Surface Design Using Global Constraints on Total Curvature

Michael Plavnik and Gershon Elber
Computer Science Department Technion - Israel Institute of Technology
Haifa, Israel

February 26, 1998

Abstract
Tensor product surfaces are now widely used in application areas from industrial design to computer animation. Yet, the quest for more effective design methods continues. We consider a new approach to the design of free form surfaces using global second order differential geometric constraints defined over the entire domain of the surface or a prescribed subset of it. This work considers two cases of convexity preservation and of developability preservation.

Keywords: *Geometric design, tensor product surfaces, geometric constraints, total curvature.*

1 Introduction

Computer aided geometric design distincts between building a model from already defined surfaces (high level) and designing the surfaces (low level). While modeling requires the modification of the shape of the surface, such operations are usually restricted to regular operators such as rigid motion, stretching/shrinking and bending [2]. In this work, we limit ourselves to surface design. Traditional methods for designing tensor product surfaces include operations upon the underlying mathematical building blocks (control vertices and basis functions) and are sometimes denoted direct manipulation methods. The direct manipulation method is flexible, and simple to implement. Modifications of the shape of the surface are local and tractable, and are very efficient and highly interactive. However, local pretrusions and dents are not the only surface shaping operations one might desire. A conceptually simple change like bending along a diagonal line may force the precise repositioning of many control points. Consequently, there is a constant drive to develop new geometric forms that overcome some of those limitations of tensor product surfaces. Triangular surface patches [1, 11] and multi-sided patches [10] are two examples for different representations that attempt to alleviate some of these difficulties. Nevertheless, the problem of inadequate control over the design of the desired surface 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 user's need.

This paper deals with enforcing constraints on the curvature properties of Bezier surfaces. At the same time, the behavior of the surface is governed by one or more simply expressed objectives, for example the preservation of the shape of the original, unconstrained, surface. Formally, our approach specifies a surface as a solution of a nonlinear programming problem. Novelty of the approach entails the global satisfaction of the constraints *at every point* of the surface's domain, compared to specification at finite number of points or along a curve on the surface, in previous methods [7, 8]. In this paper, we report on our progress in pursuing the goal of tensor product surface modeling using a variety of global second order geometric constraints.

This paper is organized as follows. In Section 2, we present some background information on the use of constraint based methods in the geometric modeling, while in Section 3, we mathematically formulate the problem in hand and narrow the global differential constraints specification to the cases of convex and developable surfaces, and describe the symbolic tools we employ in supporting the implementation of these differential geometric constraints. Section 4 considers two possible objective functions one can employ, while in Section 5, we discuss the numerical solutions. Examples are presented in Section 6, and we conclude in Section 7.

2 Background

Recently, some work have been conducted to incorporate intuitive and interactive controls over the geometrical properties of the surface. We follow [6, 7] and define a differential property that is expressed directly in terms of a surface representation and its derivatives, and a geometrical property which is a set of differential properties. Examples of geometric properties include a point and a curve on the surface, tangent and normal, elastic energy of the surface approximation functionals, etc. Geometric properties provide the mean to achieve desired shape control over the surface without resorting to direct manipulation of the surface's representation. Geometric constraint is a restriction on a property value which should be completely satisfied, subject to some small error tolerance. The constraints allow precise control over (a portion of) the surface, and are therefore the means for a design that follows specifications. Other kinds of restrictions may be formulated that do not require a complete satisfaction. Instead, some objective function may be defined over the property and the minimum of the objective function is sought. Typically, the objective function is either an energy approximation functional [5, 12], a minimal shape change, or minimal control points' motion [6, 7]. Local minima problems may arise when the global minimum is sought and in [2, 5] user's intervention is required to resolve it. Sculpting tools that are implemented as a set of physical forces such as pressure, springs and gravity, and produce qualitative effects like enlarge, attract and flatten are examples of unconstrained restrictions used to manipulate the shape of the surface [5].

Constrained and unconstrained restrictions may be allowed to vary in time by changing existing restrictions and introducing new ones. They are useful when defining operators that preserve some surface properties during the modification process, and operators that do modify other properties of the surface. Integrating both preserving and modification operators means flexibility and precision at the same time. Nevertheless, constraint-based approaches have not been successful in practical interactive systems compared to the direct manipulation methods. In [9], problems with the constrained-based approaches are discussed and a practical approach to partially overcome these problems is proposed.

Interactive systems demand efficient solution. As noted in [5, 8, 12], the current state of computer technology limits the applicable constraints to mostly linear and the objective functionals to quadratic, in the surface degrees of freedom (e.g. control points). The constrained system of equations is usually solved by transforming the constrained problem into an unconstrained problem in which constraints are implicitly satisfied. It may be achieved by a null-projection [6, 8], via Lagrange multipliers [5, 12] or with the aid of penalty methods [5]. When no conflicts exist between the different constraints, they might be completely satisfied. Otherwise, they may be satisfied in a least squares sense [12] or by minimizing an error functional

[7]. To control the constraint's error, automatic or manual refinement of the representation of the surface is used in [5]. Control over the objective function is more difficult, because an error measure is generally unavailable [5, 6].

Welch and Witkin discuss in [5] two broad classes of geometric constraints: finite-dimensional constraints, that control the shape of the surface at a discrete set of points, and transfinite constraints, that control the shape of the surface along embedded curves. Only an interpolation constraint for a curve was presented. All other references address finite-dimensional constraints and to a less extent transfinite constraints. In addition to finite-dimensional and transfinite constraints, we define the third class of *global geometric constraints*, that globally control the properties of the surface at the entire surface's domain or at a prescribed subset of it. Examples include the interpolating of some analytic surface, controlling the normal field of the surface by prescribing the Gauss map as some bivariate vector field function, having a zero Gaussian curvature throughout (developable surface), prescribing elliptic and hyperbolic regions of the surface and constraining the parabolic regions of the surface. In [4], some global shape properties like convexity and a fixed degree of smoothness were explored through a system that automatically generates a surface with a required shape. Early work by Ferguson [13] concentrated on a global convexity control over B-spline surfaces along a prescribed isoparametric direction. A sufficient convexity condition on the isoparametric curves of the surface was very conservative. Floater [14] presented an improved weaker sufficient condition for the convexity of Bezier and B-spline surfaces. However, most of the work on the global properties is concentrated on curves' design. In [15], curvature of the curve and its derivatives are used to constrain aesthetic (visually pleasing) properties of the curve approximation. Direct manipulation of the curvature's profile using numerator and denumerator profiles was described in [16]. Constrained optimization and least squares approximation are employed to integrate best-fit curvature profiles. Combined fairing and local convexity preservation problem for interpolatory splines was solved in [17] using nonuniform $C^2 - continuous$ polynomial splines.

3 Second Order Constraints

Consider the tensor-product Bezier surface:

$$S^{\mathcal{P}}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j}B_{i,j}^{m,n}(u,v), \qquad degree(S^{\mathcal{P}}) = (m,n), \tag{1}$$

where $B_{i,j}^{m,n}(u,v) = B_i^m(u)B_j^n(v)$ are the ith and jth Bezier basis functions of degree (m,n) and $\mathcal{P} = \{P_{i,j}\}$ are the control points. Denote the initial surface as $S^0 = S^{\mathcal{P}_0}(u,v)$. Then, $S^{\mathcal{P}}(u,v)$ denotes the desired new surface, following some changes in the control points that satisfies a given set of total curvature and/or other constraints. Such a surface can be obtained as a solution to the following nonlinear programming problem (NLP), where the control points are the variables of the problem:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : h_k(\mathcal{P}) = 0,\ g_l(\mathcal{P}) \geq 0,\ \forall k,l\}, \end{aligned} \tag{2}$$

where $F(\mathcal{P})$ is some objective function, $h(\mathcal{P})$ is a vector of equality constraints and $g(\mathcal{P})$ is a vector of inequality constraints, and $M = (m+1)(n+1)$, where the control points are in a three dimensional space, $I\!R^3$.

