Dynamic NURBS Swung Surfaces for Physics-Based Shape Design

Hong Qin and **Demetri Terzopoulos**[1]
Department of Computer Science, University of Toronto,
10 King's College Road, Toronto, Ontario, M5S 1A4

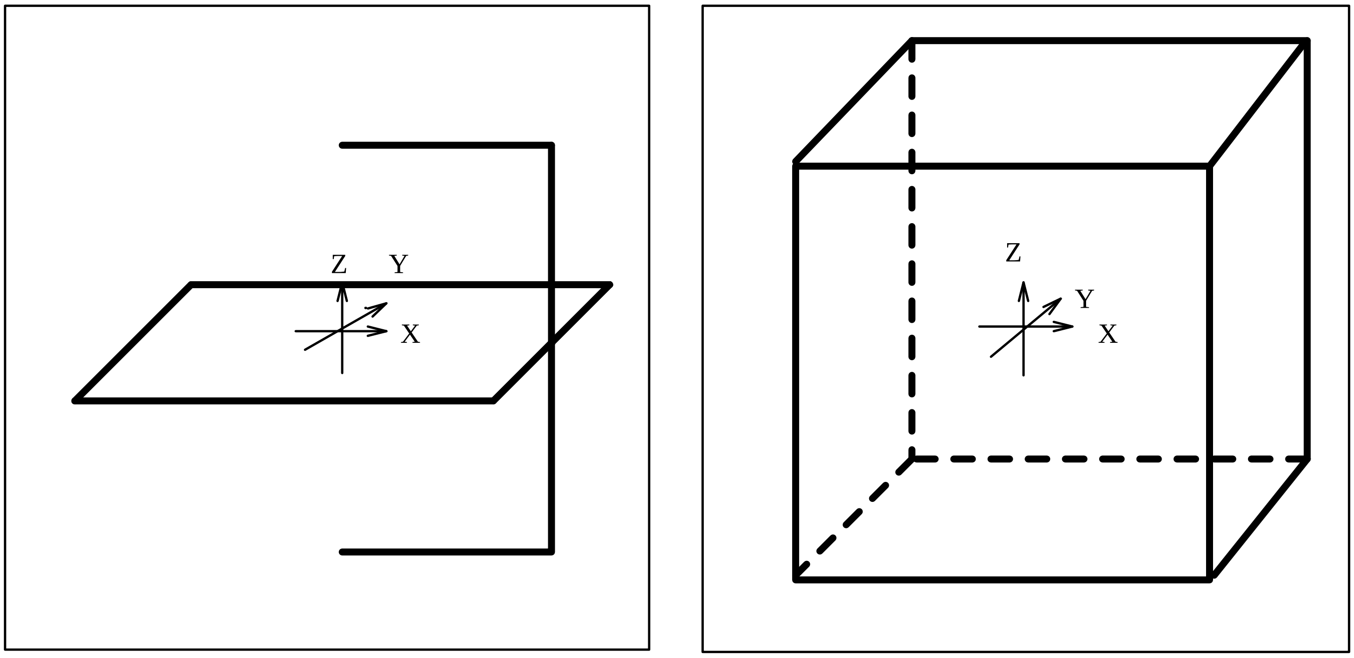
Published in *Computer-Aided Design*, **27**(2):111–127, 1995.

Abstract

We develop a dynamic, free-form surface model which is useful for representing a broad class of objects with symmetries and topological variability. The new model is based upon swung NURBS surfaces, and it inherits their desirable cross-sectional design properties. It melds these geometric features with the demonstrated conveniences of surface design within a physics-based framework. We demonstrate several applications of dynamic NURBS swung surfaces, including interactive sculpting through the imposition of forces and the adjustment of physical parameters such as mass, damping, and elasticity. Additional applications include surface design with geometric and physical constraints, by rounding solids, and through the fitting of unstructured data. We derive the equations of motion for the dynamic NURBS swung surface model using Lagrangian mechanics of an elastic surface and the finite element method. We also show that these surfaces are a special case of D-NURBS surfaces, a recently proposed physics-based generalization of standard geometric NURBS. Our free-form, rational model not only provides a systematic and unified approach for a variety of CAGD problems such as constraint-based optimization, variational design, automatic weight selection, shape approximation, etc., but it also supports interactive sculpting using physics-based manipulation tools.

Keywords: CAGD, NURBS, NURBS Swinging, Deformable Models, Dynamics, Constraints, Finite Elements, Solid Rounding, Surface Blending, Scattered Data Fitting, Interactive Sculpting.

[1]Fellow, Canadian Institute for Advanced Research
E-mail addresses: qin@cs.toronto.edu; dt@cs.toronto.edu



(a) (b)

Figure 1: Construction of a cubical NURBS swung surface. (a) NURBS profile curve on x-z plane, NURBS trajectory curve on x-y plane. (b) Cube surface wireframe.

1 Introduction

Among the surface representation schemes in CAGD, non-uniform rational B-splines (NURBS) have become an industry standard [1]. One of their most significant advantages is that they are a unified representation of both complex free-form shapes and standard analytic shapes [1, 2, 3]. NURBS objects are designed by adjusting control points and weights that are associated with NURBS surface patches.

Many objects of interest, especially manufactured objects, exhibit symmetries. Often it is convenient to model symmetric objects through cross-sectional design by specifying profile curves [4]. Woodward [5] introduced the swinging operator by extending the spherical cross-product with a scaling factor, and applied it to generate surfaces with B-spline profile curves (see also [6]). Piegl [1] carried the swinging idea over to NURBS curves. He proposed NURBS swung surfaces, a special type of NURBS surfaces formed by swinging one planar NURBS profile curve along a second NURBS trajectory curve. The two generator curves may be smooth, or they may have discontinuities. For example, Fig. 1 illustrates the design of a cubical NURBS swung surface from two NURBS profile curves.

The NURBS swung surface retains a considerable breadth of geometric coverage. It can represent common geometric primitives such as spheres, tori, cubes, quadrics, surfaces of revolution, etc. Fig. 4 illustrates four NURBS swung surfaces with distinct topological structures. The NURBS swung surface is efficient compared to a general NURBS surface, inasmuch as it can represent a broad class of shapes with essentially as few degrees of freedom as it takes to specify the two generator curves. Several geometric shape design systems, including the recent one in [7], include some form of swinging (or sweeping) among their repertoire of techniques.

In this paper, we develop a physics-based generalization of the geometric NURBS swung surface. We refer to our new models as *dynamic NURBS swung surfaces*.

1.1 Motivation

Although planar curve design is much easier than general surface design, in many real-world circumstances it is hard to achieve satisfactory results quickly. Normally the designer obtains (quasi-global) control over the free-form NURBS swung surface by adjusting the control points and weights of the two NURBS curves. Although there are a variety of algorithms and interaction devices, this indirect design process, which is characteristic of geometric design with NURBS and other free-form surface representations in general, can be clumsy and time consuming. Moreover, relevant design requirements are usually shape oriented and not control point and weight oriented. Because of the geometric "redundancy" of NURBS [1], traditional geometric shape refinement remains *ad hoc* and ambiguous. In particular, making meaningful use of weights is one of the most important areas of current NURBS research. Furthermore, typical design requirements may be posed in both quantitative and qualitative terms. It can be very frustrating with indirect design to, for example, shape a "fair" surface that approximates unstructured 3D data. Unstructured shape constraints are especially problematic for cross-sectional design.

