Variational Surface Modeling

William Welch and Andrew Witkin
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{welch|witkin}@cs.cmu.edu

Abstract

We present a new approach to interactive modeling of free-form surfaces. Instead of a fixed mesh of control points, the model presented to the user is that of an infinitely malleable surface, with no fixed controls. The user is free to apply control points and curves which are then available as handles for direct manipulation. The complexity of the surface's shape may be increased by adding more control points and curves, without apparent limit. Within the constraints imposed by the controls, the shape of the surface is fully determined by one or more simple criteria, such as smoothness. Our method for solving the resulting constrained variational optimization problems rests on a surface representation scheme allowing nonuniform subdivision of B-spline surfaces. Automatic subdivision is used to ensure that constraints are met, and to enforce error bounds. Efficient numerical solutions are obtained by exploiting linearities in the problem formulation and the representation.

Keywords: surface design, constrained optimization, interaction techniques

1 Introduction

The most basic goal for interactive free-form surface design is to make it easy for the user to control the shape of the surface. Traditionally, the pursuit of this goal has taken the form of a search for the "right" surface representation, one whose degrees of freedom suffice as controls for direct manipulation by the user. The dominant approach to surface modeling, using a control mesh to manipulate a B-spline or other tensor product surface, clearly reflects this outlook.

The control mesh approach is appealing in large measure because the surface's response to control point displacements is intuitive: pulling or pushing a control point makes a local bump or dent whose shape is quite easily controlled by fine interactive positioning. Unfortunately, local bumps and dents are not the only features one wants to create. For example, almost anyone who has used a control mesh interface has had the frustrating experience of trying to make a conceptually simple change, but being forced in the end to precisely reposition many—even all—the control points to achieve the desired effect.

This sort of problem is bound to arise whenever the controls provided to the user are closely tied to the representation's degrees of freedom, since no fixed set of controls can be expected to anticipate all of the users' needs.

The work we will describe in this paper represents an effort to escape this kind of inflexibility by severing the tie between the controls and the representation. The model we envision presenting to the user is that of an infinitely malleable piecewise smooth surface, with no fixed controls or structure of its own, and with no prior limit on its complexity or ability to resolve detail. To this surface, the user may freely attach a variety of features, such as points and flexible curves, which then serve as handles for direct interactive manipulation of the surface.

Within the constraints imposed by these controls, surface behavior is governed not by the vagaries of the representation, but by one or more simply expressed criteria—that the surface should be as smooth as possible, should conform as closely as possible to a prototype shape, etc.

Our choice of this formulation is motivated by the desire to present a simple representation-independent facade to the user; however, maintaining the facade is anything but simple. Formally, our approach entails the specification of surfaces as solutions to constrained variational optimization problems, i.e. surfaces that extremize integrals subject to constraints. To realize our goal of forming and solving these problems quickly enough to achieve interactivity, yet accurately enough to provide useful surface models, we must address these key issues:

- **Surface Representation.** We require a surface representation that is concise, yet capable of resolving varying degrees of detail with no inherent limit to surface complexity; that is capable of representing C^n surfaces (in practice we are usually content with C^2 continuity) and that supports efficient solution of the constrained optimization problems we wish to solve. On the other hand, since the representation is to be hidden from the user, we do not require the surface to respond in an intuitive or natural way to direct control-point manipulation.

- **Constrained Optimization** We must be able to accurately and efficiently impose and maintain a variety of constraints on the surface, including those requiring the surface to contain a curve, or requiring two surfaces to join along a specified trim curve. Such constraints raise special problems because the constraint equation involves an integral which must be extremized. Subject to the constraints, we must be able to extremize any of a variety of surface integrals—to create fair surfaces, minimize deviation from a specified rest shape, etc.

- **Automatic Refinement** To create surfaces that reflect the variational solution, without letting the limitations of the representation show through, the resolution of the surface representation must be automatically controlled. Ideally, subdivision should be driven by a measure of the error due to the surface approximation. As constraints are added, additional degrees of freedom must be provided to allow all constraints to be satisfied simultaneously without ill conditioning. Unlike point constraints, which can be met exactly, integral constraints require subdivision to bring their approximation error within a specified tolerance. Additional subdivision should be driven by estimates of the error with which the constrained variational minimum is approximated.

In this paper we report on our progress to date in pursuing the substantial research agenda that these requirements define. Following a discussion of background and related work, we will address each of the issues outlined above. First, the need to compactly represent arbitrarily detailed surfaces leads us to consider schemes for locally refinable representations. Although many have been developed, none meets all of our requirements. We describe a surface representation based on sums of tensor-product B-splines at varying levels of detail. Next we consider the constrained optimization problem itself. We give formulations for several quadratic objective functions, and discuss linear constraints for controlling arbitrary points and curves on the surface. We then turn to the problem of automatic surface refinement based on two kinds of approximation error: objective function error, and constraint error. Finally, we describe a preliminary implementation and present results.