From the designer's point of view, one is interested in shaping a convex surface or a surface that is developable. From the point of view of the underneath representation, the convex surface is a solution of the NLP problem Equation (2) with constraints $g(\mathcal{P})$ coercing the positivity of the total (Gaussian) curvature, K, of the surface:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : K(\mathcal{P},u,v) \geq 0, \ \ \forall u,v \in [0,1]\}. \end{aligned} \tag{3}$$

In order to achieve strict convexity, $K(\mathcal{P},u,v) > 0$ is required. Nevertheless, in order to numerically solve the NLP problem, $K(\mathcal{P},u,v) \geq \epsilon$ is allowed, where ϵ is a small number ≥ 0. Analogously, the developable surface is a solution of the NLP where the total curvature of the surface, K, is prescribed as equality constraints $h(\mathcal{P})$.

$$\begin{array}{ll} \text{minimize:} & F(\mathcal{P}) \\ \text{subject to:} & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!\!R^{3M} : K(\mathcal{P}, u, v) = 0, \;\; \forall u, v \in [0,1]\}. \end{array} \tag{4}$$

Hence, one seeks an efficient way to compute the sign of the total curvature at every point of the domain of the surface. That is, whether the total curvature is everywhere positive, or everywhere zero. For a regular surface, the sign of the total curvature, K, equals the sign of the determinant of the matrix of the second fundamental form of the surface:

$$\overline{K}(u,v) \;\; = \;\; det \begin{bmatrix} \text{e(u,v)} & \text{f(u,v)} \\ \text{f(u,v)} & \text{g(u,v)} \end{bmatrix}, \tag{5}$$

where,

$$\begin{aligned} N(u,v) \;\; &= \;\; \frac{\partial S^{\mathcal{P}}(u,v)}{\partial u} \times \frac{\partial S^{\mathcal{P}}(u,v)}{\partial v}, && degree(N) = (2n-1, 2m-1), \\ e(u,v) \;\; &= \;\; \left\langle \frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial u^2}, N(u,v) \right\rangle, && degree(e) = (3m-3, 3n-1), \\ f(u,v) \;\; &= \;\; \left\langle \frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial u \partial v}, N(u,v) \right\rangle, && degree(f) = (3m-2, 3n-2), \\ g(u,v) \;\; &= \;\; \left\langle \frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial v^2}, N(u,v) \right\rangle, && degree(g) = (3m-1, 3n-3). \end{aligned}$$

To obtain the sign of K, one can compute the sign of $\overline{K}(u,v)$. From Equation (5), one can see that the desired sign of the total curvature constraint, $\overline{K}(u,v)$, is representable as a scalar Bezier surface of degree $(6m-4, 6n-4)$.

$$\overline{K}(u,v) = \sum_{i=0}^{6m-4} \sum_{j=0}^{6n-4} k_{i,j} B_{i,j}^{6m-4,6n-4}(u,v) \tag{6}$$

For example, for a bicubic Bezier surface, $degree = (3,3)$, $\overline{K}$ has $(6 \cdot 3 - 3) \times (6 \cdot 3 - 3) = 225$ coefficients.

However, the representation of the sign of the total curvature by the sign of the second fundamental form has a drawback of being sensitive to a reparametrization and scaling transformations, due to our introduced ignorance in the first fundamental form. This ignorance has hindering effects on solving the NLP problem. To partially overcome this deficiencies, one may down scale the constraints by the values of the determinant of the matrix of the first fundamental form of S^0, computed at the node points of the Bezier basis functions of $\overline{K}(u,v)$, without introducing any substantial overhead in the constraints evaluation. Yet, merely down-scaling the coefficients of the surface is insufficient to improve the algorithm's stability. In addition, we down scale every constraint by the length of the gradient of the constraint. In practice, we found that this process of combined down-scaling works quite well.

Given the control points, $\mathcal{P}$, of the surface $S^{\mathcal{P}}$, one can compute the coefficients of $\overline{K}$. Nevertheless, one is required to determine whether $\overline{K}$ is positive throughout, or zero everywhere. For $\overline{K}(u,v)$ to be identically zero and S to be developable it is necessary and sufficient for all $k_{i,j}(\mathcal{P})$ to be zero. Hence, the coefficients of $\overline{K}$ introduce $(6m-3)\times(6n-3)$ equality constraints, $h(\mathcal{P})$. In contrast, there is no simple procedure to determine whether $\overline{K}$ is positive throughout and S is convex. Alternatively, one can define sufficient conditions for $\overline{K}$ to be positive throughout, if all coefficients of $\overline{K}$ are positive. Thus, once again, the $(6m-3)\times(6n-3)$ coefficients of $\overline{K}$ are used as inequality constraints $g(\mathcal{P})$. For example, for a bicubic Bezier surface, the convex surface constraint problem of Equation (3) is reduced to:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in \mathbb{R}^{3M} : k_{i,j}(\mathcal{P}) \geq 0,\ i,j = 0,..,14\}, \end{aligned} \tag{7}$$

whereas the developable surface constraint problem of Equation (4) is reduced to:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in \mathbb{R}^{3M} : k_{i,j}(\mathcal{P}) = 0,\ i,j = 0,..,14\}. \end{aligned} \tag{8}$$

One can employ symbolic analysis operators over Bezier surfaces [18] to derive the evaluation expressions for $k_{i,j}(\mathcal{P})$.

$$\frac{\partial S^{\mathcal{P}}(u,v)}{\partial u} = \sum_{i=0}^{m-1}\sum_{j=0}^{n} P_{i,j}^{10} B_{i,j}^{m-1,n}(u,v), \tag{9}$$

$$\frac{\partial S^{\mathcal{P}}(u,v)}{\partial v} = \sum_{i=0}^{m}\sum_{j=0}^{n-1} P_{i,j}^{01} B_{i,j}^{m,n-1}(u,v), \tag{10}$$

$$\frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial u^2} = \sum_{i=0}^{m-2}\sum_{j=0}^{n} P_{i,j}^{20} B_{i,j}^{m-2,n}(u,v), \tag{11}$$

$$\frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial u \partial v} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} P_{i,j}^{11} B_{i,j}^{m-1,n-1}(u,v), \tag{12}$$

$$\frac{\partial^2 S^{\mathcal{P}}(u,v)}{\partial v^2} = \sum_{i=0}^{m}\sum_{j=0}^{n-2} P^{02}_{i,j} B^{m,n-2}_{i,j}(u,v), \tag{13}$$

where,

$$\begin{aligned}
P^{10}_{i,j} &= m(P_{i+1,j} - P_{i,j}),\\
P^{01}_{i,j} &= n(P_{i,j+1} - P_{i,j}),\\
P^{20}_{i,j} &= m(m-1)(P_{i,j} - 2P_{i+1,j} + P_{i+2,j}),\\
P^{11}_{i,j} &= mn(P_{i,j} - P_{i+1,j} - P_{i,j+1} + P_{i+1,j+1}),\\
P^{02}_{i,j} &= n(n-1)(P_{i,j} - 2P_{i,j+1} + P_{i,j+2}).
\end{aligned}$$

Substitute by the $\otimes$ symbol, the inner, vector, or scalar product operators over the control points and/or coefficients in $I\!R^3$, $I\!R^3$ and $I\!R^1$ spaces, respectively. Then, one can derive the surface product operator as:

$$\begin{aligned}
S^{\mathcal{P}}(u,v) \otimes S^{\mathcal{Q}}(u,v) &= \sum_{i,j} P_{i,j} B^{m_P n_P}_{i,j}(u,v) \otimes \sum_{i,j} Q_{i,j} B^{m_Q n_Q}_{i,j}(u,v) \\
&= \sum_{i,j} R_{i,j} B^{m_P+m_Q, n_P+n_Q}_{i,j}(u,v) \\
&= S^{\mathcal{R}}(u,v),
\end{aligned} \tag{14}$$

where

$$R_{k,l} = \sum_{i=max(0,k-m_Q)}^{min(m_P,k)} \sum_{j=max(0,l-n_Q)}^{min(n_P,l)} \frac{\binom{m_P}{i}\binom{m_Q}{k-i}\binom{n_P}{j}\binom{n_Q}{l-j}}{\binom{m_P+m_Q}{k}\binom{n_P+n_Q}{l}} P_{i,j} \otimes Q_{k-i,l-j}.$$

The differentiation and product operators in Equations (9)-(14) are algebraic in nature and express the coefficients or control points of the resulting surface through a summation of terms, factored by a scalar weight. A term is the coefficient or control point of the operand surface, as in Equations (9)-(13) or the product of the control points of the operand surfaces, as in Equation (14).