Physics-based modeling techniques provide a means of overcoming these difficulties. It is possible to construct free-form surfaces with natural dynamic behavior governed by physical laws [8, 9, 10]. Terzopoulos and Fleischer [11] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [12] developed an interesting prototype system for interactive free-form design based on this idea and the finite-element optimization of energy functionals. Bloor and Wilson [13] used similar energies optimized through numerical methods and they employed B-splines for this purpose. Subsequently, Welch and Witkin [14] extended the approach to trimmed hierarchical B-splines. Thingvold and Cohen [15] proposed a hybrid deformable B-spline whose control points are mass points connected by elastic springs and hinges. Moreton and Sequin [16] interpolated a minimum energy curve network with quintic Bezier patches by minimizing the variation of curvature. Halstead et al. [17] implemented smooth interpolation with Catmull-Clark surfaces using a thin-plate energy functional [18].

The dynamic NURBS swung surfaces proposed in the present paper were inspired by D-NURBS, our physics-based generalization of standard geometric NURBS [19]. Like D-NURBS, dynamic NURBS swung surfaces have continuous mass and damping distributions, as well as a deformation energy. With proper choice of physical parameters, they behave like physical surfaces. This allows a designer to interactively sculpt and directly manipulate shapes in a natural and predictable way using a variety of force-based tools. The surfaces in Fig. 4 were interactively sculpted in this fashion from the prototype shapes indicated in the caption. An important advantage of a dynamic model built upon the standard NURBS geometric foundation is that shape design may proceed interactively or automatically at the physical level, while existing geometric toolkits are concurrently applicable at the geometric level.

In the physics-based approach, functional design requirements can be readily implemented as surface deformation (fairness) energies and as geometric constraints on the surface. As a dynamic surface reaches equilibrium, it acts as a nonlinear shape optimizer since it minimizes its energy subject to the imposed constraints. However, shape design is generally a time-varying process—a designer is often interested not only in the final equilibrium shape but also in the intermediate shape variation due to parameter changes. Time is fundamental to the physics-based formulation. Physics-based models are governed by dynamic differential equations which, when integrated numerically through time, can continuously evolve control points and weights in response to applied forces to produce physically meaningful and intuitively predictable shape variation. A key advantage of our physics-based framework is that it permits appropriate weight values to be determined automatically in accordance with various physical parameters and geometric requirements.

We use Lagrangian mechanics to formulate the equations of motion of dynamic NURBS swung surfaces, and finite element analysis to reduce these equations to efficient algorithms that can be simulated numerically using standard techniques. One of the challenges in this effort is coping with the nonlinear dynamic formulation stemming from the underlying swung NURBS geometry. The NURBS swung surface is inherently nonlinear with respect to its degrees of freedom, even if both NURBS generator curves are reduced to simple B-splines by fixing their weights to unity. As a consequence of the nonlinearity, the

[1] A particular shape can often be represented nonuniquely using NURBS, with different values of knots, control points, and weights.

mass, damping, and stiffness matrices in the dynamic formulation must be recomputed at each simulation time step.

1.2 Overview

Section 2 defines kinematic versions of the basic NURBS curve generators in the swung surface and gives the kinematic equations. In Section 3, we formulate the dynamic NURBS swung surface and derive their equations of motion. We discuss the numerical simulation of these equations in Section 4. Section 5 discusses the use of forces and constraints for physics-based design. Section 6 presents applications of dynamic NURBS swung surfaces to interactive sculpting, scattered data fitting, and rounding/blending and discusses the results. In Section 7 we show that dynamic NURBS swung surfaces are a special case of D-NURBS surfaces [19] that have been subjected to a dimensionality-reducing nonlinear constraint. Section 8 concludes the paper.

2 Kinematic NURBS Curve

A kinematic NURBS curve extends the geometric NURBS definition by explicitly incorporating time. The kinematic curve is a function of both the parametric variable u and time t:

$$\mathbf{c}(u,t) = \frac{\sum_{i=0}^{n} \mathbf{p}_i(t) w_i(t) B_{i,k}(u)}{\sum_{i=0}^{n} w_i(t) B_{i,k}(u)}. \tag{1}$$

where the $B_{i,k}(u)$ are the usual recursively defined piecewise rational basis functions [20, 21], $\mathbf{p}_i(t)$ are the $n+1$ control points, and $w_i(t)$ are associated non-negative weights. Assuming basis functions of degree $k-1$, the curve has $n+k+1$ knots t_i in non-decreasing sequence: $t_0 \le t_1 \le \ldots \le t_{n+k}$. In many applications, the end knots are repeated with multiplicity k in order to interpolate the initial and final control points $\mathbf{p}_0$ and $\mathbf{p}_n$.

To simplify notation, we define the vector of generalized coordinates $\mathbf{p}_i(t)$ and weights $w_i(t)$ as

$$\mathbf{p}(t) = \left[\begin{array}{cccccc} \mathbf{p}_0^\top & w_0 & \cdots & \mathbf{p}_n^\top & w_n \end{array} \right]^\top,$$

where $^\top$ denotes transposition. We then express the curve (1) as $\mathbf{c}(u,\mathbf{p})$ in order to emphasize its dependence on $\mathbf{p}$ whose components are functions of time.

The velocity of the kinematic spline is

$$\dot{\mathbf{c}}(u,\mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{2}$$

where the overstruck dot denotes a time derivative and $\mathbf{J}(u,\mathbf{p})$ is the Jacobian matrix. Because $\mathbf{c}$ is a 3-component vector-valued function and $\mathbf{p}$ is an $4(n+1)$ dimensional vector, $\mathbf{J}$ is a $3 \times 4(n+1)$ matrix, which is expressed as

$$\mathbf{J} = \left[\begin{array}{cccc} \cdots & \left[\begin{array}{ccc} \frac{\partial \mathbf{c}_x}{\partial \mathbf{p}_{i,x}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{c}_y}{\partial \mathbf{p}_{i,y}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{c}_z}{\partial \mathbf{p}_{i,z}} \end{array} \right] & \frac{\partial \mathbf{c}}{\partial w_i} & \cdots \end{array} \right] \tag{3}$$

where

$$\frac{\partial \mathbf{c}_x}{\partial \mathbf{p}_{i,x}} = \frac{\partial \mathbf{c}_y}{\partial \mathbf{p}_{i,y}} = \frac{\partial \mathbf{c}_z}{\partial \mathbf{p}_{i,z}} = \frac{w_i B_{i,k}}{\sum_{j=0}^{n} w_j B_{j,k}};$$

$$\frac{\partial \mathbf{c}}{\partial w_i} = \frac{\sum_{j=0}^{n} (\mathbf{p}_i - \mathbf{p}_j) w_j B_{i,k} B_{j,k}}{(\sum_{j=0}^{n} w_j B_{j,k})^2}.$$

The subscript x, y, and z denote the component of a 3-vector. Furthermore, we can express the curve as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{c}(u, \mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{4}$$

The proof of (4) can be found in [19].