2 Background

2.1 Direct control of curve and surface points

The limitations of control meshes as interactive handles have been noted before. To address them, Fowler and Bartels[11, 12] present techniques that allow the user to directly manipulate arbitrary points on linear blend curves and surfaces: the curve/surface is constrained to interpolate the grabbed point. As the point is moved interactively, the change to the control points is minimized subject to the interpolation constraint. Parametric derivatives are also presented to the user for direct manipulation, to control surface orientation and curvature at a point. Moving beyond point constraints, Celniker and Welch[6] presented a technique for freezing the shape of the surface along an embedded curve, although the issues involved in having the surface track a moving control-curve were not addressed.

2.2 Nonuniform surface refinement

One of our key requirements is the ability to represent smooth surfaces with no *a priori* limit on the detail that can be resolved. Although a number of nonuniform refinement schemes have been developed, no existing one meets all of our needs. Most of these fail to provide C^2 continuity we require. In computer graphics, Bezier patches [8] have been most widely used for nonuniform refinement. In general, however, higher-order continuity between Bezier patches is not preserved if they are manipulated after subdivision, though [20] formulates adaptive Bezier patch refinement with G^1 continuity. Triangular patches, which support topologically irregular meshes, are widely used in finite element analysis, but have been restricted to first-order continuity. Recent developments[9] point to triangular B-spline patches as a way of constructing a surface with higher-order continuity across a triangular mesh, although a computationally efficient refinement scheme for such a representation has not yet been presented.

Forsey [10] presents a refinement scheme that uses a hierarchy of rectangular B-spline overlays to produce C^2 surfaces. Overlays can be added manually to add detail to the surface, and large- or small-scale changes to the surface shape can be made by manipulating control points at different levels. The hierarchic offset scheme may be well-suited to direct user manipulation of the control points, but it does not meet our need for a refinable substrate for constrained variational optimization. One of the fundamental advantages of conventional tensor product surfaces is linearity: surface points and derivatives are linear functions of the control points. Under Forsey's formulation linearity is lost because unit normals are used to compute offsets. We depend heavily on linearity in later sections; use of the hierarchic offset representation would have a devastating impact on performance.

2.3 Constrained optimization

Variational constrained optimization plays a central role in the formulation of so-called natural splines, piecewise cubic C^2 plane curves that interpolate their control points. The proof that natural splines minimize the integral of second derivative squared subject to the interpolation constraints frequently appears as a demonstration problem in the calculus of variations[22].

Surface models based on variational principals have been widely used in computer vision to solve surface reconstruction problems, in which a surface is fit to stereo measurements, noisy position data, surface orientations, shading information etc. [14, 15, 24]. Similar formulations have been employed in computer graphics for physically based modeling of deformable surfaces [23]. All of these are based on regular finite difference grids of fixed resolution.

Constrained optimization based on second-derivative norms has been used in fairing B-spline surfaces[18]. Moreton[19] minimizes variation of curvature to generate surfaces which skin networks of curves while seeking circular or straight-line cross-sections. Such schemes can give rise to very fair surfaces, but the nonlinearity of their fairness metrics prevents them from being used for interactive surface design.

Celniker [5] proposed a physically-based model for interactive free-form surface design, in which the surface is modeled using a C^1 mesh of triangular patches, and positions and normals may be controlled along patch boundaries. Interactivity is possible because the surface fairing problem is formulated as a minimization of a quadratic functional subject to linear constraints. Our approach is closely related in this respect, although we consider more general formulations for both surface functionals and shape control constraints.

3 Surface Representation

We require a representation for smoothly deformable surfaces, which has no *a priori* limit on the detail that can be resolved. Further, we require that points on such a surface be linear functions of its shape control parameters, yielding a more tractable control problem.

Tensor-product B-splines[8] conveniently represent C^n piecewise polynomial surfaces as control-point weighted sums of nonlinear shape functions, and they form the basis of our representation scheme. Unfortunately, the standard tensor-product construction does not allow detail to be nonuniformly added to the surface through local refinement. We instead represent such a locally refined region as a sum of the original surface and smaller, more finely parameterized surfaces. Surface patches at various levels are evaluated and summed to compute the nonuniform surface's value. This is related to Forsey's overlay scheme for B-spline surface refinement [10], but the formulation is much simpler because there is no notion of hierarchic offsets for overlays. The nonuniform surface is a simple sum of sparse, uniform surface layers, which may overlap in arbitrary ways. Further, the resulting surface shape remains a linear function of the control-points, leading to a tractable surface control problem.