One can consider the coefficients $k_{i,j}(\mathcal{P})$ as multivariate polynomials in the control points $\mathcal{P}$ of the surface $S^{\mathcal{P}}$. Given control points $\mathcal{P}$, one can compute the coefficients of $\overline{K}$ following Equations (9)-(14), independently of the Bezier basis functions. The $k_{i,j}(\mathcal{P})$ constraints are multivariate polynomials of degree six in $P_{i,j}$, and, generally, are not convex functions, enforcing the use of a general NLP solution methods. To efficiently solve NLP problems, gradients of the constraints must be readily available. Numeric evaluation of the gradients requires excessive number of evaluations of constraints, depending on the method of choice. Also, the gradient of the polynomial function may be evaluated as fast as the function itself or even faster. To make the evaluation efficient, symbolic computation tools for Bezier surfaces were developed, implementing the operators defined in Equations (9)-(14), as well as symbolic derivative operators with respect to control points. Such tools allow one to extract pure symbolic expressions for the constraints and their gradients, as well as to explore optimizations based on symbolic manipulations and transformations.

A naive approach to the symbolic computation of $k_{i,j}(\mathcal{P})$ is very similar to the numerical one. Instead of numeric values of control points, $\mathcal{P}$, one processes symbols representing the control points and coefficients. A symbol is a name of the control point together with its indices. The control points can be regarded as the summation of factored terms. Summation of terms can be represented as a sequence of the factored terms. That is, $\{t_0, t_1, ..., t_s\}$ denotes $t_0 + t_1 + ... + t_s$. Operators (9)-(13) take the control points of the operand surface and construct the control points of the resulting surface by literally merging sequences of the factored terms of the corresponding control points or coefficients. The product operators (14) are more complicated. $P_{i,j} \otimes Q_{k-i,l-j}$ can be expanded by distributing the operator $\otimes$ over terms in $P_{i,j}$ and $Q_{k-i,l-j}$, and then factoring them by $\frac{\binom{m_P}{i}\binom{m_Q}{k-i}\binom{n_P}{j}\binom{n_Q}{l-j}}{\binom{m_P+m_Q}{k}\binom{n_P+n_Q}{l}}$, resulting in a new sequence of terms. All the sequences, generated while computing $R_{k,l}$, are literally merged. Order of the product operations is important, because of the cross and dot products and, in general, this order can be captured in a structure of an expression tree. Memory overhead can be reduced by storing one syntax tree per surface, because all symbolic expressions corresponding to the control points of the surface share same product operations in the same order. The tree representation does not contain references to the symbols, but indexes into a shared symbol table. A term can be represented as a sequence of the control point's symbols together with a factorization. For example, $\{1, P_{2,0}\}$ represents $P_{2,0}$, and $\{-12, P_{0,0}, P_{4,1}, P_{3,2}\}$ represents $-12\,\langle P_{0,0} \times P_{4,1}, P_{3,2}\rangle$. In-order traversal is used to restore order of operations and operands. Finally, the computation of $\overline{K}$ in Equation (5) can be implemented via the operators of Equations (9)-(14).

Derivative of the surface $S^{\mathcal{Q}} = \sum_k \sum_l Q_{k,l} B^{m,n}_{k,l}(u,v)$ with respect to $P_{i,j}$ equals

$$\frac{\partial(S^{\mathcal{R}} \otimes S^{\mathcal{Q}})}{\partial P_{i,j}} = \frac{\partial S^{\mathcal{R}}}{\partial P_{i,j}} \otimes S^{\mathcal{Q}} + S^{\mathcal{R}} \otimes \frac{\partial S^{\mathcal{Q}}}{\partial P_{i,j}}, \tag{15}$$

$$\frac{\partial S^{\mathcal{Q}}}{\partial P_{i,j}} = \sum_k \sum_l \frac{\partial Q_{k,l}}{\partial P_{i,j}} B^{m_Q,n_Q}_{k,l}(u,v), \tag{16}$$

and can be implemented by scanning all the terms in the sequence $Q_{k,l}$ and replacing every term containing $P_{i,j}$ by a sequence of terms corresponding to the rule of derivatives of a product. Vanishing terms will be eliminated altogether from the sequence. For $P_{i,j} = \{x_{i,j}, y_{i,j}, z_{i,j}\} \in I\!R^3$, one is interested in the partial derivatives $(\frac{\partial S^{\mathcal{Q}}}{\partial x_{i,j}}, \frac{\partial S^{\mathcal{Q}}}{\partial y_{i,j}}, \frac{\partial S^{\mathcal{Q}}}{\partial z_{i,j}})$ with respect to the x, y, and z components of the vector $P_{i,j}$. Derivatives of $P_{i,j}$ with respect to itself are replaced by a special unit symbol I. For example, $\frac{\partial(P_{0,0} \cdot P_{1,0} \cdot P_{0,0} \cdot P_{0,1})}{\partial P_{0,0}}$ is replaced by $(I \cdot P_{1,0} \cdot P_{0,0} \cdot P_{0,1}) + (P_{0,0} \cdot P_{1,0} \cdot I \cdot P_{0,1})$. The symbol I equals one when the coefficients $P_{i,j} \in I\!R$, but I has a special meaning for the control points $P_{i,j} \in I\!R^3$. Symbolically, both cases can be represented using I, but when evaluating a symbolic expression, one must replace I with $(1,0,0)$ for x, (0,1,0) for y and (0,0,1) for z.

For bicubic Bezier surfaces, the upper limit on the number of terms in the symbolic expressions of the coefficients of e, f, g is clearly bound by 16^3. In practice, this number varies between 24 terms for the corner coefficient, $f_{0,0}$, to 862 terms for the interior coefficients. For bicubic Bezier surfaces, one is expected to experience severe memory and computational overheads in computing the symbolic expressions for $k_{i,j}(\mathcal{P})$ (bound by 16^6, and, in practice, numbering between 250 terms for a corner coefficient, $k_{0,0}$, to 4.5×10^6 terms for an interior coefficient) and in evaluating the values of these expressions, especially taking into account that 225 such expressions exist. Nonetheless, the structure of e, f, g in Equation (5) makes some critical optimizations possible. Applying the symbolic product operator, one can realize that any of the coefficients of the e, f, g factors can be represented as summation of terms of the form of:

$$\sum_k C_k \langle P_{i,j}, P_{q,r} \times P_{s,t} \rangle = \sum_k C_k \, det\left[P_{i,j}, P_{q,r}, P_{s,t}\right], \; (i,j) \neq (q,r) \neq (s,t)$$

