian curvature. The third surface in each row is a shaded rendering of the functional offset surface, where offset is proportional to Gaussian curvature. The MVS we show here are limited by the very large number of degrees of freedom in the more complex examples (Table 5.1). It is unrealistic to calculate MVS for such problems; in the cases with large numbers of degrees of freedom (>500) it took several days to compute the solutions shown.

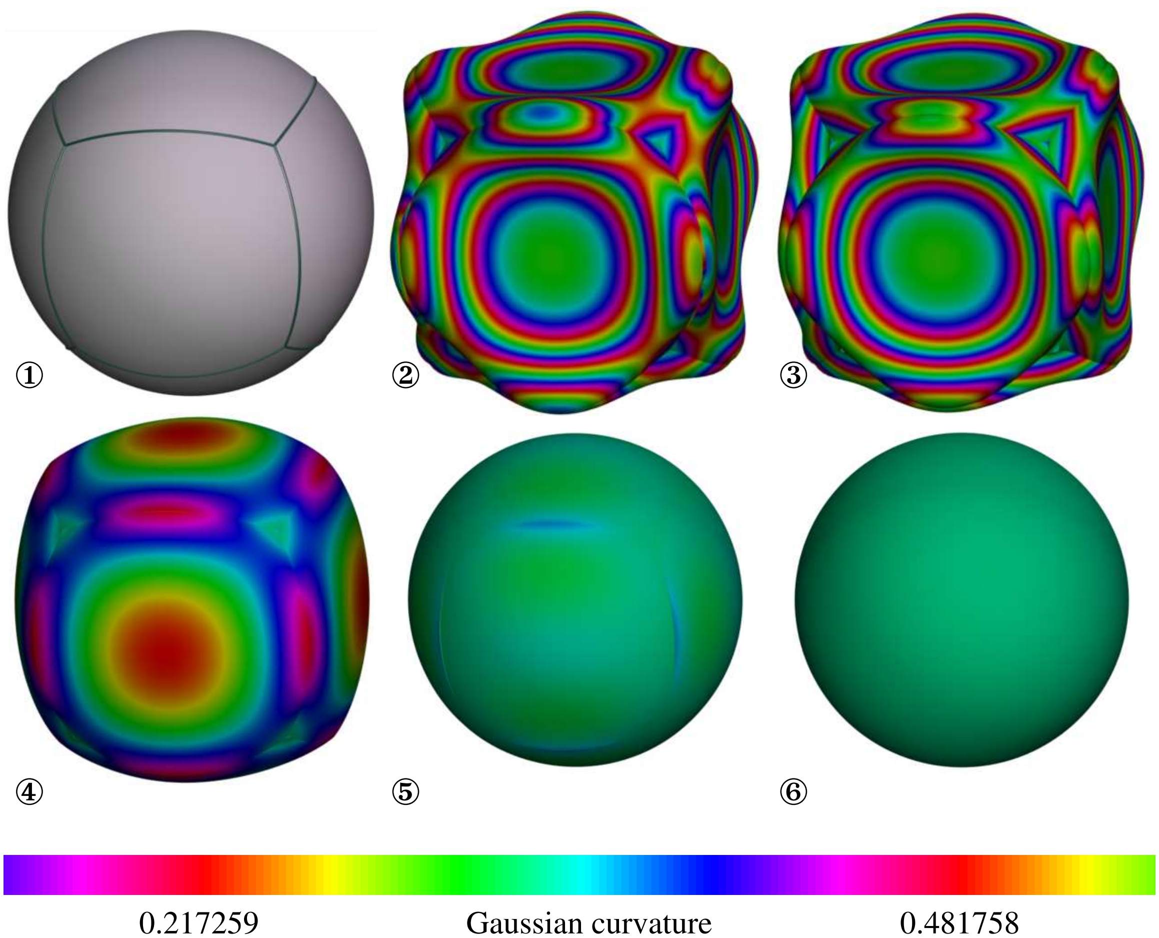


Figure 6.16. Surfaces Interpolating The 8 Corners of a Cube.

① The MVN and MVS interpolating the cube. Pseudo-coloring combined with functional offset surfaces illustrate the differences among the different objective functions. ②-G^1 penalty alone, ③-G^2 penalty alone. ④-linearized strain energy, ⑤-strain energy, ⑥-MVS.

6.5.3 Three Handles

Figure 6.22 illustrates the application of MVS to a more complicated example. In ① we provide the MVN interpolated to create the G^1 MVS. ② illustrates the surface rendered with lines of reflection. In Figure 6.22, strain energy ③,④ and the MVS functional ⑤,⑥ are compared. The differences are subtle, curvature varies more smoothly and is distributed more evenly over the MVS surface.

Data Set	# of vertices	# of curves	# of patches	# of degrees of freedom
Octahedron	6	12	8	144
Sphere6	14	36	24	432
Franke4	36	85	50	900
Torus	50	150	100	1800
Capsule	74	216	144	2592

Table 5.1. University of Washington Data Set Stats.

N.b. the surfaces computed the using the Shirman-Séquin and DBHS methods are made up of quartic triangular patches, three quartic patches for every quintic patch used in an MVS. Thus there are three times the *number* of patches used by these methods.

6.5.4 Flexible MVN

In Fig. 6.23 and Fig. 6.24 we examine the consequences of computing an MVS with a flexible MVN framework. Also included here is the same basic surface with all but two point constraints removed. The combination of these two points and symmetry constraints produces a stable solution. Generally, we find that the added degrees of freedom of a flexible network allow for substantial improvement in the curvature distribution of the surface. In Fig. 6.23①,②&③ we see the shaded renderings of the surfaces, the MVS with flexible MVN② appears rounder and more full. The MVS with the largest number of degrees of freedom③ is the fairest. We have rendered these surfaces with lines of reflection in Fig. 6.23④-⑥; the more “flexible” the MVS the more regular the lines of reflection. Lastly, in Fig. 6.24, we directly examine the distribution of curvature across the surfaces

6.5.5 Minimum Topological Shapes

The MVS is a new class of surface. Like minimum energy surfaces and minimal surfaces there are shapes that are characteristic of minimum variation surfaces. We have discussed the cyclides as one class of surface readily formed by MVS under compatible constraints. Here we consider, briefly, what shapes are formed in the absence of *any* constraints. For surfaces of genus zero and one, the cyclide nearest and downhill from the starting point is formed, but for surfaces of higher genus the answer is less clear. The problem arrises that, without any constraints, surfaces of genus greater than one expand toward infinity. This is because as the scale of an object is increased, its MVS functional is reduced in value; the value of the MVS functional is not scale invariant (see (6.1) on p. 124). The genus zero and one surfaces do not expand limitlessly because they are able to reach zero energy