A degree$-r$ tensor-product B-spline surface span is formulated as

$$\mathbf{w}(u,v) = \sum_{i}^{r+1} \sum_{j}^{r+1} N_i(u) N_j(v) P_{ij}, \qquad (1)$$

where $\mathbf{N}(t)$ is the vector of $r+1$ uniform B-spline basis functions evaluated at t, and $\mathbf{P}$ is an $(r+1) \times (r+1)$ array of control-points. A mesh of such spans will form a C^{r-1} composite surface if neighboring spans share r rows of control-points along their common boundary. A $U \times V$ array of control-points thus yields a $(U-r) \times (V-r)$-span composite surface.

It is easy to get lost in the sea of summations and indices when working with composite surfaces and formulas involving (1). To simplify notation in this and following sections, we take advantage of the fact that the X, Y, and Z dimensions of the surface in this representation scheme may be treated independently, and state our formulas in one dimension only. Further, we will represent the nth uniform surface layer w^n in terms of a $1 \times (U_n V_n)$ *control vector* $\mathbf{p}$, which is related to the $U_n \times V_n$ control-point matrix $\mathbf{P}$ by the structure-flattening constant $\mathbf{S}$:

$$P_{ij} = \sum_k S_{ijk} p_k,$$

where $S_{ijk} = 1$ if P_{ij} corresponds to p_k, and is 0 otherwise (S converts between a matrix and eg. its row-major representation). Then

$$b_k(u,v) = \sum_i \sum_j N_i(u) N_j(v) S_{ijk}$$

is a vector of basis functions, and its dot-product with the control vector yields the value of the surface at that point. A nonuniform surface w represented as the sum of n uniform surfaces at varying levels of refinement is then

$$\begin{aligned} w(u,v) &= \sum_i^n \mathbf{p}^i \cdot \mathbf{b}^i(u,v) \\ &= \mathbf{p}^T \mathbf{b}(u,v) \end{aligned} \qquad (2)$$

where $\mathbf{b}$ and $\mathbf{p}$ are concatenations of the individual layers' basis and control vectors into respective global vectors.

Note that although each level's parameterization spans the (u,v) domain of the base surface, its control-vector will in general be sparse, with nonzero entries corresponding to local refinements of the surface. In our implementation, we represent a particular layer of refinement as a sparse plugboard of control-points which exist independently of any spans which reference them. Whenever a new span is created within a particular level, it is connected to the appropriate subset of the control-points for that level. This implements the necessary control-point sharing between adjacent spans. New control-points are created within a layer the first time they are referenced by a span belonging to that layer. Thus, only the regions where refinement has actually taken place within a level are explicitly represented and evaluated.

It remains to ensure that the resulting sum of surfaces has the proper degree of parametric continuity. Although the summed surface is not formulated as a hierarchic offset surface as in [10], Forsey's technique for enforcing continuity over a composite offset surface is still applicable. For any patch belonging to a particular level of refinement, we constrain an r-wide band of control-points associated with the patch's boundary to be 0. This forces each patch's position and $r-1$ derivatives to 0 at the patch boundaries, which is sufficient to guarantee that the summed surface is C^{r-1} regardless of the way in which patches in various layers overlap. When disjoint patches within a particular layer have grown to the extent that they meet at a common boundary, the constraints along that boundary may be discarded and the patches merged into one.

4 Constrained Optimization

We would like to control a surface by attaching points and curves to it, letting the surface interpolate between the controls in an appropriate way, so that the user need not completely specify the surface at every point. Exactly what characterizes desirable surface behavior depends on the application, though there is often the requirement that the surface prefer fair, graceful shapes. Regardless of their particulars, many such behaviors can be cast as minimum principles over the surface. One formulates a measure of "goodness" at each surface point, and then integrates this measure over the entire surface to get a single number which characterizes the desirability of the surface shape under that metric. We then search for surface shapes which optimize this quantity while still satisfying the geometric constraints specified by the user.

More formally, we seek shapes which extremize the integral of a surface metric subject to geometric constraints. Such shapes are not intrinsically linked to any particular surface representation scheme, but exist instead as the solutions to constrained variational optimization problems[22]. Our modeler must construct acceptable approximations to such infinite-dimensional variational surfaces using a finite number of control parameters. The approach taken is to approximate the ideal surface as a piecewise polynomial surface using the nonuniform B-spline representation of the previous section.