3 Dynamic NURBS Swung Surface

In this section, we formulate the underlying geometry of the dynamic swung surfaces and derive the Jacobian and basis function matrices that lead to succinct expressions analogous to (2) and (4) for the velocity and position functions of the surface, respectively. This allows us to derive equations of motion for the dynamic swung surface including mass, damping, and deformation energy distributions.

Geometrically, a dynamic swung surface is generated from two planar kinematic NURBS profile curves through the swinging operation [1] (Fig 1). Let the two generator curves $\mathbf{c}_1(u, \mathbf{a})$ and $\mathbf{c}_2(v, \mathbf{b})$ be of the form (1). The swung surface is then defined as

$$\mathbf{s}(u, v, t) = \left[\begin{array}{ccc} \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,x} & \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,y} & \mathbf{c}_{1,z} \end{array} \right]^\top \tag{5}$$

where α is an arbitrary scalar. The second subscript denotes the component of a 3-vector.

Assume that $\mathbf{c}_1$ has basis functions of degree $k-1$ and that it has $m+1$ control points $\mathbf{a}_i(t)$ and weights $w_i^a(t)$. Similarly, $\mathbf{c}_2$ has basis functions of degree $l-1$ and that it has $n+1$ control points $\mathbf{b}_j(t)$ and weights $w_j^b(t)$. Therefore,

$$\mathbf{a}(t) = [\mathbf{a}_0^\top, w_0^a, \ldots, \mathbf{a}_m^\top, w_m^a]^\top$$

and

$$\mathbf{b}(t) = [\mathbf{b}_0^\top, w_0^b, \ldots, \mathbf{b}_n^\top, w_n^b]^\top$$

are the generalized coordinate vectors of the profile curves. We collect these into the generalized coordinate vector

$$\mathbf{p} = \left[\begin{array}{ccc} \alpha & \mathbf{a}^\top & \mathbf{b}^\top \end{array} \right]^\top.$$

This vector has dimensionality $M = 1 + 4(m+1) + 4(n+1)$. Thus the model has $O(n+m)$ degrees of freedom, compared to $O(nm)$ for general NURBS surfaces.

3.1 Jacobian Matrix

Denoting the Jacobian matrices of the two profile curves as $\mathbf{J}_1(u, \mathbf{a})$ and $\mathbf{J}_2(v, \mathbf{b})$, the curve position and velocity functions take the form of (2) and (4):

$$\begin{array}{ll} \mathbf{c}_1(u, \mathbf{a}) = \mathbf{J}_1\mathbf{a}, & \dot{\mathbf{c}}_1(u, \mathbf{a}) = \mathbf{J}_1\dot{\mathbf{a}}, \\ \mathbf{c}_2(v, \mathbf{b}) = \mathbf{J}_2\mathbf{b}, & \dot{\mathbf{c}}_2(v, \mathbf{b}) = \mathbf{J}_2\dot{\mathbf{b}}, \end{array}$$

where $\mathbf{J}_1$ is a $3 \times 4(m+1)$ matrix, and $\mathbf{J}_2$ is a $3 \times 4(n+1)$ matrix. Both are of the form (3).

If we express each row vector of the Jacobian matrices explicitly as $\mathbf{X}_i$, $\mathbf{Y}_i$ and $\mathbf{Z}_i$, we can write the block forms:

$$\mathbf{J}_1 = \left[\begin{array}{ccc} \mathbf{X}_1^\top & \mathbf{Y}_1^\top & \mathbf{Z}_1^\top \end{array} \right]^\top$$

and

$$\mathbf{J}_2 = \left[\begin{array}{ccc} \mathbf{X}_2^\top & \mathbf{Y}_2^\top & \mathbf{Z}_2^\top \end{array} \right]^\top.$$

The swung surface is therefore written as

$$\mathbf{s}(u,v,\mathbf{p}) = \begin{bmatrix} \alpha(t)(\mathbf{X}_1\mathbf{a})(\mathbf{X}_2\mathbf{b}) \\ \alpha(t)(\mathbf{X}_1\mathbf{a})(\mathbf{Y}_2\mathbf{b}) \\ \mathbf{Z}_1\mathbf{a} \end{bmatrix}. \tag{6}$$

The velocity of the swung surface is

$$\dot{\mathbf{s}}(u,v,\mathbf{p}) = \mathbf{L}\dot{\mathbf{p}} \tag{7}$$

where $\mathbf{L}(u,v,\mathbf{p})$ is the Jacobian matrix with respect to the generalized coordinate vector $\mathbf{p}$. Hence, $\mathbf{L}$ comprises the vectors $\partial\mathbf{s}/\partial\alpha$, $\partial\mathbf{s}/\partial\mathbf{a}$, and $\partial\mathbf{s}/\partial\mathbf{b}$, which are given as follows:

$$\frac{\partial\mathbf{s}}{\partial\alpha} = \begin{bmatrix} (\mathbf{X}_1\mathbf{a})(\mathbf{X}_2\mathbf{b}) \\ (\mathbf{X}_1\mathbf{a})(\mathbf{Y}_2\mathbf{b}) \\ 0 \end{bmatrix} = \mathbf{A}\mathbf{c}_2 = (\mathbf{B}-\mathbf{C})\mathbf{c}_1$$

where

$$\mathbf{A}(u,\mathbf{a}) = \begin{bmatrix} \mathbf{X}_1\mathbf{a} & 0 & 0 \\ 0 & \mathbf{X}_1\mathbf{a} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \mathbf{B}(v,\mathbf{b}) = \begin{bmatrix} \mathbf{X}_2\mathbf{b} & 0 & 0 \\ \mathbf{Y}_2\mathbf{b} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$\frac{\partial\mathbf{s}}{\partial\mathbf{a}} = \begin{bmatrix} \alpha(\mathbf{X}_2\mathbf{b})\mathbf{X}_1 \\ \alpha(\mathbf{Y}_2\mathbf{b})\mathbf{X}_1 \\ \mathbf{Z}_1 \end{bmatrix} = \mathbf{B}_\alpha\mathbf{J}_1,$$

where $\mathbf{B}_\alpha(\alpha,v,\mathbf{b}) = \alpha\mathbf{B} + (1-\alpha)\mathbf{C}$; and

$$\frac{\partial\mathbf{s}}{\partial\mathbf{b}} = \begin{bmatrix} \alpha(\mathbf{X}_1\mathbf{a})\mathbf{X}_2 \\ \alpha(\mathbf{X}_1\mathbf{a})\mathbf{Y}_2 \\ \mathbf{0} \end{bmatrix} = \mathbf{A}_\alpha\mathbf{J}_2,$$

where $\mathbf{A}_\alpha(\alpha,u,\mathbf{a}) = \alpha\mathbf{A}$. Hence, we express the Jacobian matrix as

$$\mathbf{L} = \begin{bmatrix} \mathbf{A}\mathbf{c}_2 & \mathbf{B}_\alpha\mathbf{J}_1 & \mathbf{A}_\alpha\mathbf{J}_2 \end{bmatrix} \tag{8}$$

Note that $\mathbf{A}$, $\mathbf{A}_\alpha$, $\mathbf{B}$, $\mathbf{B}_\alpha$, and $\mathbf{C}$ are 3×3 matrices. Therefore, $\mathbf{A}\mathbf{c}_2$ is a 3 vector, $\mathbf{B}_\alpha\mathbf{J}_1$ is a $3\times 4(m+1)$ matrix, and $\mathbf{A}_\alpha\mathbf{J}_2$ is a $3\times 4(n+1)$ matrix. Thus, $\mathbf{L}$ is a $3\times M$ matrix.