For example, for a biquadratic surface $S^{\mathcal{P}}$, $e(\mathcal{P})$ and $f(\mathcal{P})$ (Equation (5)) have $degree(e) = (3,5)$ and $degree(f) = (4,4)$ respectively. Then, the symbolic expressions of the coefficient $e_{0,0}$ of e and control coefficient $f_{0,0}$ of f equal:

$$\begin{aligned} e_{0,0} &= 8det[P_{00}P_{01}P_{02}] - 8det[P_{00}P_{01}P_{10}] + 8det[P_{00}P_{02}P_{10}] - 8det[P_{01}P_{02}P_{10}], \qquad (17) \\ f_{0,0} &= -16det[P_{00}P_{01}P_{10}] + 16det[P_{00}P_{01}P_{11}] - 16det[P_{00}P_{10}P_{11}] + 16det[P_{01}P_{10}P_{11}]. \end{aligned}$$

Our implementation identifies and eliminates terms violating the $(i,j) \neq (q,r) \neq (s,t)$ constraint. An ordering on the symbols of the coefficients, based on the rules of elementary matrix transformation, is applied to combine all terms featuring the same triples of symbols into one term, corresponding to a mixed product operation. Symbol ordering is defined using a lexicographic ordering on names and as two digit number on indices. For example, $P_{0,1} < P_{1,0}$ and $P_{1,0} < Q_{0,1}$. The term $C_2\ det\,[P_{0,0}, P_{2,0}, P_{0,2}]$ can be combined with the term $C_8\ det\,[P_{2,0}, P_{0,0}, P_{0,2}]$ to produce a new term $(C_8 - C_2)\ det\,[P_{0,0}, P_{0,2}, P_{2,0}]$. Then, an expression tree containing the two product operations is replaced with a single mixed product operation.

From the point of view of pure symbolic computation, one can also apply the following optimizations:

$$B^{m,n}_{i,j}(u,v) = \binom{m}{i}\binom{n}{j}(1-u)^{m-i}u^i(1-v)^{n-j}v^j = B^{m,n}_{m-i,n-j}(1-u,1-v).$$

One can immediately identify that there is a transitive symmetry relation between indices of control points (see Figure 1):

$P_{i,j}$ is symmetric around the horizontal axis to $P_{m-i,j}$, $i = 0..m,\ j = 0..n$,

$P_{i,j}$ is symmetric around the vertical axis to $P_{i,n-j}$, $i = 0..m,\ j = 0..n$,

$P_{i,j}$ is symmetric around the major diagonal axis to $P_{j,i}$, $i = 0..m,\ j = 0..n$,

$P_{i,j}$ is symmetric around the minor diagonal axis to $P_{n-j,m-i}$, $i = 0..m,\ j = 0..n$.

This symmetry reduces the storage complexity of the control points of a Bezier surfaces by approximately a factor of eight. The gray triangle on Figure 1 marks the required storage region. For example, recalling that $degree(e) = (3,5)$ for a biquadratic surface, we can obtain the symbolic expression for the control point $e_{0,5}$ from the symbolic expression of $e_{0,0}$ (compare with Equation (17)):

$$e_{0,5} = -8det[P_{02}P_{01}P_{00}] + 8det[P_{02}P_{01}P_{12}] - 8det[P_{02}P_{00}P_{12}] + 8det[P_{01}P_{00}P_{12}]$$

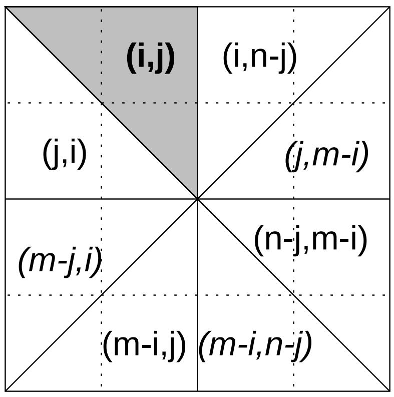

Figure 1: Bold font denotes source indices, regular font denotes symmetric indices and italic font denotes transitively symmetric indices.

$$= \quad 8det[P_{00}P_{01}P_{02}] - 8det[P_{00}P_{01}P_{12}] + 8det[P_{00}P_{02}P_{12}] - 8det[P_{01}P_{02}P_{12}],$$

via a symmetry around the vertical axis.

Using the structure of e, f and g, one can improve the evaluation of $k_{i,j}(\mathcal{P})$ as well.

$$\overline{K}(\mathcal{P}) = e(\mathcal{P})g(\mathcal{P}) - f(\mathcal{P})^2, \tag{18}$$

$$\frac{\partial \overline{K}}{\partial P_{i,j}} = \frac{\partial e}{\partial P_{i,j}} g + e\frac{\partial g}{\partial P_{i,j}} - 2f\frac{\partial f}{\partial P_{i,j}} \tag{19}$$

The coefficients of e, f, g can be precomputed and reused in the computation of $k_{i,j}$ and $\frac{\partial k_{i,j}}{\partial P_{i,j}}$ in Equations (18)-(19), using numerical equivalent of a product Equation (14). Further, one can easily notice that terms in e, f, g are featuring the same mixed products. For example, the first terms of $e_{0,0}$ and $f_{0,0}$ (Equation (17)) contain $det[P_{0,0}P_{0,1}P_{1,0}]$. For a bicubic surface, the maximal number of instances of a mixed product term is achieved for $det[P_{0,1}, P_{1,0}, P_{1,1}]$ and its seven other symmetric terms, that appears 53×8 times in e, f, g. Table of mixed products $det\left[P_{i,j}P_{q,r}P_{s,t}\right]$, $(i,j) < (q,r) < (s,t)$ can be precomputed, and evaluation of the value of one term of the control points of e, f, g and their gradients is as simple as looking up into the table. The size of this table is governed by the combinatorial formula for the number of ways one can select three numbers $i < j < k, \quad i,j,k \in 1, ..., M$ and is equal to $\binom{M}{3} = \frac{M(M-1)(M-2)}{6}$ (560 for bicubic surface, $M = 16$).

A special kind of a developability constraints, *developability with the conjugate parametric net*, might be useful in practice and can be handled more efficiently then the general case described above. The conjugate parametric net condition is identified by:

$$f(u,v) = \left\langle \frac{\partial^2 S(u,v)}{\partial u \partial v}, N(u,v) \right\rangle = 0, \ \forall u, \ v. \tag{20}$$

It follows, that the developability problem in Equation (4) is reduced to:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!\!R^{3M} : f_{i,j}(\mathcal{P}) = 0, \ i = 0,..,3m-3, \ j = 0,..,3n-3\} \\ \text{and subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!\!R^{3M} : e_{i,j}(\mathcal{P}) = 0, \ i = 0,..,3m-4, \ j = 0,..,3n-2\} \\ & \text{when U isoparametric lines are planar, or subject to:} \\ & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!\!R^{3M} : g_{i,j}(\mathcal{P}) = 0, \ i = 0,..,3m-2, \ j = 0,..,3n-4\} \\ & \text{when V isoparametric lines are planar.} \end{aligned} \tag{21}$$

4 The Objective Function

When a surface is altered during a constraints satisfaction process, it is desired to preserve the original design as much as possible. Naturally, this requirement can be expressed by an objective function, with a minimum that is achieved when $S^{\mathcal{P}} = S^0$. In our implementation, we used the following L_2-norm based functional:

$$\sqrt{\int_0^1 \int_0^1 ||S^{\mathcal{P}} - S^0||_2^2 du dv}. \tag{22}$$

The minimum of this functional is the same as the minimum of the following objective function:

$$F(\mathcal{P}) = \int_0^1 \int_0^1 ||S^{\mathcal{P}} - S^0||_2^2 du dv. \tag{23}$$

In practice, one can use a simpler alternative, for Bezier surfaces,

$$F(\mathcal{P}) = \sum_i \sum_j ||P_{i,j} - P^0_{i,j}||_2^2, \tag{24}$$

which is an approximation to Equation (23), as follows:

$$F(\mathcal{P}) \ = \ \int_0^1 \int_0^1 ||S^{\mathcal{P}} - S^0||_2^2 du dv$$

$$
\begin{aligned}
&= \int_0^1\int_0^1 ||\sum_i\sum_j (P_{i,j} - P^0_{i,j})B_{i,j}(u,v)||_2^2 du dv \\
&\leq \int_0^1\int_0^1 \sum_i\sum_j ||(P_{i,j} - P^0_{i,j})||_2^2 B_{i,j}(u,v)^2 du dv \\
&\leq \int_0^1\int_0^1 \sum_i\sum_j ||(P_{i,j} - P^0_{i,j})||_2^2 du dv \\
&= \sum_i\sum_j ||P_{i,j} - P^0_{i,j}||_2^2.
\end{aligned}
$$

Both objective functions, Equations (23) and (24), are quadratic with respect to $\mathcal{P}$, and thus have a global minimum, satisfying $S^{\mathcal{P}} = S^0$.