In this section we describe techniques for computing the B-spline control-points which optimize surface shape while satisfying user-supplied geometric constraints. We first discuss a number of possible surface metrics (*objective functions*, in optimization parlance). We then formulate point and curve constraints for controlling the surface. Finally, we discuss techniques for solving the resulting constrained optimization problems at interactive speeds.

4.1 Surface Objective Functions

One might choose to extremize any number of functions over a surface to achieve fair shapes. One such function measures how much the surface is stretched and bent by looking at the differential area and curvature at each point[23]:

$$Q(\mathbf{w}) = \int_{\mathbf{W}} \|\mathbf{G}\|_\alpha^2 + \|\mathbf{B}\|_\beta^2, \tag{3}$$

where $\mathbf{G}$ and $\mathbf{B}$ represent the first and second fundamental surface forms, and α and β weight the matrix norms. The α and β terms determine resistance to stretching and resistance to bending, respectively.

The vector and matrix norms in this function make it highly nonlinear, leading to a difficult nonlinear optimization problem. It is therefore common[23, 5, 24] to simplify this objective function by linearizing the matrix norms and $\mathbf{B}$ (this is the *thin plate under tension* model [21, 25]):

$$Q(\mathbf{w}) = \int_{\mathbf{W}} \sum_{i,j=1}^{2} \alpha_{ij} D_i\mathbf{w} D_j\mathbf{w} + \beta_{ij}(D_i D_j \mathbf{w})^2, \tag{4}$$

where $D_i\mathbf{w}$ represents the partial derivative of the surface $\mathbf{w}$ with respect to the ith parameter. The approximation is only accurate near the actual minimum (where higher order terms tend to 0) but it is still well-behaved away from the minimum, and the computational benefits are enormous: for a linear surface representation such as a tensor product B-spline, this simplified objective function is quadratic in the underlying surface degrees of freedom, and we can cast the optimization problem as a constrained least-squares minimization.

First, we formulate (4) in terms of the composite B-spline surface of equation(2):

$$Q(\mathbf{w}) = \int_{\mathbf{W}} \sum_{i,j=1}^{2} \left(\begin{array}{c} \alpha_{ij}\mathbf{p}^T D_i\mathbf{b}\ \mathbf{p}^T D_j\mathbf{b} \\ + \\ \beta_{ij}(\mathbf{p}^T D_i D_j \mathbf{b})^2 \end{array} \right).$$

Since the surface is linear in the control vector $\mathbf{p}$ it may be brought outside and the integration in (u, v) completed to yield a $\dim(\mathbf{p}) \times \dim(\mathbf{p})$ matrix $\mathbf{H}$:

$$\begin{aligned} Q(\mathbf{w}) &= \mathbf{p}^T \int_{\mathbf{w}} \sum_{i,j=1}^{2} \begin{pmatrix} \alpha_{ij} D_i \mathbf{b} \otimes D_j \mathbf{b} \\ + \\ \beta_{ij} D_i D_j \mathbf{b} \otimes D_i D_j \mathbf{b} \end{pmatrix} \mathbf{p} \\ &= \mathbf{p}^T \mathbf{H} \mathbf{p}. \end{aligned} \tag{5}$$

(the symbol $\otimes$ signifies an outer product)

Minimizing (5) yields a value for the control vector $\mathbf{p}$ corresponding to the optimal approximation to the variational surface defined by (4), for the given surface refinement. This is the Rayleigh-Ritz method of approximating a continuum solution with a finite set of continuous linear functions[22]. Clearly a minimal solution to equation (5) is 0 — we must add constraints to keep the surface from collapsing to a point if things are to be at all interesting. As we will see in the next subsection, a variety of geometric constraints can be expressed as linear relations over the control vector. Then the optimization becomes a linearly constrained quadratic minimization, which can be efficiently solved using techniques described in Section 4.3.

But first, consider another possibility for the objective function. Suppose we measure the amount the surface has deformed from some prototype shape. In the absence of constraints, minimizing this deformation metric causes the surface to assume the prototype shape. When constraint handles are attached to such a surface and manipulated, the surface should gracefully deform from this shape.

One way of formulating such a shape attractor is to modify (4) to measure the change in stretch and bending from that of a rest shape $\hat{\mathbf{w}}$ [23, 18]. This yields

$$Q(\mathbf{w}) = \int_{\mathbf{w}} \sum_{i,j=1}^{2} \begin{pmatrix} \alpha_{ij} D_i(\mathbf{w} - \hat{\mathbf{w}}) D_j(\mathbf{w} - \hat{\mathbf{w}}) \\ + \\ \beta_{ij} (D_i D_j (\mathbf{w} - \hat{\mathbf{w}}))^2 \end{pmatrix},$$

leading to

$$Q(\mathbf{w}) = (\mathbf{p} - \hat{\mathbf{p}})^T \mathbf{H} (\mathbf{p} - \hat{\mathbf{p}}) \tag{6}$$

where $\hat{\mathbf{p}}$ is the control vector corresponding to the prototype shape. We must then minimize (6) subject to constraints.