3.2 Basis Function Matrix

Unlike $\mathbf{J}$ in (4), $\mathbf{L}$ cannot also serve as the basis function matrix of the swung surface. Let

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{B}_\alpha\mathbf{J}_1 & \mathbf{0} \end{bmatrix}, \qquad \mathbf{H}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{C}\mathbf{J}_1 & \mathbf{A}_\alpha\mathbf{J}_2 \end{bmatrix},$$

$$\mathbf{H}_3 = \begin{bmatrix} \mathbf{A}\mathbf{c}_2 & \mathbf{C}\mathbf{J}_1 & \mathbf{0} \end{bmatrix}, \qquad \mathbf{H}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{C}\mathbf{J}_1 & \mathbf{0} \end{bmatrix}.$$

It is straightforward to verify that

$$3\mathbf{s}(u,v,\mathbf{p}) = \mathbf{H}_1\mathbf{p} + \mathbf{H}_2\mathbf{p} + \mathbf{H}_3\mathbf{p} = \mathbf{L}\mathbf{p} + 2\mathbf{H}_4\mathbf{p}.$$

Thus we have

$$\mathbf{s}(u,v,\mathbf{p}) = \mathbf{H}\mathbf{p}, \tag{9}$$

where

$$\mathbf{H} = (\mathbf{L} + 2\mathbf{H}_4)/3 \tag{10}$$

is the $3 \times M$ basis function matrix.

3.3 Equations of Motion

The equations of motion of our dynamic NURBS swung surface are derived from the work-energy version of Lagrangian dynamics [22]. To proceed with the Lagrangian formulation, we express the kinetic energy due to a prescribed mass distribution function $\mu(u, v)$ over the parametric domain of the surface and a Raleigh dissipation energy due to a damping density function $\gamma(u, v)$. To define an elastic potential energy, we adopt the *thin-plate under tension* energy model which was proposed in [18] and also used in [12, 14, 17, 19]

$$U = \frac{1}{2}\iint \left(\alpha_{1,1}\frac{\partial \mathbf{s}^\top}{\partial u}\frac{\partial \mathbf{s}}{\partial u} + \alpha_{2,2}\frac{\partial \mathbf{s}^\top}{\partial v}\frac{\partial \mathbf{s}}{\partial v} + \beta_{1,1}\frac{\partial^2 \mathbf{s}^\top}{\partial u^2}\frac{\partial^2 \mathbf{s}}{\partial u^2} + \beta_{1,2}\frac{\partial^2 \mathbf{s}^\top}{\partial u \partial v}\frac{\partial^2 \mathbf{s}}{\partial u \partial v} + \beta_{2,2}\frac{\partial^2 \mathbf{s}^\top}{\partial v^2}\frac{\partial^2 \mathbf{s}}{\partial v^2}\right) du\, dv. \tag{11}$$

The $\alpha_{i,j}(u, v)$ and $\beta_{i,j}(u, v)$ are elasticity functions which control tension and rigidity, respectively, in the two parametric coordinate directions. Other energies are applicable, including the nonquadratic, curvature-based energies in [8, 16]).

Applying the Lagrangian formulation, we obtain the second-order nonlinear equations of motion

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \tag{12}$$

where the mass matrix is

$$\mathbf{M}(\mathbf{p}) = \iint \mu \mathbf{L}^\top \mathbf{L}\, du\, dv,$$

the damping matrix is

$$\mathbf{D}(\mathbf{p}) = \iint \gamma \mathbf{L}^\top \mathbf{L}\, du\, dv,$$

and the stiffness matrix is

$$\mathbf{K}(\mathbf{p}) = \iint (\alpha_{1,1}\mathbf{L}_u^\top \mathbf{H}_u + \alpha_{2,2}\mathbf{L}_v^\top \mathbf{H}_v + \beta_{1,1}\mathbf{L}_{uu}^\top \mathbf{H}_{uu} + \beta_{1,2}\mathbf{L}_{uv}^\top \mathbf{H}_{uv} + \beta_{2,2}\mathbf{L}_{vv}^\top \mathbf{H}_{vv})\, du\, dv$$

(the subscripts on $\mathbf{L}$ and $\mathbf{H}$ denote parametric partial derivatives). $\mathbf{M}$, $\mathbf{D}$ and $\mathbf{K}$ are $M \times M$ matrices. The generalized force, obtained through the principle of virtual work [22] done by the applied force distribution $\mathbf{f}(u, v, t)$ is

$$\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{L}^\top \mathbf{f}(u, v, t)\, du\, dv.$$

Because of the geometric nonlinearity, generalized inertial forces

$$\mathbf{g}_p(\mathbf{p}) = -\iint \mu \mathbf{L}^\top \dot{\mathbf{L}}\dot{\mathbf{p}}\, du\, dv$$

are also associated with the models. The derivation of the equations of motion (12) proceeds in the same manner as for D-NURBS (see [19] for the details).

4 Numerical Simulation

The evolution of $\mathbf{p}$, determined by (12) with time-varying matrices, cannot be solved analytically in general. Instead, we pursue an efficient numerical implementation using finite-element techniques [23].

Standard finite element codes explicitly assemble the global matrices that appear in the discrete equations of motion [23]. We use an iterative matrix solver to avoid the cost of assembling the global $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$. In this way, we work with the individual element matrices and construct finite element data structures that permit the parallel computation of element matrices.