5 Numerical solution

We have formulated the surface approximation problem with a quadratic objective function and generally nonlinear constraints:

$$
\begin{aligned}
\text{minimize:}\quad & F(\mathcal{P}) = \sum_i\sum_j ||P_{ij} - P^0_{ij}||_2^2, & (25)\\
\text{subject to:}\quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : k_{i,j}(\mathcal{P}) = 0,\ i = 0,..,6m-4,\ j = 0,..,6n-4\}, \\
\text{or:}\quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : k_{i,j}(\mathcal{P}) \geq 0,\ i = 0,..,6m-4,\ j = 0,..,6n-4\}.
\end{aligned}
$$

Now, we seek an NLP method satisfying the following criteria. It should be robust, probably at the expense of computation time. It should deal with a large number of equality and/or inequality constraints, typically much greater than number of variables ($(6m-3)(6n-3) >> 3(m+1)(n+1)$). Due to properties of Bezier surfaces, the number of variables is medium (up to several hundreds) and the Jacobian matrix (composed of gradients) of constraints is not sparse.

Current SQP (sequential quadratic programming) methods are considered the most efficient methods for solving NLP's of small and medium size [19]. The SQP method tries to obtain the solution of the NLP problem by iteratively computing a sequence of candidate solutions $\{\mathcal{P}^l\}_l$, where l is the iteration number. When an SQP solver is at iteration l with solution $\mathcal{P}^l$, it should compute a correction direction d^l from solution point $\mathcal{P}^l$ such that all constraints are satisfied, or feasible, at a new point. That is,

$$\mathcal{P}^{l+1} = \mathcal{P}^l + \sigma^l d^l,\quad \mathcal{P}^{l+1} \in \Omega,\quad \sigma^l \in (0,1], \tag{26}$$

and the objective function is minimized. Usually, these conditions are contradictory, and an SQP method should select the correct tradeoff between temporary violation of the constraints and fast minimization. σ^l in Equation (26) is such a tradeoff factor guaranteeing the condition $\mathcal{P}^{l+1} \in \Omega$. The computation of d^l is carried out as a solution of a quadratic programming subproblem of the original NLP problem:

$$\begin{aligned} \text{minimize:} \quad & \nabla F(\mathcal{P}^l)^T d^l + \frac{1}{2} {d^l}^T B_l d^l, \\ \text{subject to:} \quad & \nabla h_i(\mathcal{P}^l)^T d^l + h_i(\mathcal{P}^l) = 0, \ \forall i, \\ \text{or subject to:} \quad & \nabla g_i(\mathcal{P}^l)^T d^l + g_i(\mathcal{P}^l) \geq 0, \ \forall i, \end{aligned} \tag{27}$$

where B_l is an approximation to the combined second order derivatives' information of the constraints $h(\mathcal{P}^l)$, $g(\mathcal{P}^l)$ and the objective function of the surface's modeling problem, $F(\mathcal{P})$. B_{l+1} is obtained from B_l using only first order derivatives, i.e. gradients. We employ a technique called BFGS (Broyden, Fletcher, Goldfarb, and Shanno) update [19] to maintain B_l. For example, for a bicubic Bezier surface one gets:

$$\begin{aligned} \text{minimize:} \quad & 2\sum_{i=0}^{3}\sum_{j=0}^{3}(P_{i,j}^l - P_{i,j}^0)d_{i,j}^l + \frac{1}{2} {d^l}^T B_l d^l, \\ \text{subject to:} \quad & \nabla k_{i,j}(\mathcal{P}^l)^T d^l + k_{i,j}(\mathcal{P}^l) = 0, \ (i,j) \in \{0,..,14 \times 0,..,14\}, \\ \text{or subject to:} \quad & \nabla k_{i,j}(\mathcal{P}^l)^T d^l + k_{i,j}(\mathcal{P}^l) \geq 0, \ (i,j) \in A_l \subset \{0,..,14 \times 0,..,14\}, \end{aligned} \tag{28}$$

where the set of constraints used at the lth iteration is denoted the active set A_l and is typically less then $15^2 = 225$ in the inequality NLP problem. Many of the inequality constraints are feasible most of the time throughout the solution process, and are infeasible in a small number of iterations. When the inequality constraint is feasible, it has no influence on the minimization, and thus, can be left suspended, until it becomes almost infeasible. Nevertheless, one should care not to change the active set too often. In order for d^l to be well defined, B_l is required to be positive definite on the subspace restricted to the active constraints and thus, a unique global minimum is obtained.

A new technique for inconsistent QP problems in SQP method was recently developed in [20]. Our present implementation uses a solver that is based on this method and is described briefly below. This method considers only those constraints which are near active or active,

$$A_l \;=\; \{i : g_i(\mathcal{P}^l) \leq \delta^l \in I\!R_+\}, \tag{29}$$

$\delta^l \to 0$ implies that $\mathcal{P}^l$ is regarded as local constrained minimum of the NLP problem. $\sigma^l > 0$ is chosen by a decrease test for a so called L_1(-norm) penalty function, for which the state is sought when the objective function is at the minimum and no constraint violations are observed:

$$\Phi(P^l, \beta, \gamma) \;=\; F(\mathcal{P}) + \sum_i \beta_i \;\max(-g_i(\mathcal{P}^l), 0) + \sum_i \gamma_i |h_i(\mathcal{P}^l)|,$$

where β and γ are weights given to the violation of constraints in exact (distance) sense.

To make the active constraints consistent, the QP subproblem is also extended by introducing slack variables with weights for every constraint. Careful handling that drives the values of these variables to zero provides sufficiently fast, and robust realization of the SQP method. When the inconsistencies are small, *interactive speed* of surface manipulation can be achieved. For example, preservation of the convexity of the bicubic Bezier surface can be carried out during interactive manipulation of the control points. But when the infeasibility is substantial and/or the problem is highly inconsistent, the minimization attempts may fail or at least take enormous number of iterations, costing minutes of computation time. For example, the developability problem defined in Equation (8) results in an over constrained quadratic subproblem in Equation (28): there are 225 linear constraints and 48 variables. One should carefully choose an NLP method that overcomes this difficulty. When such an NLP method is not available, one can transform the problem in Equation (8) into:

$$\begin{aligned} \text{minimize:} \quad & F(\mathcal{P}) \\ \text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : T_i(\overline{K}(\mathcal{P})) = 0,\; i = 1..L,\; L \leq 3M\}, \end{aligned} \tag{30}$$

where T_i is a *constraints transformation function* having a C^2 continuity and satisfying the following condition:

$$T_i(\overline{K}(\mathcal{P})) = 0 \;\;\Leftrightarrow\;\; \overline{K}(\mathcal{P}) = 0.$$

These conditions ensure that the constraints of the original problem are satisfied at the solution and the QP subproblem is not over constrained. A simple example of such a transformation function is:

$$T(\overline{K}(\mathcal{P})) = \frac{\overline{K}(\mathcal{P})^2}{2}, \tag{31}$$

$$\nabla T(\mathcal{P}) = \nabla \overline{K}(\mathcal{P})^T \overline{K}(\mathcal{P}).$$

This function has an obvious deficiency, its gradient and Hessian are zero when the constraints are satisfied, while the original gradient and Hessian may be different, resulting in slow convergence near the solution. In practice, one can rely on the objective function to achieve an acceptable convergence rate. Advantages of function (31) are the small computational overhead over the computation of $\overline{K}(\mathcal{P})$, the faster solver processing due to the reduced dimensions of the matrices used in solving the QP subproblem in Equation (27) and no impact on the conditional number of B_l in Equation (27). One can explore other practical schema, like transforming only the interior constraints of the problem.