The curve manipulation techniques of [2, 11] can also be cast as a shape-attracting quadratic optimization. They minimize absolute control-point displacement subject to point constraints. That is, their deformation metric is simply $(\mathbf{p} - \hat{\mathbf{p}})^T (\mathbf{p} - \hat{\mathbf{p}})$, where $\hat{\mathbf{p}}$ is the control vector for the current shape. This is not a satisfactory objective function from the standpoint of representation-independent control because, as the authors noted, the local properties of the basis show through. Components of $\mathbf{p}$ which are not directly affected by constraints will always keep their original values, as this produces the minimum "deformation". Thus, the size of the bump raised by pulling on such a surface depends on the level of refinement in the underlying representation.

4.2 Geometric Constraints

The user must be able to attach points and curves to the surface and use them to control the surface shape during sculpting. We implement these handles as geometric constraints which the surface must satisfy while optimizing its objective function. In this section, we discuss two broad classes of geometric constraints: finite-dimensional constraints, which control the surface shape at discrete points, and transfinite constraints, which control the surface shape along embedded curves or sub-regions of the surface.

Within each of these classes, we focus on constraint formulations which are *linear* in the surface control vector, as they lead to a constrained optimization problem which can be solved at interactive speeds. In general, such constraints involve surface features whose surface u, v coordinates remain fixed, so that the B-spline basis functions may be evaluated once to yield linear relationships among the components of the control vector. Thus, constraints involving sliding surface points of attachment are excluded in such formulations.

Within the class of linear constraints, we further restrict our focus to constraint formulations which do not couple the (independent) control vectors for each of the surface's spatial dimensions. Computing an independently constrained solution in each spatial dimension is significantly cheaper than computing a single coupled solution for all dimensions because the system matrices involved grow as the square of the size of the control vector, though for some applications the additional cost may be justified.

4.2.1 Finite-Dimensional Constraints

In sculpting a surface one might specify shape requirements which must hold at a set of discrete points on the surface. Surface point positions, point normal directions, and offset relationships between points are all examples of useful point constraints for controlling the surface. Each generates some fixed number of constraints, depending on the way in which the relationship is formulated.

For example, the constraint that a surface point $\mathbf{w}(u_0, v_0)$ remain fixed at a world-space point x can be written as

$$\begin{aligned} \mathbf{x} &= \mathbf{w}(u_0, v_0) \\ &= \mathbf{p}^T \mathbf{b}(u_0, v_0). \end{aligned}$$

This actually represents three independent constraints, one in each spatial dimension, and $\mathbf{b}(u_0, v_0)$ may be evaluated to yield linear constraints in their respective $\mathbf{p}$'s. Constraints to control various parametric derivatives of the surface, such as tangent vectors at a point, are similarly formulated [11].

4.2.2 Transfinite Constraints

A constraint which involves a one- or two-dimensional surface entity, such as an embedded curve or surface patch, must be formulated as an integral over the entity. For example, given a parametric (u, v) curve $\mathbf{c}(t) = (u(t), v(t))$, the constraint that the surface curve $\mathbf{C}(t) = (\mathbf{w} \circ \mathbf{c})(t)$ align itself with the space curve $\mathbf{D}(t)$ would be written:

$$\int_{\mathbf{C}} (\mathbf{C} - \mathbf{D})^2 = 0. \tag{7}$$

Such a constraint statement is dimensionally infinite, and because our surface representation has only a finite number of control points the surface will in general not be able to exactly satisfy the constraint. We will instead constrain the discretized surface to optimally approximate the constraint in a least-squares sense.

Analogous to the objective functions of the previous section, we formulate a transfinite constraint as a quadratic function which achieves a global minimum when the constraint is satisfied[6].