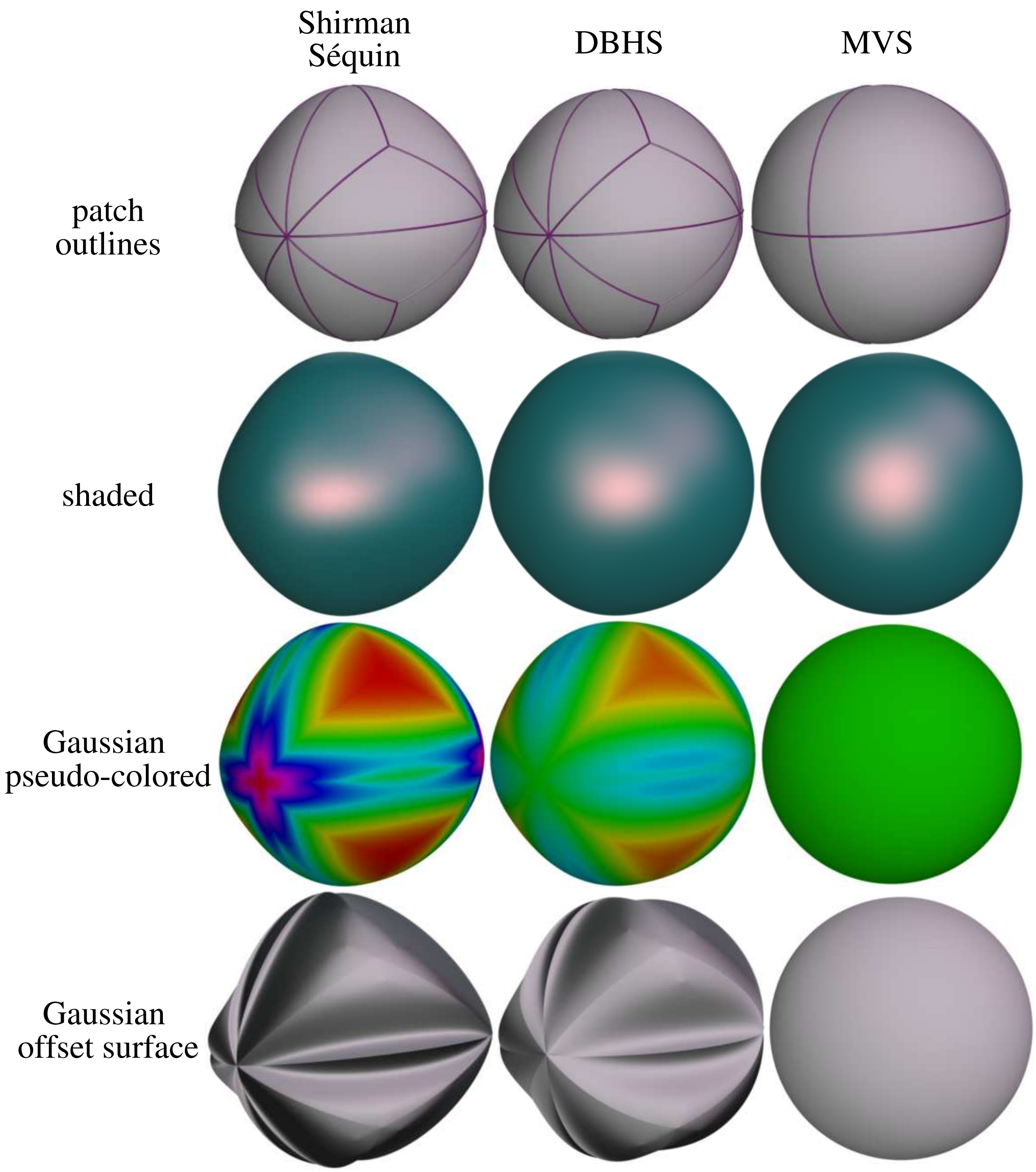


Figure 6.17. Octahedron.

Both Shirman-Séquin and DBHS produce surfaces with higher curvature at the boundaries of the original network.

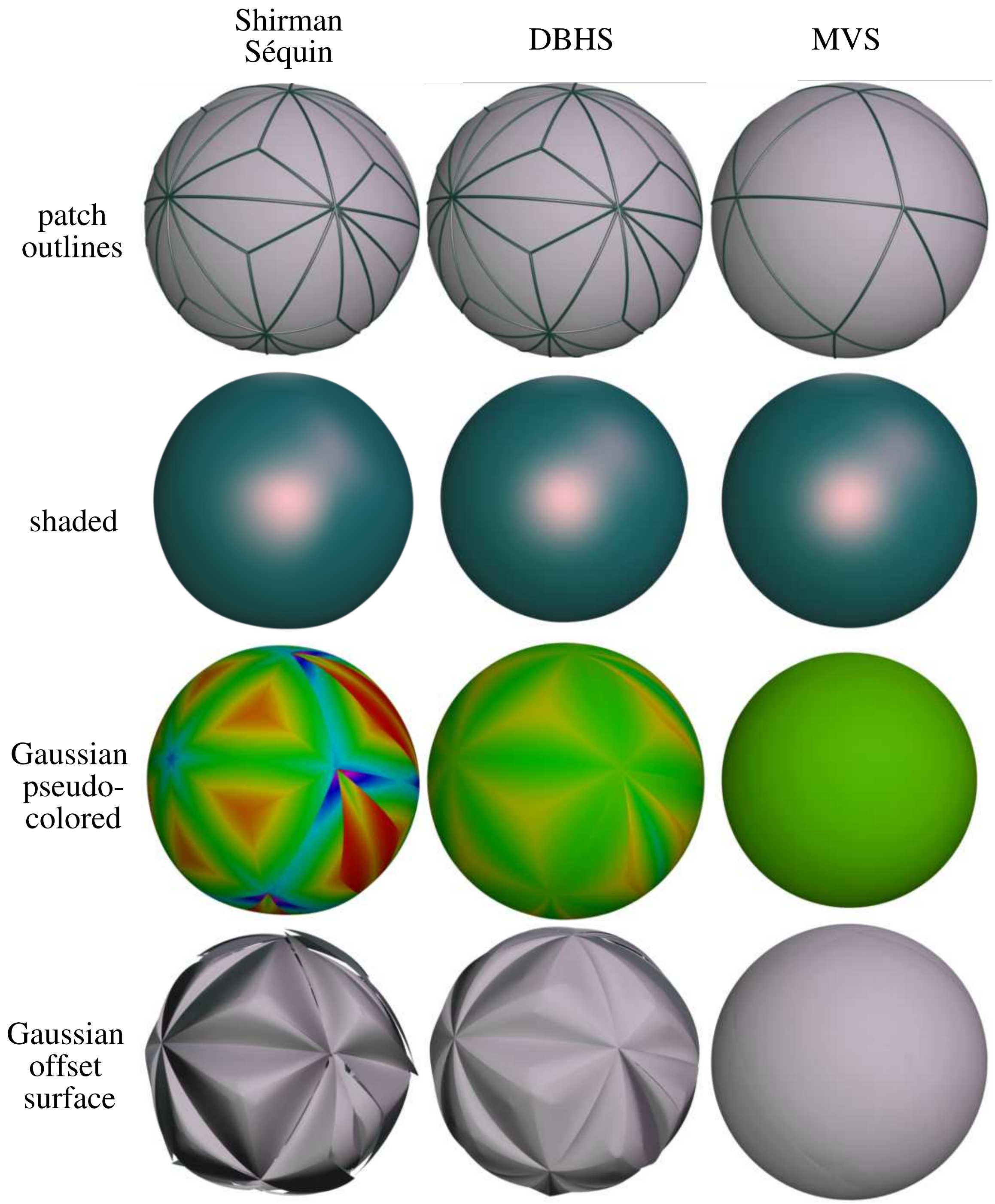


Figure 6.18. Sphere6

The irregular spacing of sample points on the sphere cause Shirman-Séquin to generate a surface with large curvature discontinuities.