4.1 Matrix Structure and Computation

We examine the mass and damping matrices. Both matrices involve the integration of $\mathbf{L}^\top\mathbf{L}$ in the parametric domain where $\mathbf{L}$ is given in (8). Based on (8), $\mathbf{L}^\top\mathbf{L}$ is decomposed into the following block matrices:

$$\mathbf{L}^\top\mathbf{L} = \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \mathbf{F}_{1,3} \\ & \mathbf{F}_{2,2} & \mathbf{F}_{2,3} \\ \textbf{symmetric} & & \mathbf{F}_{3,3} \end{bmatrix} \tag{13}$$

where
$\mathbf{F}_{1,1} = \mathbf{c}_2^\top\mathbf{A}^\top\mathbf{A}\mathbf{c}_2 = (\mathbf{X}_1\mathbf{a})^2\,\|\mathbf{c}_2\|^2$,
$\mathbf{F}_{1,2} = \mathbf{c}_2^\top\mathbf{A}^\top\mathbf{B}_\alpha\mathbf{J}_1 = \alpha\,\|\mathbf{c}_2\|^2\,(\mathbf{X}_1\mathbf{a})\mathbf{X}_1$,
$\mathbf{F}_{1,3} = \mathbf{c}_2^\top\mathbf{A}^\top\mathbf{A}_\alpha\mathbf{J}_2 = \alpha(\mathbf{X}_1\mathbf{a})^2\mathbf{c}_2^\top\mathbf{J}_2$,
$\mathbf{F}_{2,2} = \mathbf{J}_1^\top\mathbf{B}_\alpha^\top\mathbf{B}_\alpha\mathbf{J}_1 = \alpha^2\,\|\mathbf{c}_2\|^2\,\mathbf{X}_1^\top\mathbf{X}_1 + \mathbf{Z}_1^\top\mathbf{Z}_1$,
$\mathbf{F}_{2,3} = \mathbf{J}_1^\top\mathbf{B}_\alpha^\top\mathbf{A}_\alpha\mathbf{J}_2 = \alpha^2\mathbf{X}_1^\top(\mathbf{X}_1\mathbf{a})\mathbf{c}_2^\top\mathbf{J}_2$, and
$\mathbf{F}_{3,3} = \mathbf{J}_2^\top\mathbf{A}_\alpha^\top\mathbf{A}_\alpha\mathbf{J}_2 = \alpha^2(\mathbf{X}_1\mathbf{a})^2\mathbf{J}_2^\top\mathbf{J}_2$.

See Appendix A for the details about the stiffness matrix.

4.2 Element Data Structures

We define an element data structure which contains the geometric specification of the surface patch element along with its physical properties. A complete dynamic swung surface is then implemented as a data structure which consists of an ordered array of elements with additional information. The element structure includes pointers to appropriate components of the global vector $\mathbf{p}$ (control points and weights). Neighboring elements will share some generalized coordinates. The shared variables will have multiple pointers impinging on them. We also allocate in each element an elemental mass, damping, and stiffness matrix, and include in the element data structure the quantities needed to compute these matrices. These quantities include the mass $\mu(u,v)$, damping $\gamma(u,v)$, and elasticity $\alpha_{i,j}(u,v)$, $\beta_{i,j}(u,v)$ density functions, which may be represented as analytic functions or as parametric arrays of sample values.

4.3 Calculation of Element Matrices

The integral expressions for the mass, damping, and stiffness matrices associated with each element are evaluated numerically using Gaussian quadrature [24]. We shall explain the computation of the element mass matrix; the computation of the damping and stiffness matrices follow suit. Assuming the parametric domain of the element is $[u_0,u_1]\times[v_0,v_1]$, the expression for entry m_{ij} of the mass matrix takes the integral form

$$m_{ij} = \int_{u_0}^{u_1}\int_{v_0}^{v_1} \mu(u,v) f_{ij}(u,v)\,du\,dv,$$

where f_{ij} are entries of the matrix in (13). Given integers N_g and N_h, we can find Gauss weights a_g, b_h and abscissas u_g, v_h in the two parametric directions such that m_{ij} can be approximated by [24]

$$m_{ij} \approx \sum_{g=1}^{N_g}\sum_{h=1}^{N_h} a_g b_h \mu(u_g,v_h) f_{ij}(u_g,v_h).$$

We apply the de Boor algorithm [25] to evaluate $f_{ij}(u_g, v_h)$. In general, Gaussian quadrature evaluates the integral exactly with N weights and abscissas for polynomials of degree $2N - 1$ or less. In our system we choose N_g and N_h to be integers between 4 and 7. Our experiments indicate that matrices computed in this way lead to stable, convergent solutions.

Note that in the case where the mass, damping, and stiffness properties are uniform over the surface and, therefore, reduce to scalar quantities, the double sum in the Gaussian integration formula decomposes into the product of two independent sums over each of the univariate domains of the generator curves and it becomes much more efficient.

4.4 Discrete Dynamics Equations

To integrate (12) in an interactive modeling environment, it is important to provide the modeler with visual feedback about the evolving state of the dynamic model. Rather than using costly time integration methods that take the largest possible time steps, it is more crucial to provide a smooth animation by maintaining the continuity of the dynamics from one step to the next. Hence, less costly yet stable time integration methods that take modest time steps are desirable.

The state of the dynamic NURBS swung surface at time $t + \Delta t$ is integrated using prior states at time t and $t - \Delta t$. To maintain the stability of the integration scheme, we use an implicit time integration method, which employs discrete derivatives of $\mathbf{p}$ using backward differences

$$\ddot{\mathbf{p}}^{(t+\Delta t)} = (\mathbf{p}^{(t+\Delta t)} - 2\mathbf{p}^{(t)} + \mathbf{p}^{(t-\Delta t)})/\Delta t^2,$$

and

$$\dot{\mathbf{p}}^{(t+\Delta t)} = (\mathbf{p}^{(t+\Delta t)} - \mathbf{p}^{(t-\Delta t)})/2\Delta t.$$

We obtain the time integration formula

$$\left(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K}\right)\mathbf{p}^{(t+\Delta t)} = 2\Delta t^2(\mathbf{f}_p + \mathbf{g}_p) + 4\mathbf{M}\mathbf{p}^{(t)} - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)} \tag{14}$$

where the superscripts denote evaluation of the quantities at the indicated times. The matrices and forces are evaluated at time t.

We employ the conjugate gradient method to obtain an iterative solution [24]. To achieve interactive simulation rates, we limit the number of conjugate gradient iterations per time step to 10. We have observed that 2 iterations typically suffice to converge to a residual of less than 10^{-3}. More than 2 iterations tend to be necessary when the physical parameters (mass, damping, tension, stiffness, applied forces) are changed significantly during dynamic simulation. Hence, our implementation permits the real-time simulation of dynamic swung surfaces on common graphics workstations. Quadratic and cubic surfaces with more than 200 constrained control points can be simulated at interactive rates.

The equations of motion allow realistic dynamics such as would be desirable for physics-based computer graphics animation. It is possible, however, to make simplifications that further reduce the computational cost of (14) to interactively sculpt larger surfaces. For example, in CAGD applications such as data fitting where the modeler is interested only in the final equilibrium configuration of the model, it makes sense to simplify (12) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish. This economizes on storage and makes the algorithm more efficient. With zero mass density, (12) reduces to the first-order system

$$\mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p. \tag{15}$$

Discretizing the derivatives of $\mathbf{p}$ in (15) with backward differences, we obtain the integration formula

$$(\mathbf{D} + \Delta t\mathbf{K})\,\mathbf{p}^{(t+\Delta t)} = \Delta t\mathbf{f}_p + \mathbf{D}\mathbf{p}^{(t)} \tag{16}$$

5 Physics-Based Shape Design

In the physics-based shape design approach, design requirements may be satisfied through the use of energies, forces, and constraints. The designer may apply time-varying forces to sculpt shapes interactively or to optimally approximate data. Certain aesthetic constraints such as "fairness" are expressible in terms of elastic energies that give rise to specific stiffness matrices $\mathbf{K}$. Other constraints include position or normal specification at surface points, and continuity requirements between adjacent surface patches. By building the dynamic swung surface upon the standard geometry of the NURBS swung surface, we allow the modeler to continue to use the whole spectrum of advanced geometric design tools that have become prevalent, among them, the imposition of geometric constraints that the final shape must satisfy.