One should distinguish between the interactive mode and the batch mode of solving the convex and developable surface constraint problems. In an interactive mode, the initial surface S^0 *satisfies* either the convexity or the developability constraints. The desired surface $S^{\mathcal{P}}$ is obtained via a *small* change in the position of a single control point. Such a modification can be formulated with the additional three linear constraints in the problem Equation (25):

$$\begin{aligned}
\text{minimize:} \quad & F(\mathcal{P}) = \sum_i \sum_j ||P_{ij} - P^0_{ij}||^2_2, \\
\text{subject to:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : k_{i,j}(\mathcal{P}) = 0,\ i = 0,..,6m-4,\ j = 0,..,6n-4\}, \\
\text{or:} \quad & \mathcal{P} \in \Omega = \{\mathcal{P} \in I\!R^{3M} : k_{i,j}(\mathcal{P}) \geq 0,\ i = 0,..,6m-4,\ j = 0,..,6n-4\}, \\
\text{and subject to:} \quad & P_{i,j} - P^0_{i,j} = 0, \text{ for the certain control point } P_{i,j}.
\end{aligned}$$

In contrast, in batch mode, the initial surface does not satisfy the constraints, in general. However, at the initial phase of the optimization process, the solver of the NLP problem tries to obtain a surface that satisfies the constraints, while it needs not to minimize the objective function $F(\mathcal{P})$. Alternatively, one can provide the solver with an initial surface $\overline{S}^0$ that differs from the surface S^0 that is used in the objective

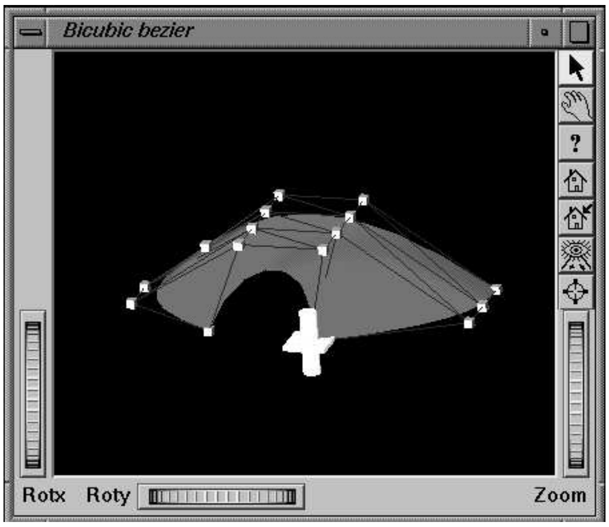

Figure 2: A snapshot of the modeling application of Bezier surfaces.

function. The surface $\overline{S}^0$ should satisfy the constraints of the prescribed problem and should not be too far from the surface S^0. For example, one can obtain such a surface by constructing a best fit planar surface based on the control points of S^0.

6 Results

An interactive tool for modeling Bezier surfaces that implements the techniques described in this work has been developed using the Open Inventor [21] toolkit (see Figure 2). The modeler allows the user to interactively manipulate control points of the Bezier surface while preserving various types of constraints.

All forthcoming figures are snapshots of intermediate surfaces during the optimization process until convergence is reached. See, for example, Figure 3. In the batch mode, one is allowed to modify an arbitrary number of control points. At any moment, one can select a desired constraint from a prescribed menu and instantly observe the changes in the shape of the surface on the screen, until the convergence is complete. Examples of modeling a bicubic Bezier surface using a variety of constraints is shown in Figures 3-7. Global second order geometric constraints are most effectively used in a conjunction with other constraints, because by themselves they leave too many degrees of freedom unspecified. For example, the parametric nets in the examples of Figures 4 and 7 are not necessary orthogonal. To properly simulate

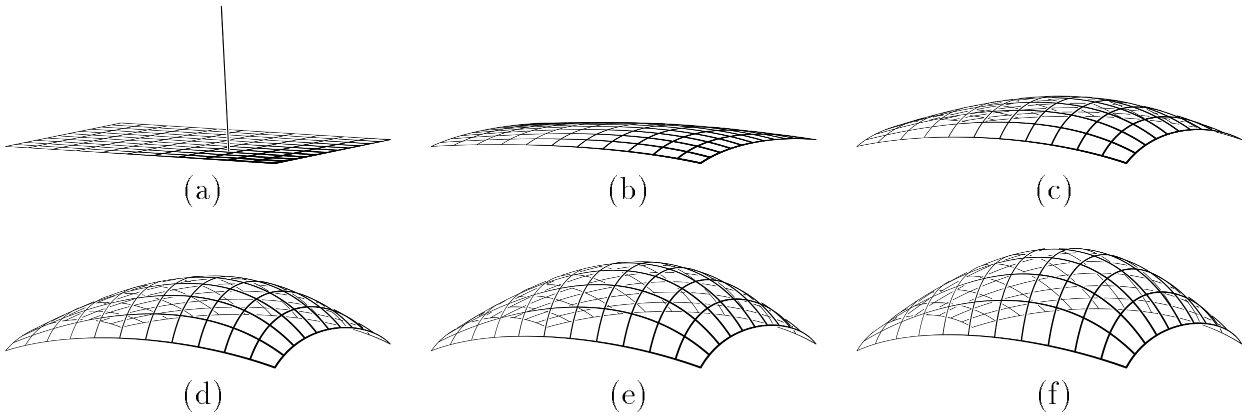

Figure 3: A planar bicubic Bezier surface is altered by dragging one of the four interior control points of the surface while preserving convexity and locations of the corners. The sequence (a)-(f) demonstrates sample frames of the manipulation, computed in an interactive session, all in few seconds on a 133 MHz Pentium machine.

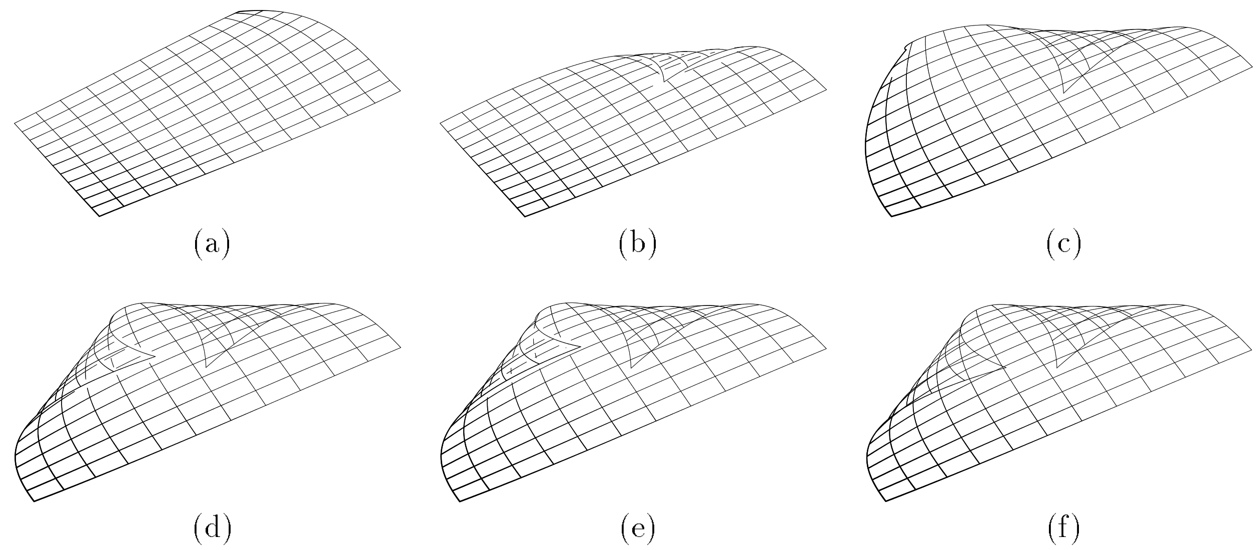

Figure 4: A planar surface is folded into a flat shape resembling an envelope by interactively dragging two corner control points one after another while preserving developability and the locations of the three other corners. (f) is a flat surface. Note, we do not preserve the first fundamental form in this example. The sequence is computed in an interactive session, all in few seconds on a 133 MHz Pentium machine.