Thus, equation (7) is at a minimum when its gradient with respect to the control vector is 0, and yields the discretized gradient constraints:

$$\begin{aligned} 0 &= \frac{\partial \int (\mathbf{C} - \mathbf{D})^2}{\partial \mathbf{p}} \\ &= \int_{\mathbf{C}} (\mathbf{C} - \mathbf{D}) \frac{\partial \mathbf{C}}{\partial \mathbf{p}} \\ &= \int_{\mathbf{c}} ((\mathbf{w} \circ \mathbf{c}) - \mathbf{D}) \frac{\partial (\mathbf{w} \circ \mathbf{c})}{\partial \mathbf{p}} \end{aligned}$$

For our B-spline surface this becomes

$$\begin{aligned} 0 &= \mathbf{p}^T \int_{\mathbf{c}} (\mathbf{b} \circ c)(t) \otimes (\mathbf{b} \circ c)(t) - \\ &\quad \int_{\mathbf{c}} \mathbf{D} (\mathbf{b} \circ c)(t), \end{aligned}$$

The integration is completed independently of $\mathbf{p}$ (analytically, or numerically by point sampling), leading to a system of linear constraints in the control vector $\mathbf{p}$. The constraints clamp the surface in a shape which minimizes its least-square deviation from the control curve c. Unacceptably large deviations can be eliminated by refining the parts of the surface through which the curve passes.

The constraints generated by this gradient-clamping operation are not necessarily independent. The shape of the embedded curve in u, v determines number of independent constraint rows generated — an iso-paramteric line only produces $r+1$ independent constraint rows per span, while a zigzag stitch across a span might produce fully $(r+1)^2$ independent rows, so that the space curve would completely control the shape of the underlying surface span. Thus, the presence of dependent constraint rows, while numerically inconvenient, is desirable because it means the surface can interpolate the constraint curve with some of its control points remaining undetermined. The surface is then free to change so as to minimize its objective function or to respond to other sculpting operators while still preserving the constraint.

As with any technique in which the question of linear independence arises, there are delicate numeric issues which must be considered. In particular, a u, v curve with very slight high-degree oscillations will give rise to nearly-dependent constraint rows for an interpolation constraint, and thus will lock down all degrees of freedom in the spans it passes through. When such a curve is attached to a space curve, the surface shape is completely determined by the space curve, and thus the surface can behave arbitrarily badly with respect to the objective function. Such constraint leverage can be reduced by adaptively refining the surface representation. Another technique, *reduced quadrature* [27], involves numerically evaluating the constraint integral by sampling as if the surface representation was of a lower order. Such undersampling leads to a dependent set of constraints, thereby eliminating some of the locking behavior associated with a higher-order integration.

4.3 Linearly Constrained Quadratic Optimization

Having formulated the surface approximation problem with a quadratic objective function and linear constraints, a vast body of optimization literature can be brought to bear. In particular, very efficient techniques exist for enforcing linear constraints in this context, and we discuss two we have used: one method using Lagrange multipliers to enforce a least-squares fit to the constraint matrix, the other using a penalty-based approach.

We seek solution methods for the linearly constrained quadratic optimization problem

$$\begin{array}{l} \text{min } \left\| \tfrac{1}{2} \mathbf{p}^T \mathbf{H} \mathbf{p} - \mathbf{p}^T \mathbf{f} \right\| \\ \mathbf{p} \text{ subject to } \mathbf{A}\mathbf{p} = \mathbf{b} \end{array} \tag{8}$$

where $\mathbf{H}$ is the Hessian of the quadratic metric to be minimized, $\mathbf{f}$ is the gradient optimization term, and $\mathbf{A}\mathbf{p} = \mathbf{b}$ is the system of linear constraints to be satisfied (each row of $\mathbf{A}$ represents a single linear constraint, and the corresponding component of $\mathbf{b}$ is its value). Solution methods generally transform this to an unconstrained system

$$\min_{\widehat{\mathbf{p}}} \left\| \frac{1}{2} \widehat{\mathbf{p}}^T \widehat{\mathbf{H}} \widehat{\mathbf{p}} - \widehat{\mathbf{p}}^T \widehat{\mathbf{f}} \right\|,$$

whose solutions $\widehat{\mathbf{p}}$, when transformed back to $\mathbf{p}$'s, are guaranteed to satisfy the constraints. The unconstrained system is at a minimum when its derivatives are 0, thus we are led to solve the system

$$\widehat{\mathbf{H}} \widehat{\mathbf{p}} = \widehat{\mathbf{f}}$$

to find the minimizing $\widehat{\mathbf{p}}$, then transform it back to $\mathbf{p}$ to recover the constrained minimum solution.