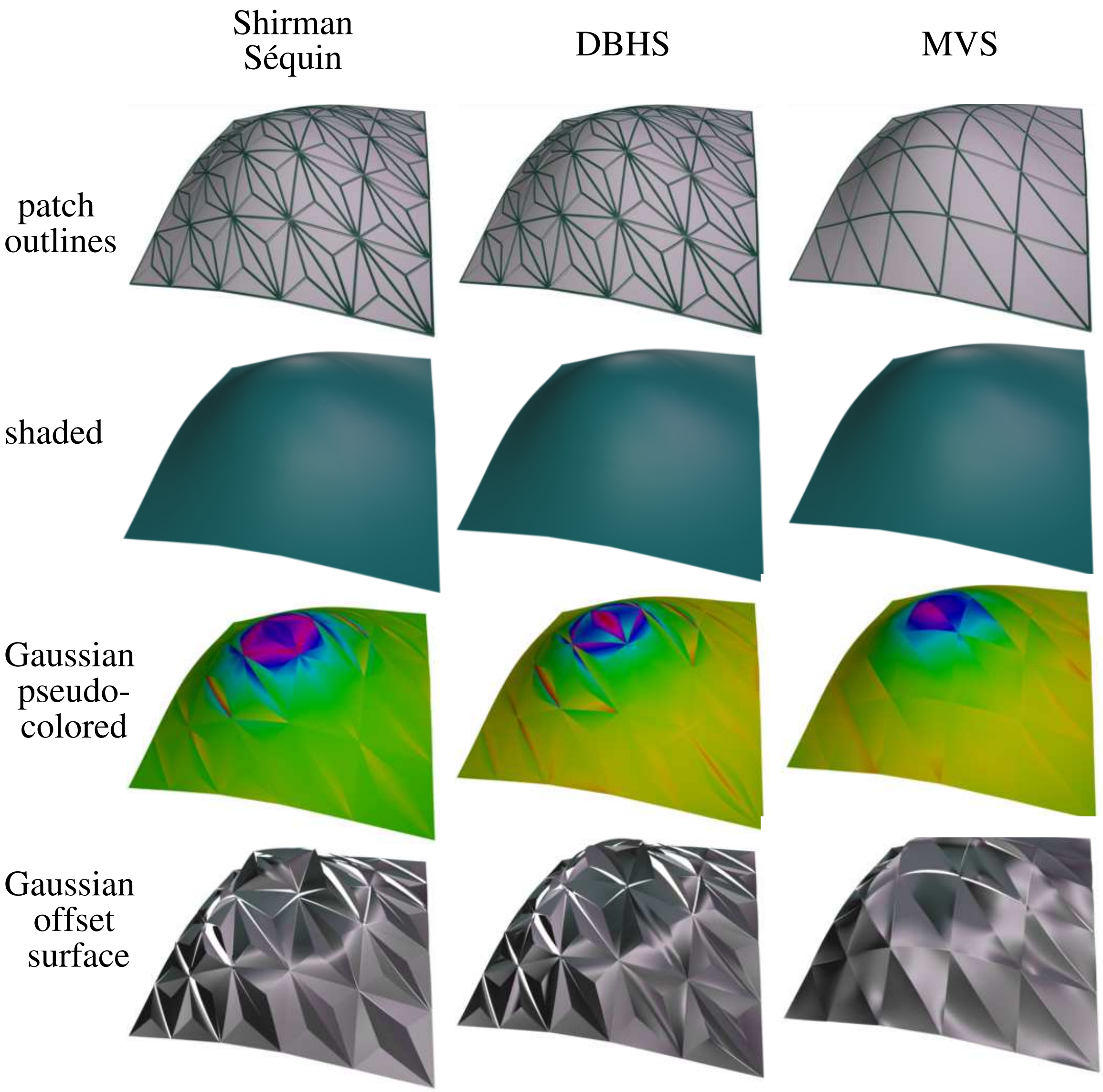


Figure 6.19. Franke4.

The quality of the MVSs computed for this and the remaining University of Washington data sets were limited by the large number of degrees of freedom. Surface computations were terminated before completion.

without expanding. We have, as an experiment, computed surfaces of genus greater than one with only enough constraints to prevent scaling. The TetraThing (Fig. 6.27) is an

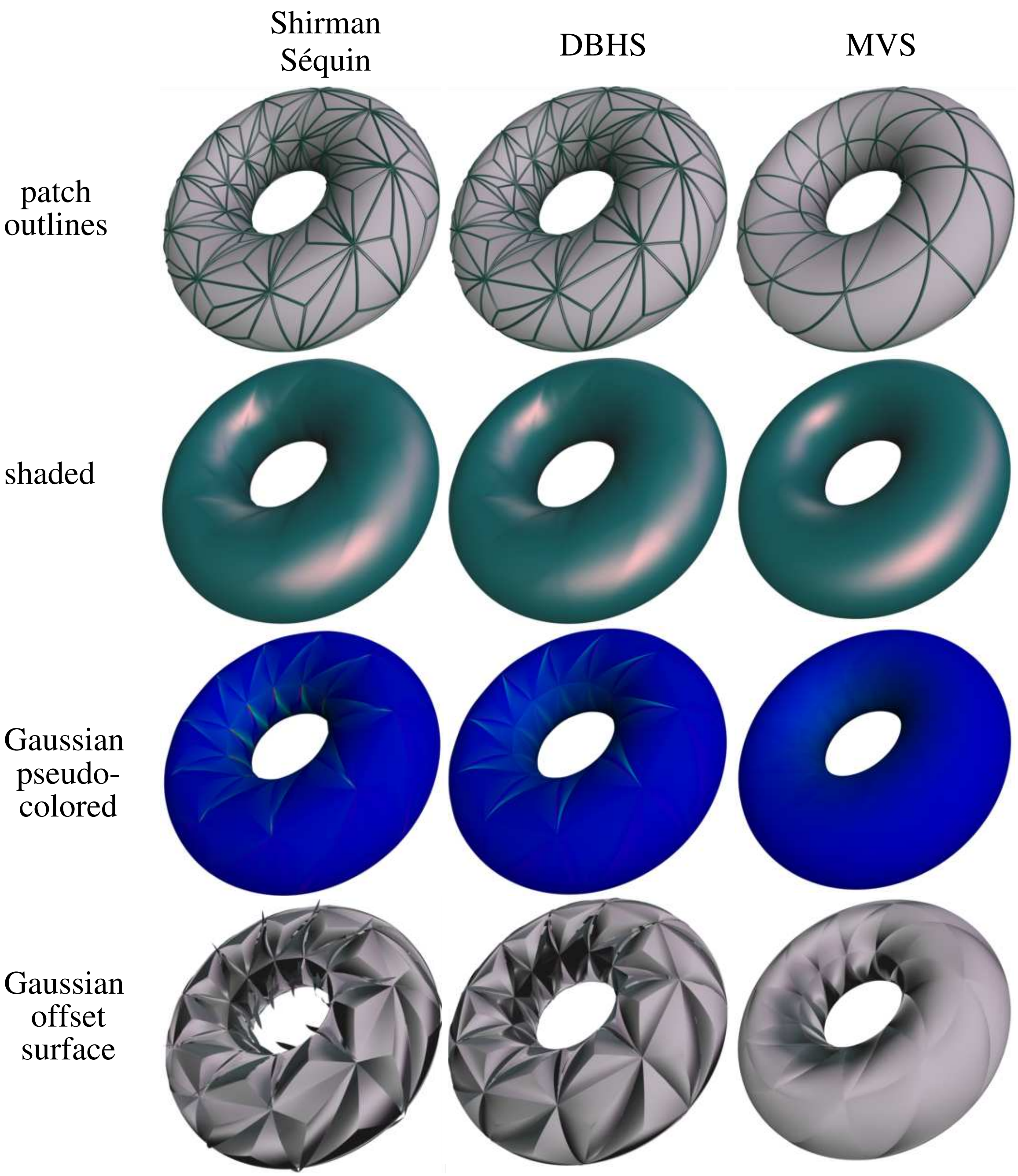


Figure 6.20. Torus.