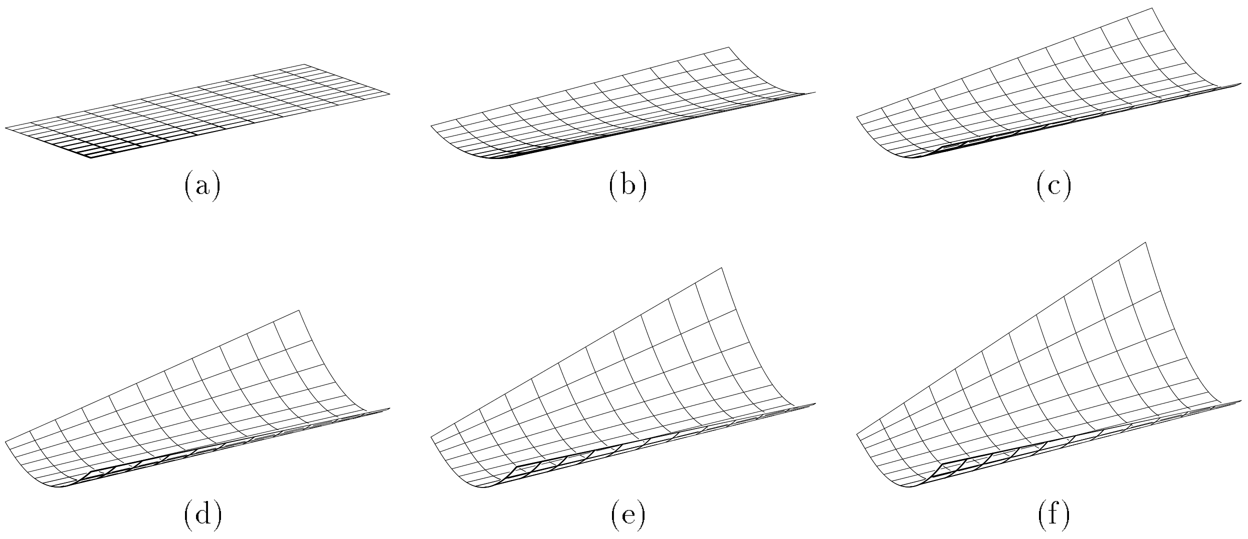
Figure 5: A planar surface is altered by dragging the far corner control point of the bicubic Bezier while preserving developability, conjugate parametric net, planarity along the V parametric direction and the locations of the three other corners. The sequence is computed in an interactive session, all in few seconds on a 133 MHz Pentium machine.

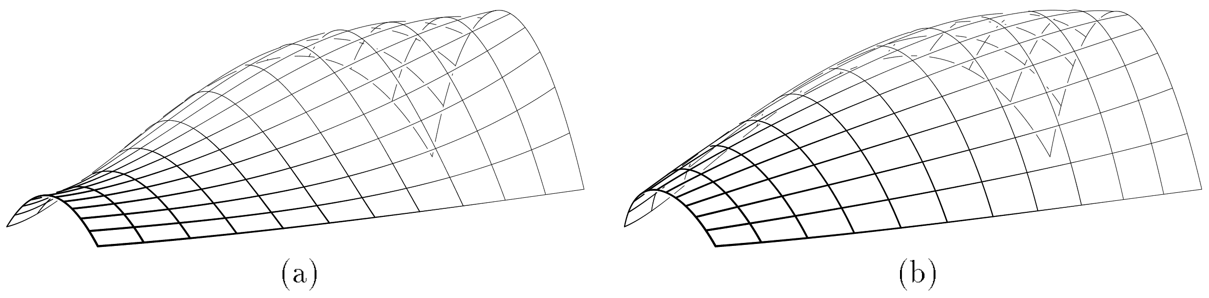
Figure 6: The designer is interested in a hull shape resembling the one in (a), which obviously violates convexity restrictions. In (b), the shape of the hull is globally convex and closely resembles the original shape. During the optimization process, the locations of the corner control points are preserved. The optimization process took approximately 4 seconds on a 133 MHz Pentium machine.

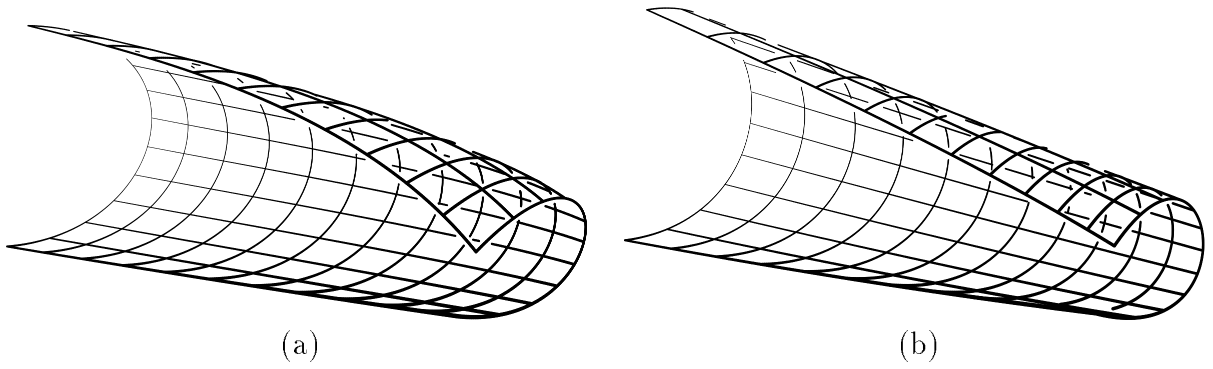
Figure 7: The plane is folded as in (a), which obviously violates developability restrictions. In (b), the shape of the folded plane is globally developable. One can see that the parametric net is not necessary orthogonal throughout the optimization process, since we do not preserve the first fundamental form. During the optimization process the locations of the corner control points are preserved. The optimization process took approximately 9 seconds on a 133 MHz Pentium machine.

the folding behavior of a sheet of paper, one is required to put a global orthogonality constraint upon the surface. Restricting the surface to a arclength preserving orthogonal parametric net can be easily incorporated into the presented framework, resulting in an isometric mapping.

Following [13], we consider the use of our tool with an 'unreasonable' initial surface. Figure 8 shows an initial surface with a loop in the control mesh as well as the final solution surfaces without a loop. Two initial guesses were employed. The first is identical to the initial surface and the other is planar without a loop in the control mesh. The techniques employed in this work allows one to solve the problem from both initial guesses, while in [13] only 'reasonable' initial guess (without a loop) succeeded. Nevetheless, solving the problem with the first initial with the loop guess took considerably more time.