5.1 Applied Forces

Sculpting tools may be implemented as applied forces. The force $\mathbf{f}(u, v, t)$ represents the net effect of all applied forces. Typical force functions are spring forces, repulsion forces, gravitational forces, inflation forces, etc. [8].

For example, consider connecting a material point (u_0, v_0) of a dynamic swung surface to a point $\mathbf{d}_0$ in space with an ideal Hookean spring of stiffness k. The net applied spring force is

$$\mathbf{f}(u, v, t) = \iint k(\mathbf{d}_0 - \mathbf{s}(u, v, t))\delta(u - u_0, v - v_0)\, du\, dv, \tag{17}$$

where the δ is the unit delta function. Equation (17) implies that $\mathbf{f}(u_0, v_0, t) = k(\mathbf{d}_0 - \mathbf{s}(u_0, v_0, t))$ and vanishes elsewhere on the surface, but we can generalize it by replacing the δ function with a smooth kernel (e.g., a unit Gaussian) to spread the applied force over a greater portion of the surface. Furthermore, the points (u_0, v_0) and $\mathbf{d}_0$ need not be constant, in general. We can control either or both using a mouse to obtain an interactive spring force.

5.2 Constraints

In practical applications, design requirements may be posed as a set of physical parameters or as geometric constraints. Nonlinear constraints can be enforced through Lagrange multiplier techniques [26, 27, 28]. This approach increases the number of degrees of freedom, hence the computational cost, by adding unknowns λ_i, known as Lagrange multipliers, which determine the magnitudes of the constraint forces. The augmented Lagrangian method [26] combines the Lagrange multipliers with the simpler penalty method [9]. The Baumgarte stabilization method [29] solves constrained equations of motion through linear feedback control (see also [10, 19]). These techniques are appropriate for the dynamic swung surfaces with constraints.

Linear geometric constraints such as point, curve, and surface normal constraints can be easily incorporated into dynamic swung surface by reducing the matrices and vectors in (12) to a minimal unconstrained set of generalized coordinates. For example, the two generator curves must be embedded in $x - z$ and $x - y$ planes, respectively. If the model is confined as a surface of revolution, the degrees of freedom associated with the second profiles must be constrained geometrically to admit a circle. Linear constraints can be implemented by applying the same numerical solver on an unconstrained subset of $\mathbf{p}$. See [19] for a detailed discussion on constraints in the context of D-NURBS.

Dynamic surfaces constructed from NURBS geometry have an interesting idiosyncrasy due to the weights. While the control point components of $\mathbf{p}$ may take arbitrary finite values in $\Re$, negative weights may cause the denominator to vanish at some evaluation points, causing the matrices to diverge. Although not forbidden, negative weights are not useful. We enforce positivity of weights at each simulation time step by simply projecting any weight value that has drifted below a small positive threshold back to this lower bound (nominally 0.1). Another potential difficulty is that lower weight values tend to flatten the surface in the vicinity of the control points, lowering the deformation energy; thus the weights may tend to decrease. One solution is to use a more complex deformation energy that does not favor flat surfaces as in [16]. Alternatively, we can counteract the tendency and also give the designer the option of constraining the weights near certain desired target values w_i^0 by including in the surface energy the penalty term $c\sum(w_i - w_i^0)$, where c controls the tightness of the constraint.

6 Applications and Results

We have developed a prototype modeling system based on dynamic NURBS swung surfaces. Currently, the system implements surfaces with basis functions of order 2, 3, or 4 (i.e., from linear to cubic) and geometric constraints. The system is written in C and is packaged as an interactive Iris Explorer module on Silicon Graphics workstations. It may be combined with existing Explorer modules for data input and surface visualization. Our parallelized iterative numerical algorithm takes advantage of an SGI Iris 4D/380VGX multiprocessor.

Users can sculpt surface shapes in conventional geometric ways, such as by sketching control polygons of arbitrary profile curves, repositioning control points, and adjusting associated weights, or according to the physics-based paradigm through the use of forces. They can satisfy design requirements by adjusting the internal physical parameters such as the mass, damping, and stiffness densities, along with force gain factors, interactively through Explorer control panels. The following sections demonstrate applications of dynamic NURBS swung surfaces to rounding and blending, scattered data fitting, and interactive sculpting.

6.1 Rounding and Blending

The rounding and blending of surfaces is usually attempted geometrically, by enforcing continuity requirements on the fillet which interpolates between two or more surfaces. By contrast, the dynamic NURBS swung surface can produce a smooth fillet by minimizing its internal deformation energy subject to position and normal constraints. The dynamic simulation automatically produces the desired final shape as it achieves equilibrium.

Fig. 5 demonstrates the rounding of a polyhedral toroid. The profile curve on the x-z plane is a quadratic NURBS curve with 17 control points. The trajectory curve on the x-y plane is also a quadratic NURBS with 17 control points. Note that, the corners of the curves can be represented exactly with multiple control points or approximately by setting a very large weight. If this model were a general NURBS surface, it would have 289 control points and weights. As a swung surface it has only 34 control points and weights which are considered the generalized coordinates of the dynamic model. The wireframe and shaded shapes is shown in Fig. 5(a) and Fig. 5(b). After initiating the physical simulation, the corners and sharp edges are rounded as the final shape equilibrates into the minimal energy state shown in Fig. 5(c-d).

Fig. 6 illustrates the rounding of a cubical solid. The profile is a quadratic NURBS with 15 control points, and the trajectory is a linear NURBS with 5 control points. The rounding operation is applied in the vicinity of the middle edge. In Fig. 6(a-b) shows the wireframe and shaded objects, respectively. The rounded shape is shown in Fig. 6(c-d).

Fig. 7 shows a blending example involving a cylindrical pipe. The circular profile is a quadratic curve with 7 control points. The piecewise linear trajectory has 5 control points. The initial right-angle pipe and the final rounded pipe are shown in Fig. 7(a-d).