Of the University of Washington data sets, the torus data set is the most technically challenging. Both Shirman-Séquin and DBHS produce surfaces with visible creases. Despite the large number of degrees of freedom, the MVS surface is smooth with smoothly varying curvature.

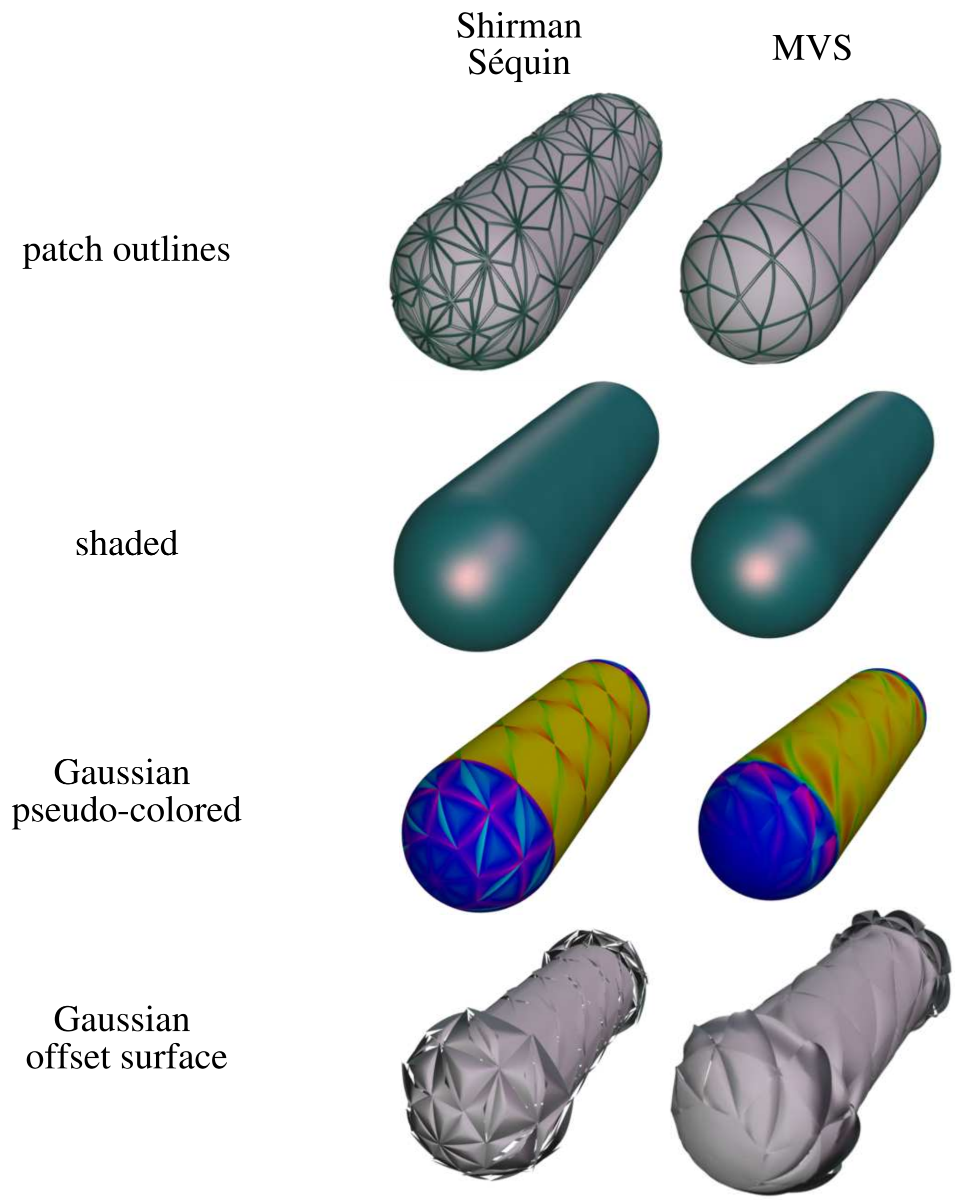


Figure 6.21. Capsule.

Because the MVS tends toward a G^2 continuous surface, there is evidence of ringing where the hemisphere and cylinder meet, a G^2 discontinuity in the data set. Allowing the MVS to contain a G^2 discontinuity along the cylinder-hemisphere boundary would correct this flaw. DBHS is not included because the method requires surface curvature information at all points, there is no consistent curvature information along the G^2 discontinuity.

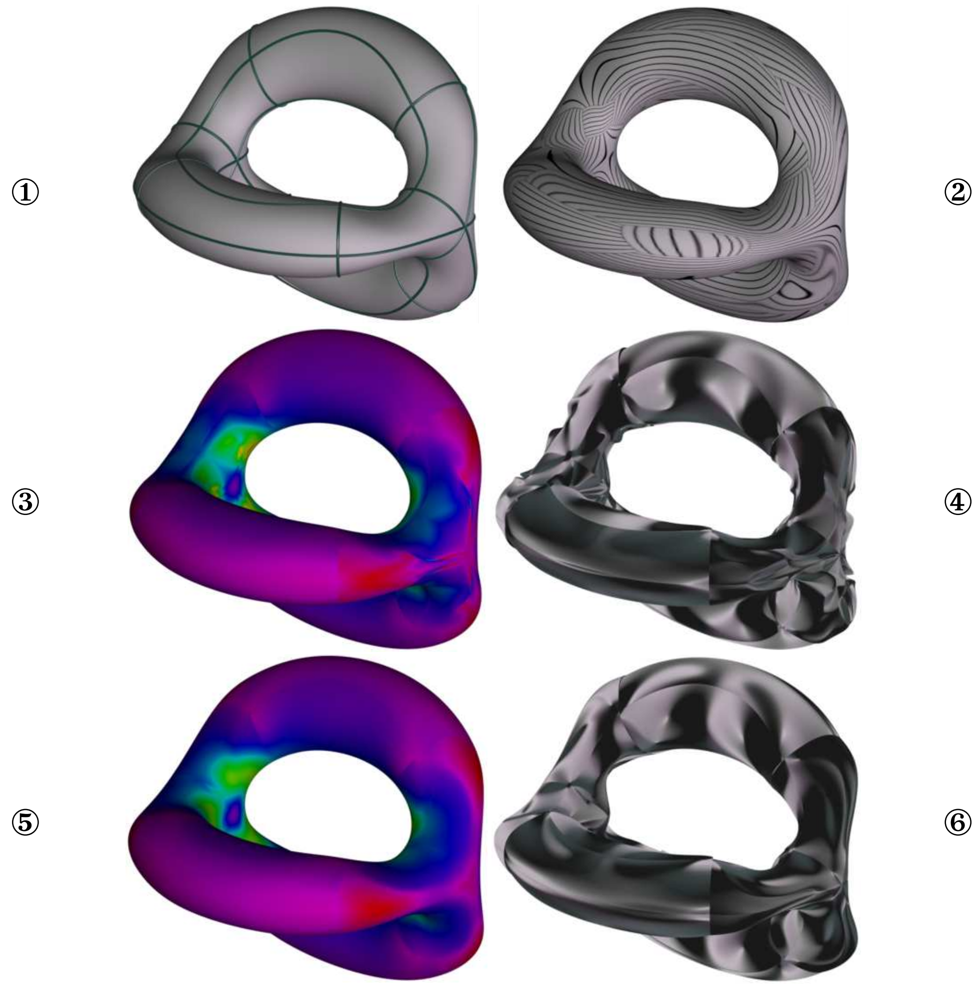


Figure 6.22. Three Handles.