7 Conclusions

Our experience shows a high level of utility of the global second order constraints in both batch and interactive shape modeling. To make the incorporation of the global second order constraints practical, further research is necessary in three directions. The first direction has the goal of extending the techniques discussed in this paper to support B-spline surfaces. The second direction would add other global constraints on the basis of both the first and second fundamental forms to those already employed in the paper. Exam-

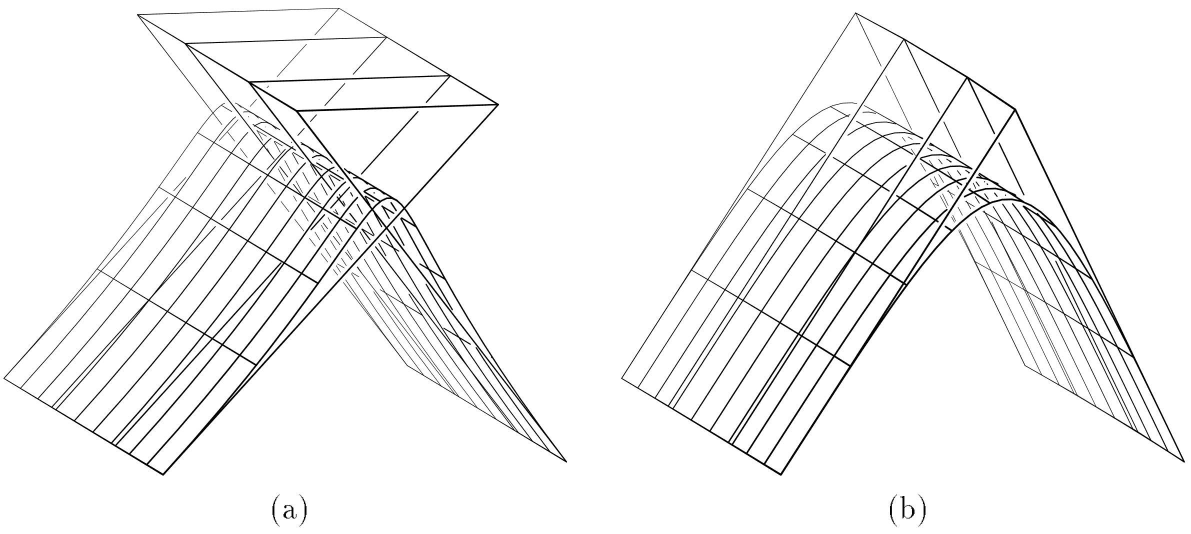

Figure 8: The control mesh of the bicubic Bezier surface in (a) has a loop. Solution of the strict convexity (with tolerance $5 * 10^{-4}$) problem is shown in (b). One can see that the solution eliminates the loop in the control mesh, while preserving the general shape of the surface. During the optimization process, the locations of the four corner control points are preserved. The optimization process took approximately 10 seconds on a 133 MHz Pentium machine when the initial guess was s planar surface without a loop in the control mesh, and took approximately 30minutes, when the initial guess was identical to the surface in (a).

ples include modeling minimal surfaces and surfaces of constant curvature, and applying global constraints to subregions of the parametric domain (with bifocal and trifocal lenses as a possible application). To improve the computation speed in interactive applications one can extend the conditions for the convexity of the tensor-product Bezier and B-spline surfaces as described in [14]. One important application of the techniques presented in this work may be fairing of the surface shape using the theorem presented in [3], that a surface that satisfies $f(u,v) = F(u,v)(k_{min}(u,v) + k_{max}(u,v)),\ \ \forall u,v$ is smooth in the sense that $\int(k^2_{min}(u,v) + k^2_{max}(u,v))dudv \rightarrow \min$ (where $F(u,v)$ and $f(u,v)$ are the symmetric components of the first and second fundamental forms accordingly). The third direction invites research in numerical analysis to provide better techniques that are suitable for solving the NLP problems with the large number of constraints expressed as multivariate polynomials. For example, convexity problems show stable and robust behavior in our implementation. Developability problems behave almost as robustly, but as was mentioned in the discussion on the constraints' transformation function, in Section 5, SQP based optimization methods are likely to experience difficulties when attempting to solve the developability problems and may show slow convergence.

References

[1] G. Farin. Curves and Surfaces for Computer Aided Geometric Design. Academic Press, San Diego, California, second edition, 1990.

[2] A. Witkin, K. Fleischer, A. Barr. Energy Constraints On Parameterized Models. Computer Graphics, Vol. 21, No. 4, July 1987, pp. 225-232.

[3] G.Farin, H. Hagen. A Local Twist Estimator. Hagen, Hans, editor, Topics In Surface Modeling, 1992, pp. 79-84.

[4] E. Anderson, R. Anderson, M.Boman, T.Elmroth, B.E.J. Dahlberg, B. Johanson. Automatic construction of surfaces with prescribed shape. Computer Aided Design, No. 20, 1988, pp. 317-324.

[5] W. Welch, A. Witkin. Variational Surface Modeling. Computer Graphics, Vol. 26 No.2, July 1992, pp. 157-166.

[6] B.M. Fowler. Geometric Manipulation of Tensor Product Surfaces. Computer Graphics, 1992 Symposium on Interactive 3D Graphics, Vol. 25, No.2, March 1992, pp. 101-108.

[7] B.M. Fowler. Constraint Based Curve Manipulation. IEEE Computer Graphics and Applications, Vol. 13, No.5, September 1993, pp. 43-49.

[8] G. Celinker, W. Welch. Linear Constraints for Deformable B-Spline Surfaces. Computer Graphics, 1992 Symposium on Interactive 3D Graphics, Vol. 25, No.2, March 1992, pp. 165-170.

[9] M. Gleicher. Integrating Constraints and Direct Manipulation Computer Graphics, 1992 Symposium on Interactive 3D Graphics, Vol. 25, No.2, March 1992, pp. 171-174.

[10] C. Loop, T. DeRose. Generalized B-spline Surfaces of Arbitrary Topology. Computer Graphics, SIGGRAPH '90 Conference Proceedings, Vol. 24, No. 4, 1986, pp. 151-160.

[11] G. Celinker, D Gossard. Deformable Curve and Surface Finite-Elements for Free-form Shape Design. Computer Graphics, SIGGRAPH '91 Conference Proceedings, Vol. 25, No. 4, 1991, pp. 121-129.

[12] M. Kallay. Constrained Optimization in Surface Design. 1992.

[13] D. Ferguson, P. Frank, A. Jones. Surface shape control using constrained optimization on the B-spline representation, Computer Aided Geometric Design, No. 5, 1988, pp. 87-193.

[14] M. Floater. A weak condition for the convexity of tensor-product Bezier and B-spline surfaces. Advances in Computational Mathematics, No 2, 1994, pp. 67-80.

[15] H. Burchard, J. Ayers, W.Frey, N. Sapidis. Approximation with Aesthetic Constraints. Sapidis, Nickolas S., editor, Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design, 1994, pp. 3-28.

[16] A. Jones. Curvature Integration through Constrained Optimization. Sapidis, Nickolas S., editor, Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design, 1994, pp. 29-44.

[17] A. Ginnis, P. Kaklis, N.Sapidis. Polynomial Splines of Nonuniform Degree: Controlling Convexity and Fairness. Sapidis, Nickolas S., editor, Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design, 1994, pp. 253-276.

[18] G. Elber. Free form surface analysis using a hybrid of symbolic and numeric computations. PhD thesis, University of Utah, 1992.

[19] R. Fletcher. Practical methods of optimization, 2nd ed., Wiley, Chicester - New York, 1987.

[20] P. Spellucci. A new technique for inconsistent QP-problems in the SQP method. Technical University of Darmstadt, Department of Mathematics, preprint 1561, Darmstadt 1993.
ftp://ftp.mathematik.tu-darmstadt.de/pub/department/software/opti/newsqpsh.ps.gz

[21] Silicon Graphics, Inc. Open Inventor C++ Reference Manual, Addison-Wesley, 1994.