To demonstrate automatic weight variation in the dynamic model in accordance with physical parameters, we have conducted an experiment with the polyhedral toroid shown in Fig. 5(a). We fix all 137 degrees of freedom of the toroid, which occupies the cube $-1 \leq x, y, z, \leq 1$, except for one weight associated with the control point $(1.0, 0.0, 1.0)$ which is initially set to 50.0 to form the corner. This weight is permitted to vary subject to varying physical parameters and external force (Fig. 2). Initially $\gamma = 100.0$ and the remaining parameters are zero ($\Delta t = 0.248$). Starting the dynamic simulation, we first gradually increase $\alpha_{1,2}$ from zero to 150.0. As shown in Fig. 2, the value of the free weight decreases quickly towards its lower limit 0.3 in under 100 iterations (dashed plot). This reduces the deformation energy by rounding the corner. Next, we attach a spring force from the point $(1.0, 1.0, 1.0)$ to the nearest surface point. As

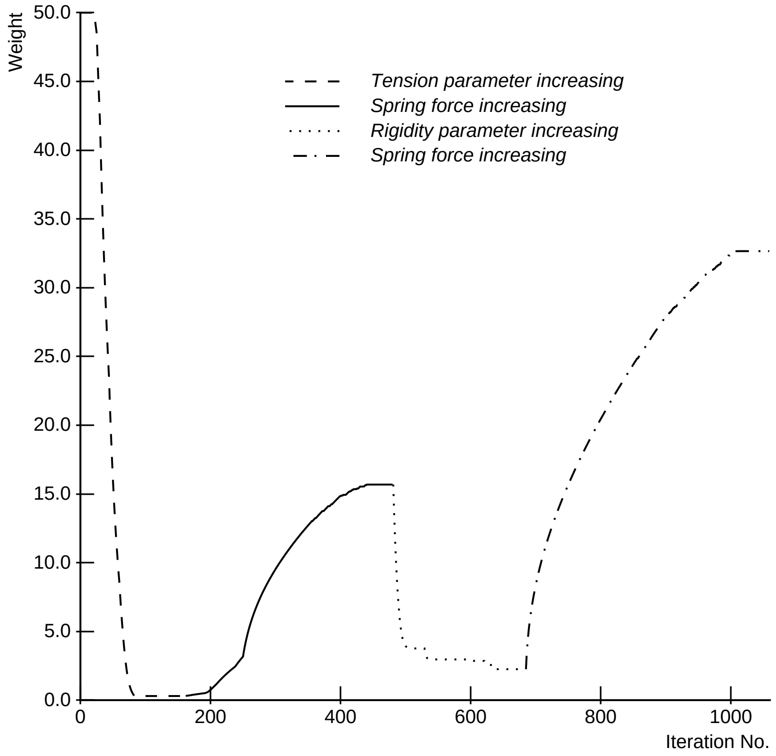


Figure 2: Free weight variation subject to varying physical parameters and external force (see text).

the stiffness constant of this spring is slowly increased from zero to 1,000.0, the weight increases (solid plot in Fig. 2) thus recreating a corner. Next, we vary $\beta_{2,2}$ from zero to 100.0 (dotted plot in Fig. 2). The increased rigidity of the surface causes the weight to decrease, again rounding the corner. Finally, we increase the spring stiffness to 10,000.0 (stippled plot in Fig. 2), thus counteracting the increased stiffness and causing the weight to increases to again recreate the corner. This experiment demonstrates that when control points are fixed, a free weight will automatically decrease to decrease the deformation energy by flattening the surface locally, unless external forces counteract this tendency. When control points are free, they will also vary as well to produce an analogous physical effect.

6.2 Scattered Data Fitting

A useful modeling technique is based on fitting surfaces to unstructured constraints, generally known as scattered data fitting. Interesting situations arise when there are fewer or more data points than there are degrees of freedom in the model, leading to underconstrained or overconstrained fitting problems. The inclusion of an elastic energy in our dynamic surfaces makes them applicable to such problems.

The data interpolation problem is amenable to common constraint techniques [26]. Approximation can be achieved by physically coupling the dynamic NURBS swung surface to the data through Hookean spring forces (17). We interpret $\mathbf{d}_0$ in (17) as the data point (generally in $\Re^3$) and (u_0, v_0) as the parametric coordinates associated with the data point (which may be the nearest material point of the surface). The spring constant c determines the closeness of fit to the data point. [2]

To find the closest point on the model for arbitrarily sampled data (x_0, y_0, z_0), we exploit the special symmetric structure of NURBS swung surface through the following two-step search scheme. We first find

[2]Cross-validation provides a principled approach to choosing the relevant physical parameters—typically the ratio of data force spring constants to surface stiffnesses—for given data sets [30]. For the special case of zero-mean Gaussian data errors, optimal approximation in the least squares residual sense results when c is proportional to the inverse variance of data errors.

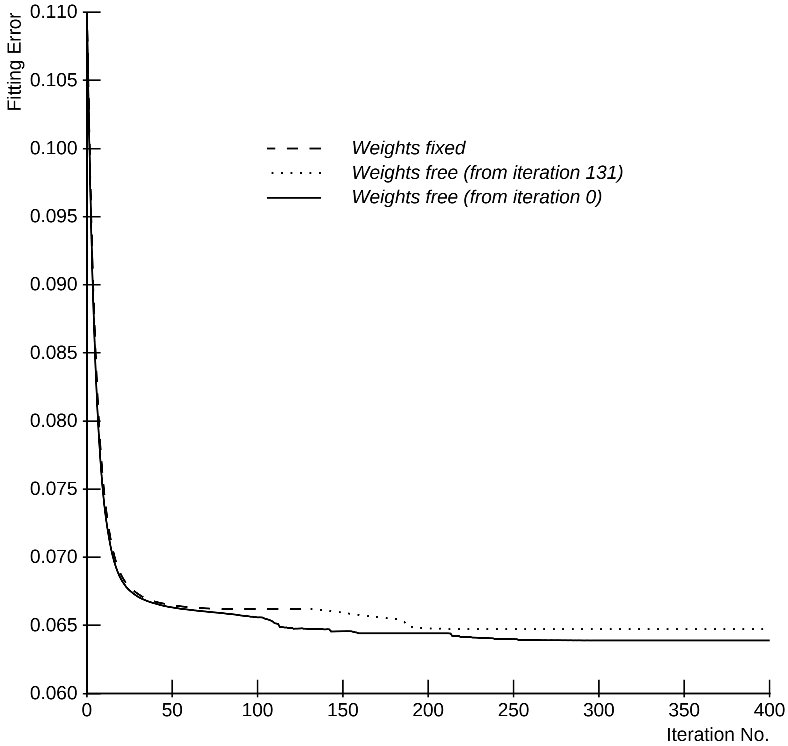


Figure 3: Least Squared Fitting Errors For NRCC Pot (see text).

the v_0 such that $\mathbf{c}_1(v_0)$ is nearest to (x_0, y_0). Then we search the isoparametric curve $\mathbf{s}(u, v_0)$ and find the u_0 such that $\mathbf{s}(u_0, v_0)$ is the closest to (x_0, y_0, z_0). Experiments show that this approximation approach leads to satisfactory results because the mapping is recomputed at each simulation step. More importantly, we reduce the complexity of optimal matching from $O(mn)$ for a general m by n D-NURBS surface [19] to $O(m + n)$. For large m and n, the dynamic simulation is speeded up significantly. Other techniques such as nonlinear optimization are applicable to finding the closest point.