The MVN used in the computation of the surfaces is shown in ①. ② is a rendering of the MVS with lines of reflection, demonstrating G^1 and ~G^2 continuity. ③ and ④ show the Gaussian pseudo-colored and Gaussian offset surface resulting from the surface computed using strain energy as an objective function. ⑤ and ⑥ illustrate these same rendering styles for the surface computed using curvature variation as an objective function. Curvature is more evenly distributed across the MVS.

example of one such surface; it was computed with only four positional constraints, one at each vertex of the underlying tetrahedron. Once implemented, the SI-MVS will allow us to explore this family of surfaces without imposing any constraints.

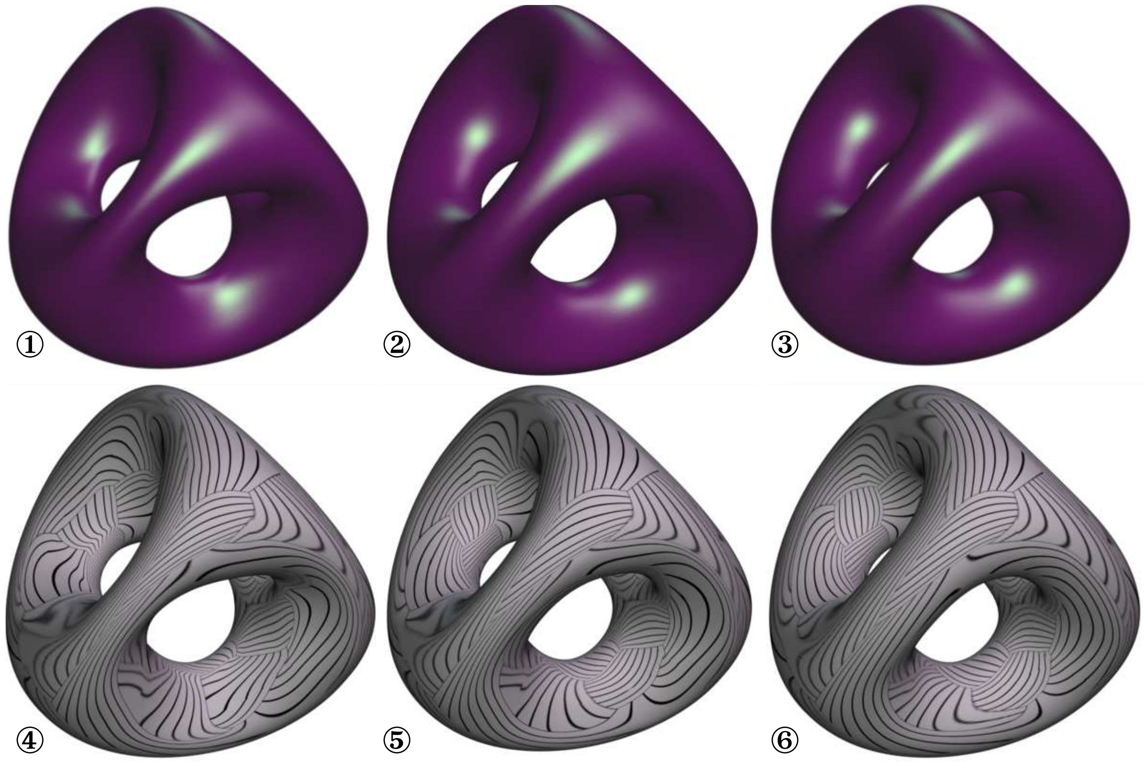


Figure 6.23. Flexible Frame Comparison

①-The MVS with fixed curve network. ②-The MVS with flexible network. ③-The MVS with only two positional constraints, to control scale, plus symmetry constraints.④-⑥ The surfaces are rendered with lines of reflection. The surfaces with “more flexibility” have more regular reflection lines.

6.5.6 Expressive Power and Aesthetics

In this section we present additional MVS, exploring their expressive power and their aesthetics.(Fig. 6.25). Our first example is a surface computed from just two positional constraints. The surface is represented by a single biquintic patch. In the second example, we computed a series of four patches tied in a knot. The knot is specified by five pairs of points, positioned symmetrically along the knot. Next we examine the surface resulting from a single constraint; position, surface tangent, and curvature are specified (Fig. 6.26). This surface specification is ambiguous, the curvature specified is hyperbolic allowing any torus or general cyclide to be formed reducing the MVS functional to zero. Since the polynomial patch used to model this surface is not capable of analytically forming a cyclide, the surface formed is the surface, from among the cyclides satisfying the

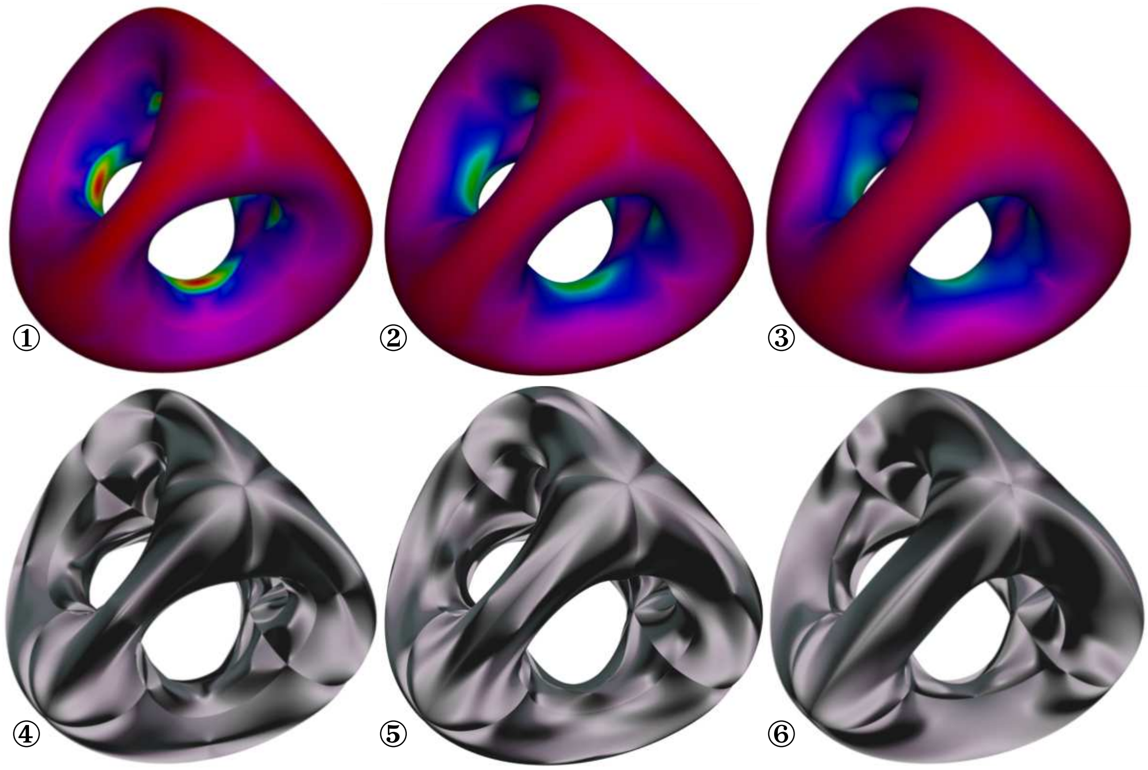


Figure 6.24. Flexible Frame Comparison (cont.)