We may make either of two transformations of the problem to an unconstrained optimization. For the first, we reformulate equation (8) by adding a single degree of freedom y_i (a Lagrange multiplier[22]) for each constraint row $\mathbf{A}_i$ and we then minimize the unconstrained

$$\min_{p} \left\| \frac{1}{2}\mathbf{p}^T\mathbf{H}\mathbf{p} - \mathbf{p}^T\mathbf{f} + (\mathbf{A}\mathbf{p} - \mathbf{b})^T\mathbf{y} \right\| .$$

Differentiating with respect to $\mathbf{p}$ then $\mathbf{y}$ leads to the augmented system

$$\begin{vmatrix} \mathbf{H} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{vmatrix} \begin{vmatrix} \mathbf{p} \\ \mathbf{y} \end{vmatrix} = \begin{vmatrix} \mathbf{f} \\ \mathbf{b} \end{vmatrix}, \qquad (9)$$

which, though no longer positive-definite, does determine the unique minimum which satisfies the constraints provided the constraint matrix contains no dependent rows. This method has the advantage that it enforces the constraints exactly, in contrast with the penalty method below. This is at the expense of adding an additional variable y_i for each constraint to be enforced. Also, an initial reduction of the constraint matrix to a set of independent rows must be performed. Note that if a least-squares fit to the constraints is to be enforced, we must actually form the normal matrix $\mathbf{A}^T\mathbf{A}$, and take its independent rows as our constraints.

A second solution method associates a penalty term with each constraint, so that the minimization becomes

$$\min_{\widehat{\mathbf{p}}} \left\| \begin{pmatrix} \mu\mathbf{A} \\ \mathbf{H} \end{pmatrix} \mathbf{p} - \begin{pmatrix} \mu\mathbf{b} \\ \mathbf{f} \end{pmatrix} \right\|, \qquad (10)$$

where μ is a large positive weight and the solution to the resulting unconstrained minimization approaches the true constrained minimum as $\mu \to \infty$. In computing a least-squares solution to (10), we can avoid the numerical conditioning problems associated with forming the constraint normal matrix $\mathbf{A}^T\mathbf{A}$ by instead performing a QR factorization[13] of the matrix in (10). Since μ must be chosen small enough to leave a well-conditioned problem, the solution to the penalty system can leave an unacceptably large constraint residual. This can be reduced by performing additional minimization steps on the residual using the same factored matrix[26].

An advantage to formulating the constrained minimization in this way is that no new variables are added to the system. This is offset, however, by the need to perform additional solver steps in refining the residual. A more important advantage to the formulation is that dependent constraint rows need not be eliminated prior to building and factoring the augmented matrix. This makes it straightforward to use factorization update techniques [3, 4] to incrementally update the factorization of the system matrix as surface constraints are added or deleted.

4.4 Automatic Refinement

We are using a piecewise polynomial surface with a finite number of control parameters to approximate an infinite-dimensional variational surface. To maintain a representation-independent facade, we must be able to control the error introduced by this approximation. We must be concerned with two kinds of approximation error. First, the discretized surface may be unable to satisfy all constraints simultaneously (*constraint error.*) Second, even if all constraints are met, the discretized surface may fail to achieve the variational minimum (*objective function error.*)

Of the two, objective function error is more difficult to handle because it cannot be measured directly[1]. How to estimate this kind of error *a priori* is an open research problem [27, 16]. More widely used are *a posteriori* methods in which an estimate is obtained by comparing higher-order solutions to lower-order ones. [1, 17, 7]

In contrast, constraint error can generally be measured directly, by calculating point-to-point or curve-to-curve distances. A straightforward refinement scheme is to compute the constraint error per surface span, refining those spans whose error exceeds a specified tolerance.

5 Results

An interactive free-form surface modeler implementing the techniques described in this paper has been developed at CMU. The modeler, which runs on Silicon Graphics Iris workstations, allows the user to interactively manipulate variational curves and surfaces, controlling and combining them through a variety of constraints and objective functions.

By default, surfaces minimize the fairness integral given in equation (4) subject to the constraints acting on them. At any time, the user may install a surface's current shape as its rest shape, or "melt" a previously remembered rest shape. We have found that additional objective function terms can be useful as interactive sculpting tools—for instance, terms which inflate or deflate the surface.

Basic surface control is provided by point constraints, which take the place of conventional control points. Ephemeral position constraints allow the user to grab and drag arbitrary points on the surface, while persistent constraints help define surface shape. Additional control is afforded by surface normal constraints.

Curve constraints have proven to be a far more powerful modeling tool. To create a surface or space curve, the user defines a sequence of surface or space points, from which an interpolating curve is constructed. Once created, curves become first-class variational objects that can be controlled by constraints. To use curves as surface controls, a surface curve is attached to a space curve, which may be manipulated by the user. This attachment can be established in either of two ways. First, a surface curve may be snapped to an independently created free-standing

[1] unless the error-free answer is available for comparison!