Pseudo-colored and functional offset renderings of the three surfaces. Curvature concentrations appear on the interior of the legs of the MVS with fixed frame. Generally, the MVS with flexible frame has more evenly distributed curvature. Likewise the MVS with movable points has the most evenly distributed curvature.

constraints, best approximated by the single biquintic patch. In Figure 6.26① the Bézier control hull for the patch is shown to illustrate the placement of control points; it is infeasible to interactively sculpt a surface of such quality by dragging control points with a mouse or other pointing device.

As our last illustration in this section, the *TetraThing*, described in section 6.5.5, is rendered with an environment map created from a photograph of a café, taken with a fish-eye lens (Fig. 6.27). Two copies of the 180° image were used to form the 360° environment. The reflections of the café are fluid and free of irregularities. This last example, fitting a surface to a tetrahedral frame, further demonstrates the versatility, fairness and power of MVS to solve difficult interpolation problems.

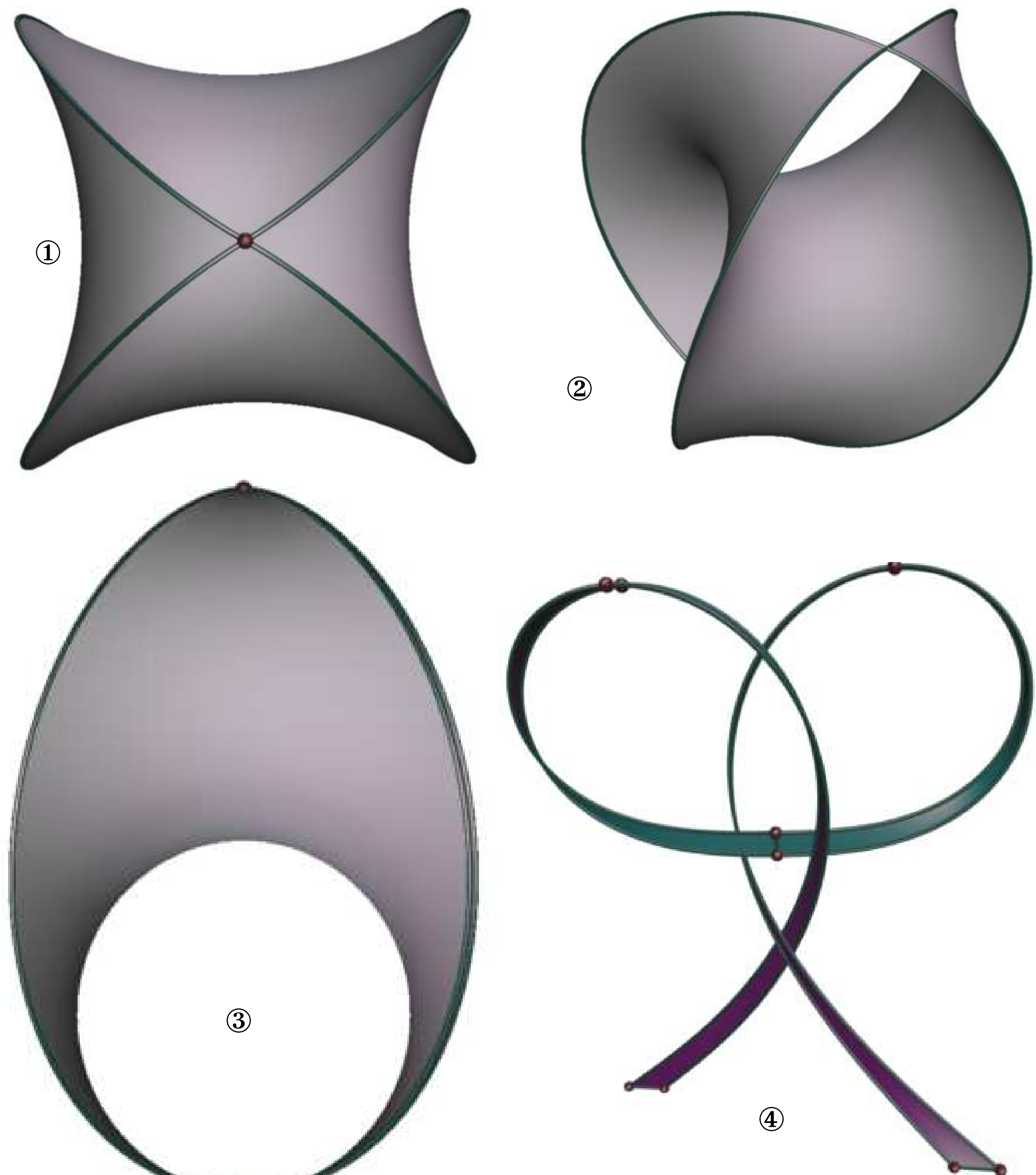


Figure 6.25. MVS Sculptures.

① top view of a surface defined by two point constraints, all other degrees of freedom were used to optimize the surface. ②,③ an isometric and side view of the same surface. ④ a ribbon tied in a knot, the surface is defined by five pairs of points.

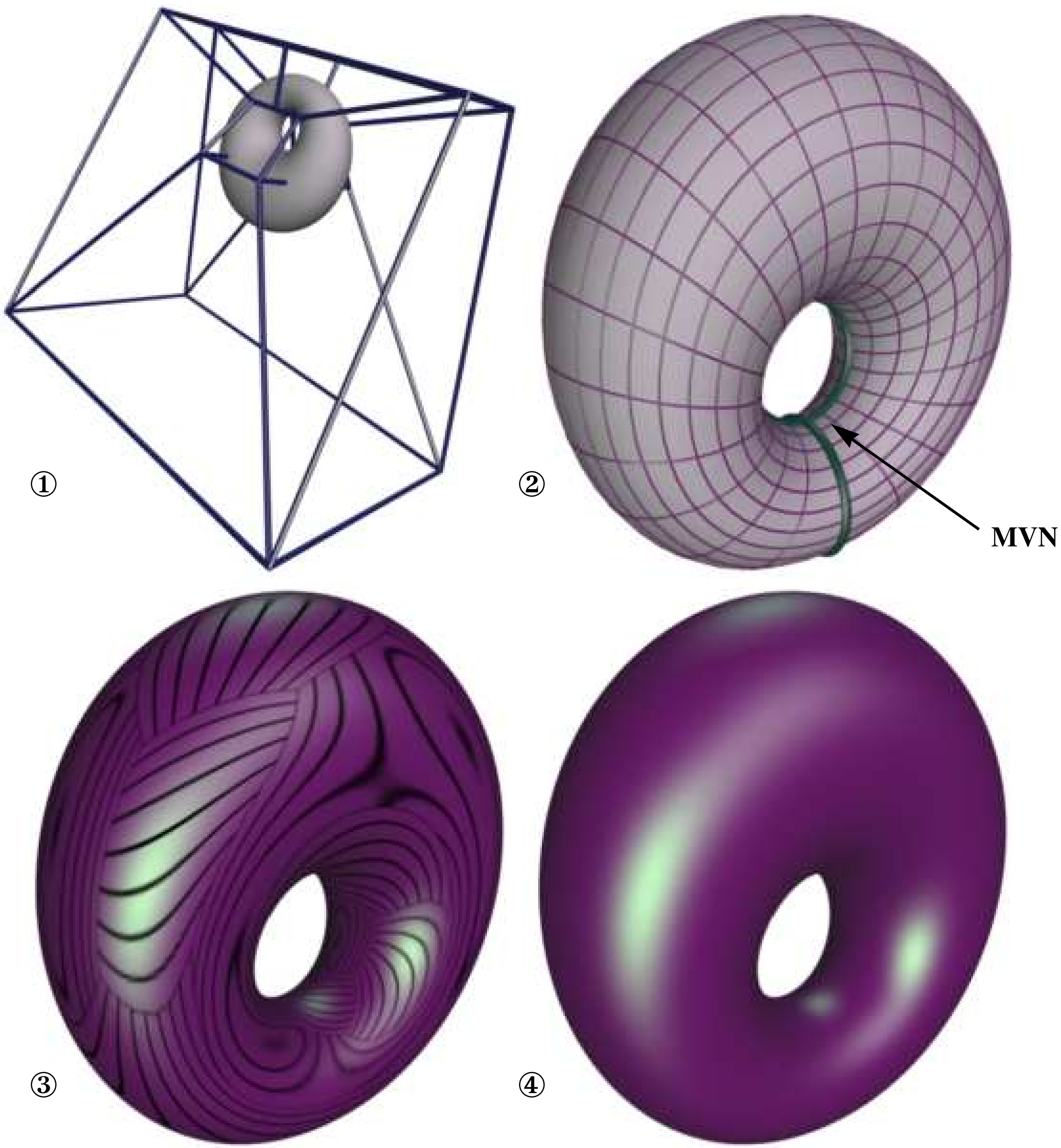


Figure 6.26. One Constraint — One Patch — One Surface

A surface specified by a single constraint set. The surface position, tangent plane, and curvature are specified at a single point. ① The Bézier control hull of the single biquintic patch computed for the constraint set. ② isoparametric lines and the MVN framework consisting of only two circles. ③Lines of reflection. ④ A specular rendering.

Figure 6.27. TetraThing.

6.6 Efficiency

In this section we discuss the cost of computing MVS, what part of the optimization process dominates the expense, how to estimate relative computation times, and how computation times might be reduced. It is logical to discuss computational expense in terms of the number of times that the functional must be evaluated at a point on the surface.

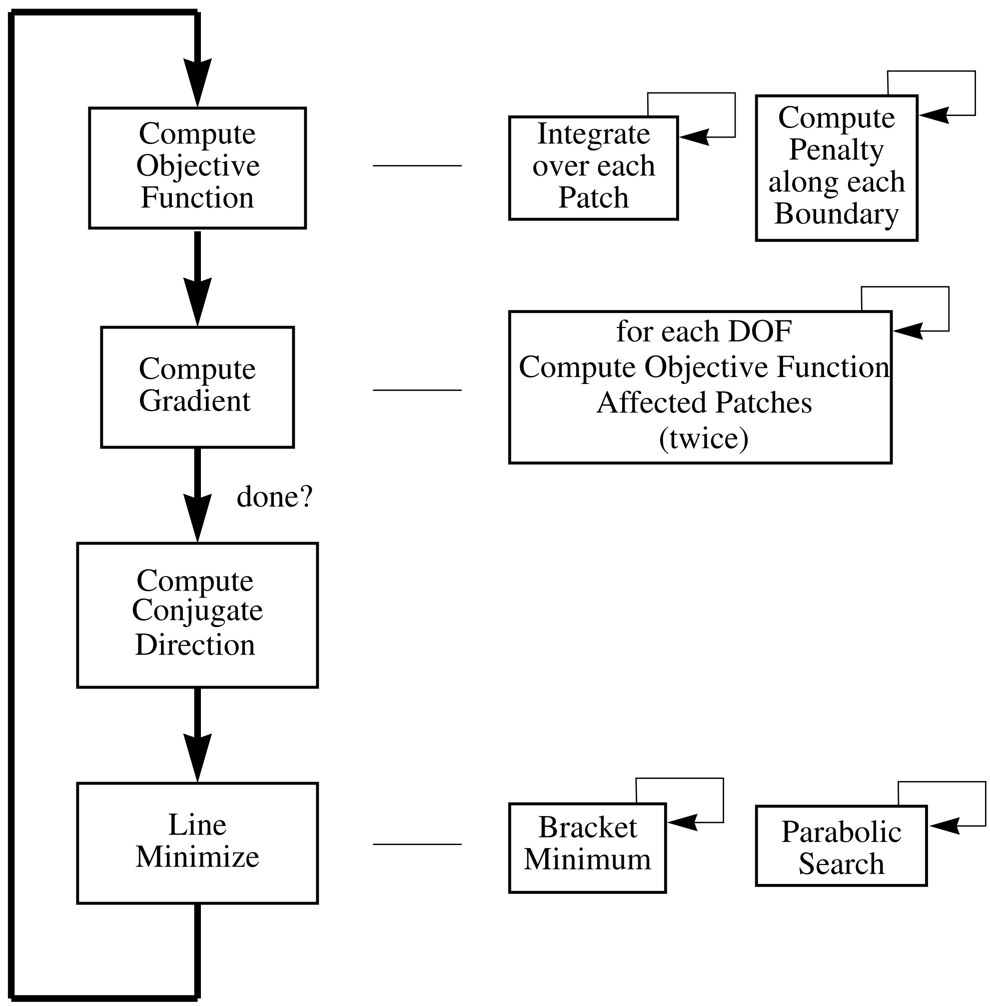


Figure 6.28. Optimization Overview

First we examine the optimization procedure at a high level (Fig. 6.28). Each iteration of the optimization begins by computing the current value of the objective function; each patch in the quilt is integrated, this requires 200 functional evaluations for every triangular patch, and 400 evaluations for every tensor product patch. The computation of the penalty functions is of negligible cost, since we compute all of the required quantities during the integration phase.

The computation of the current gradient completely dominates the computation time. For each degree of freedom, the objective function of each affected patch must be recalculated twice. Depending on the type of variable, the number of patches affected ranges from one to several. If the degree of freedom is interior to a single patch then only that patch is affected. If the DOF is part of a boundary curve definition, then two patches are affected. In the costliest case, if the DOF is part of a vertex definition, then all patches incident to the vertex must be taken into account. From this it should be apparent that it is possible to compute an MVS with fixed curve network much more quickly than a surface with only point constraints. In the case of the flexible MVN, not only is the number of DOF increased, but the cost of computing the associated partial derivatives is disproportionately higher because the number of patches associated with each DOF is higher. The cost of computing the conjugate direction and performing the line minimization is comparatively low.

We now consider a concrete example, the computation of the MVS sphere in Figure 6.16. Because of its symmetry, this problem has only twelve DOF, and each of these is limited to the interior of a patch. In Figure 6.29 we display a plot of the number of iteration vs. the log of the objective function, normalized by the same method as employed in Figure 4.23. The optimization, including problem input, initialization, optimization, and result output, took approximately 91 seconds to complete, a total of 510 tensor product patch integrations were performed. Assuming no overhead whatsoever, each integration took ~0.18 seconds. Given the total number of iterations, we may also compute what portion of the 510 integrations were devoted to gradient computation. This optimization problem took 17 iterations; ~5.3sec. per iteration. Each iteration required a partial derivative computation for each of 12 DOF; each partial derivative requires two integrations. This accounts for a total of 408 integrations, roughly 80% of the computation. This percentage is conservative, since as the number of DOF grows, the number of integrations required for gradient computation increases much more quickly than the integrations required for line minimization and objective function calculation. To understand why this is the case, consider two optimization problems, one of which requires one more tensor product patch than the other. In the larger problem, we gain at least 48 degrees of freedom corresponding to the 16 control points on the interior of the additional patch, These 48 additional DOF require 96 integrations to the gradient computation, while the additional patch only introduces one additional integration in the evaluation of the objective function and one additional integration in the line minimization loop.