Figure 1: Surface modeling with curve constraints. Upper left: the user creates a surface curve on a sheet. The push-pins identify point constraints, which keep the surface from collapsing to a point. Upper right: the surface has been trimmed. The surface curve defining the boundary has been promoted to a space curve which can be independently controlled by the user. Lower left: Point constraints are applied to the space curve to modify its shape. The surface, which is now constrained to contain the space curve, deforms to follow the curve. Lower right: a second control curve is added and manipulated.

space curve, allowing the user to define literal wireframes, then fit surfaces onto them. Second, the user may inscribe control curves on the surface to which automatically created space curves are fit. Figure 1 shows the use of curve constraints. Additionally, the user may impose ribbon constraints that control surface orientation as well as position along the curve.

Curve constraints may also be used to join surface patches by snapping a pair of surface curves to a single space curve. The result is that the two surfaces are constrained to intersect along the space curve. The intersection curve may then be directly manipulated by the user. In this way, surface sheets may be assembled and trimmed against the joined curves to form boundary representations for solids (see figure 2). As a generalization of ribbon constraints, joined surfaces may be subjected to hinge constraints that independently control the surfaces' orientations along the intersection curve.

The modeler employs the refinable surface representation described in section 3. The user may refine the surface manually by selecting regions to be refined, or request automatic refinement based on constraint error. Currently, the system does not perform refinement based on objective function error. The use of automatic refinement is illustrated in figure 3.

References

[1] I. Babuska and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15(4), 1978.

[2] Richard H. Bartels and John C. Beatty. A technique for the direct manipulation of spline curves. In *Proceedings, Graphics Interface*, 1989.

[3] Richard. H. Bartels and Gene Golub. The simplex method of linear programming using lu decomposition. *CACM*, 12(5), May 1969.

[4] Å ke Björck. A general updating algorithm for constrained linear least squares problems. *SIAM J. Sci. and Stat. Comp.*, 5(2), 1984.

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

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

Figure 2: Left: A B-rep solid constructed using variational sheets joined by curve constraints. Intersection curves are formed by constraining curves on each surface to contain a common space curve. Surfaces are then trimmed to the intersection. Right: As the user reshapes the space curve, both adjoining faces follow, maintaining the intersection.

Figure 3: Left: A single bicubic span, subject to point and curve constraints, with large constraint error. Right: the same surface, following automatic refinement driven by constraint error. Error was measured per surface span. Spans exceeding the specified error tolerance were recursively refined.

[7] J. P. De. S. R. Gago, D. W. Kelly, and O. C. Zienkiewicz. A posteriori error analysis and adaptive processes in the finite element method: Part ii – adaptive mesh refinement. *Int. J. Numer. Methods Eng.*, 19:1621–1656, 1983.

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

[9] Philip Fong and Hans-Peter Seidel. Control points for multivariate b-spline surfaces over arbitrary triangulations. *Computer Graphics Forum*, 10:309–317, 1991.

[10] D.R. Forsey and R.H. Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4), August 1988. Proceedings SIGGRAPH '88.

[11] Barry Fowler. Geometric manipulation of tensor-product surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, 1992. (to appear).

[12] Barry Fowler and Richard Bartels. Constraint-based curve manipulation. In *ACM Siggraph Course Notes, Topics in the Construction, Manipulation, and Assessment of Spline Surfaces*, 1991.

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

[14] W.E.L. Grimson. An implementation of a computational theory of surface orientation. *Computer Vision, Graphics, and Image Processing*, 22(1):39–69, 1983.

[15] B.K.P. Horn and M.J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33:174–208, 1986.

[16] D. W. Kelly. The self-equilibration of residuals and upper bound eror estimates in the finite element method. In I. Babuska, editor, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 129–146. Wiley, 1986.

[17] D. W. Kelly, J. P. De. S. R. Gago, and O. C. Zienkiewicz. A posteriori error analysis and adaptive processes in the finite element method: Part i –

error analysis. *Int. J. Numer. Methods Eng.*, 19:1593–1619, 1983.

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

[19] Henry Moreton and Carlo Séquin. Functional minimization for fair surface design. *In these proceedings*.

[20] Francis J. M. Schmitt, Brian A. Barsky, and Wen-Hui Du. An adaptive subdivision method for surface-fitting from sampled data. *Computer Graphics*, 20(4), 1986. Proceedings SIGGRAPH '86.

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

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

[23] Demetri Terzopoulis, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4), July 1987. Proceedings of SIGGRAPH '87.

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

[25] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8:413–424, 1986.

[26] Charles Van Loan. On the method of weighting for equality-constrained least-squares problems. *SIAM J. Numer. Anal.*, 22(5), 1985.

[27] O.C. Zienkiewicz and K. Morgan. *Finite Elements and Approximation*. John Wiley and Sons, 1983.