This particular optimization problem converges quickly. This is due to the fact that the solution has zero “energy”. The computation of more general surfaces, such as the one defined by two positional constraints (Fig. 6.25②), converge more slowly (Fig. 6.30). In this case the optimization has 29 degrees of freedom; these DOF are from the curvature at the corners of the patch, the curves of the network, and the interior of the patch, all

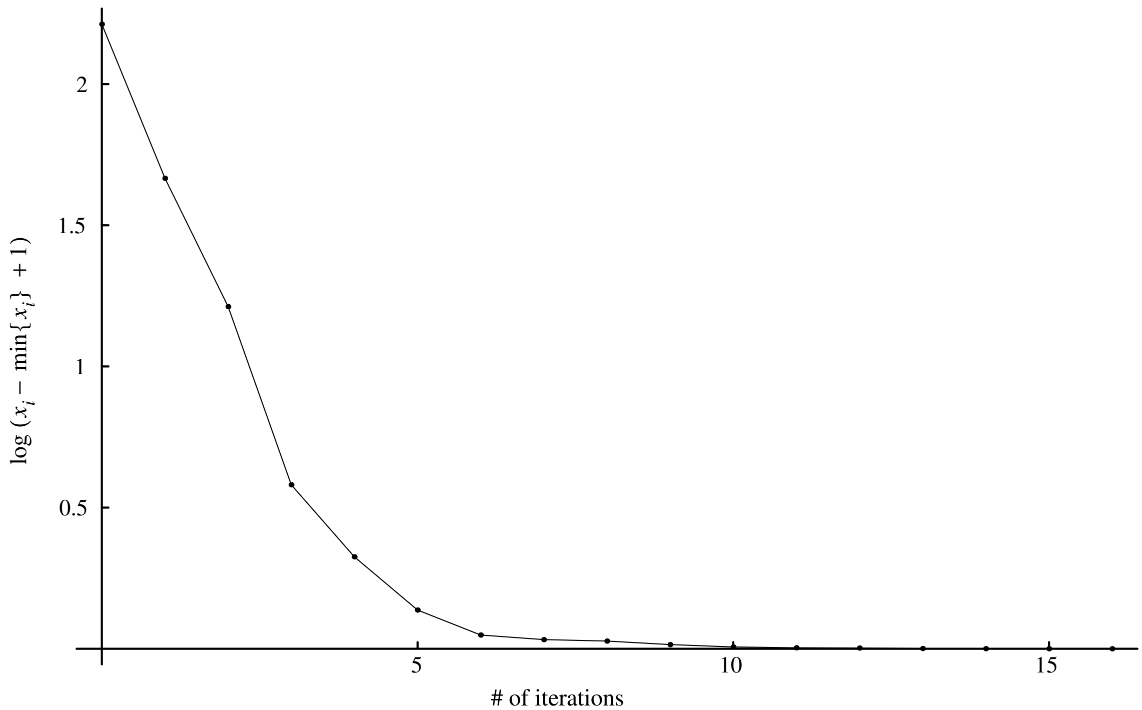


Figure 6.29. Sphere—# Iterations vs. Log(functional)

Here we show the convergence of the optimization of the MVS sphere in Figure 6.16. The optimization took approximately 91 seconds to complete. Taking advantage of symmetry, the number of DOF was reduced to 12.

reduced by symmetry. The optimization took about 30 minutes to complete, and used a total of 8646 integrations distributed over 136 iterations. To compute the fraction of this total that is devoted to derivative calculation, we multiply twice the number of DOF by the number of iterations yielding a total of 7888 integrations, or about 91% of the total.

From these examples, we see that improvement in the efficiency of the integration of the functional will directly impact the efficiency of the algorithm. Counter-intuitively, we have found that reducing the number of integration points actually slows the optimization process. This is because the accuracy of the gradient computation suffers and the descent scheme spends lots of time "wandering around". In order to improve performance, work needs to be aimed at restructuring the functional so that it may be integrated analytically.

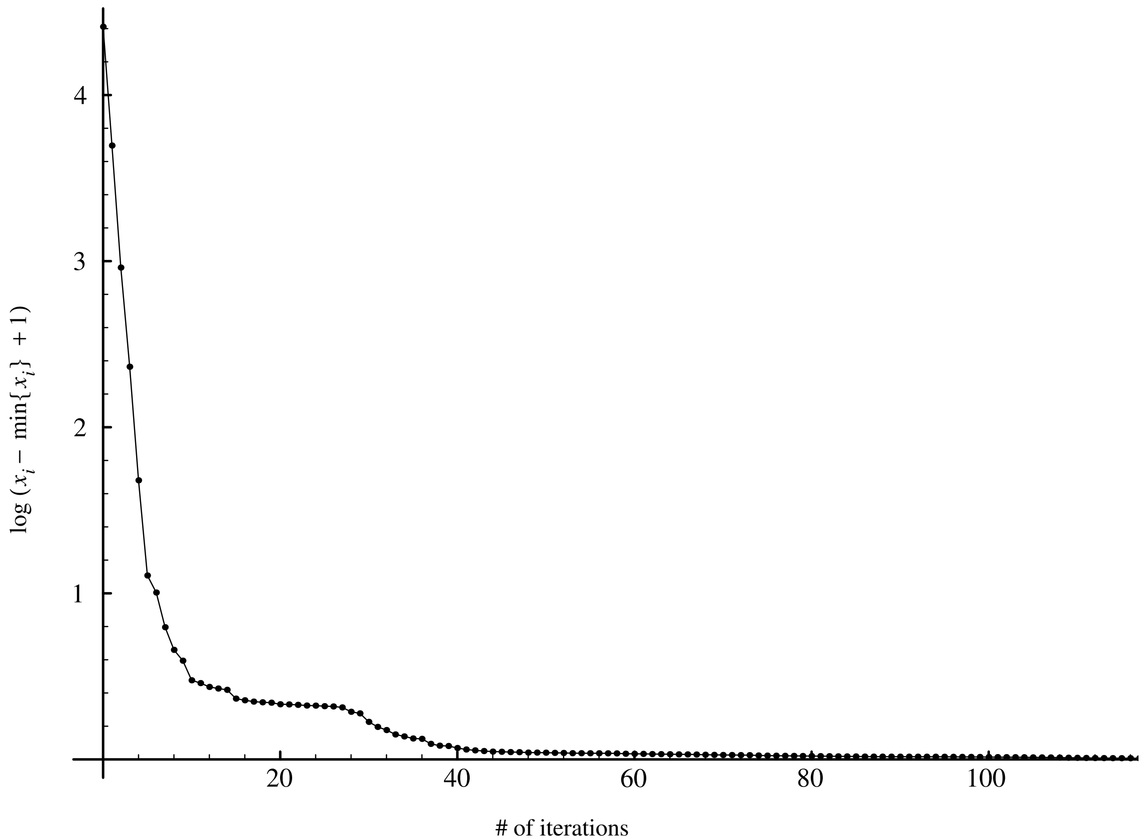


Figure 6.30. OnePatch—#iterations vs. Log(functional)

This plot illustrates the convergence of the optimization of the MVS sculpture in Figure 6.25. The optimization took approximately 30 minutes to complete. Taking advantage of symmetry, the number of DOF was reduced to 29.

6.7 Summary

Constructing a quilt of G^2 or just G^1 continuous surface patches is known to be a difficult task, and it is even harder to shape such a quilt into a satisfactory surface. The use of a general optimization procedure with suitable penalty functions greatly simplifies both tasks.

In this chapter 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.

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 optimization 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.