An important advantage of our models, despite the fact that they are profile surfaces, is that they can be fitted to arbitrarily distributed empirical data that are not aligned along any particular isoparametric curve pattern. We first use a dynamic swung surface generated by two quadratic profiles with 10 and 7 control points to reconstruct a clay pot which has been densely sampled by a cylindrical laser scanner to produce about 1.2×10^6 data points. The 7 control points and weights of the cross-section profile are constrained so as to permit only surfaces of circular cross section. We randomly selected only 20 data points from which to reconstruct a surface of revolution. Fig. 8 shows the sample points, the cylindrical initial condition, and the final fitted shape rendered with the texture map of the object acquired by the scanner. The elastic energy of the surface allows it to interpolate between data points. The physical parameters used in this experiment were mass $\mu = 0.0$, damping $\gamma = 60.0$, bending stiffness parameter $\beta_{1,1} = 14.0$ while all the others are zero, data spring constants $c_i = 900.0$. The surface fitting stabilizes in a few seconds with a time step $\Delta t = 0.3$.

We report two more experiments with the above surface fitting scenario in order to investigate the least-squares fitting errors under different circumstances. In the first experiment, we initially fix all 10 weights of the profile curve. An optimal fit is achieved after 81 iterations. The decreasing error is plotted by the dashed plot in Fig. 3. Then we free the weights at iteration 131 and the model is able to further reduce the fitting error because it now has more degrees of freedom at its disposal. The dotted plot in the figure indicates the improved fit. In the second experiment, the weights are set free from the start of the dynamic simulation. After 295 iterations, the optimal fitting is recorded by the solid plot in Fig. 3. The

Iteration No.	*Weights*									
	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
initial	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
80	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
160	1.012	1.011	1.011	1.096	0.884	0.978	0.998	1.019	0.957	1.023
240	1.042	1.030	1.041	1.260	0.513	0.941	1.031	1.029	0.893	1.065
320	1.042	1.030	1.041	1.260	0.513	0.941	1.031	1.029	0.893	1.065
final	1.042	1.030	1.041	1.260	0.513	0.941	1.031	1.029	0.893	1.065

Table 1: Variation of weights in experiment 1.

Iteration No.	*Weights*									
	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
initial	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
80	1.004	1.008	1.004	1.060	0.925	0.997	1.001	1.010	0.977	1.008
160	1.056	1.045	1.059	1.350	0.392	0.785	1.066	1.026	0.864	1.087
240	1.072	1.058	1.076	1.377	0.304	0.623	1.126	1.020	0.845	1.106
320	1.094	1.079	1.098	1.357	0.300	0.587	1.144	1.013	0.826	1.130
final	1.094	1.079	1.098	1.357	0.300	0.587	1.144	1.013	0.826	1.130

Table 2: Variation of weights in experiment 2.

weight values in the two experiments are given in Table 1 and 2, respectively. The experiments indicate that, at least in this situation, a slightly better surface fit is obtained when the model is permitted to use all of the available degrees of freedom from the start of the fitting process. Obviously, the final surface in the first experiment had attained a local optimum.

Next, we use the same surface model to approximate 10 data points sampled from a vase. The wireframe and textured images of the dynamic swung surface are illustrated in Fig. 9. Fig. 10(a-d) shows the final reconstructed shapes from four other fitting experiments using synthetic data to recover another pot, a vase, a bottle, and a wine glass. The number of randomly sampled data are 10, 13, 14, and 17, respectively.

6.3 Interactive Sculpting

In the physics-based modeling approach, not only can the designer manipulate the individual degrees of freedom with conventional geometric methods, but he can also move the object or refine its shape with interactive sculpting forces.

The physics-based modeling approach is ideal for interactive sculpting of surfaces. It provides direct manipulation of the dynamic surface to refine the shape of the surface through the application of interactive sculpting tools in the form of forces. Fig. 4(a) illustrates the results of four interactive sculpting sessions using spring forces. A sphere was generated using two quadratic curves with 4 and 7 control points and was sculpted into the ovoid shown in Fig. 4(a). A torus whose two profile curves are quadratic with 7 and 7 control points, respectively, has been deformed into the shape in Fig. 4(b). A hat shape was created from two curves with 9 and 6 control points and was then deformed by spring forces into the shape in Fig 4(d). Finally, we generated a wine glass shape using two curves with 7 and 5 control points and sculpted it into the more pleasing shape shown in Fig 4(c).

7 Constrained D-NURBS Formulation

It is known that a geometric NURBS swung surface is a NURBS surface [1]. In this section, we show that dynamic NURBS swung surfaces are, analogously, D-NURBS surfaces [19] that have been subjected to a dimensionality-reducing nonlinear constraint.

7.1 D-NURBS Surface

A D-NURBS surface generalizes the geometric NURBS surface:

$$\mathbf{s}(u,v,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{q}_{i,j}(t) w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}. \tag{18}$$

The $(m+1)(n+1)$ control points $\mathbf{q}_{i,j}(t)$ and weights $w_{i,j}(t)$, which are functions of time, comprise the D-NURBS generalized coordinates. We concatenate these $N = 4(m+1)(n+1)$ coordinates into the vector:

$$\mathbf{q}(t) = \left[\begin{array}{cccc} \cdots & \mathbf{q}_{i,j}^{\top} & w_{i,j} & \cdots \end{array}\right]^{\top}.$$

Similar to (2) and (4), we have

$$\dot{\mathbf{s}}(u,v,\mathbf{q}) = \mathbf{J}\dot{\mathbf{q}}, \qquad \mathbf{s}(u,v,\mathbf{q}) = \mathbf{J}\mathbf{q}. \tag{19}$$

where $\mathbf{J}(u,v,\mathbf{q})$ is the $3 \times N$ Jacobian matrix of the D-NURBS surface with respect to $\mathbf{q}$. The motion equations of D-NURBS surfaces are

$$\mathbf{M}_q\ddot{\mathbf{q}} + \mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{g}_q, \tag{20}$$

where the mass matrix $\mathbf{M}_q$, the damping matrix $\mathbf{D}_q$, and the stiffness matrix $\mathbf{K}_q$ are all $N \times N$ matrices, $\mathbf{f}_q$ is the generalized force vector acting on $\mathbf{q}$, and $\mathbf{g}_q$ is the generalized inertial force. See [19] for the details of the D-NURBS formulation.

To reduce the D-NURBS surface to a dynamic swung surface, we apply the nonlinear constraint

$$\begin{aligned} \mathbf{q}_{i,j} &= \left[\begin{array}{ccc} \alpha\mathbf{a}_{i,x}\mathbf{b}_{j,x} & \alpha\mathbf{a}_{i,x}\mathbf{b}_{j,y} & \mathbf{a}_{i,z} \end{array}\right]^{\top} \\ w_{i,j} &= w_i^a w_j^b. \end{aligned} \tag{21}$$

where α, $\mathbf{a}_i$, w_i^a, $\mathbf{b}_j$, and w_j^b, for $i = 0,\ldots,m$ and $j = 0,\ldots,n$, are defined as in Section 3. Differentiating (21), we obtain

$$\begin{aligned} \dot{\mathbf{q}}_{i,j} &= \left[\begin{array}{c} \dot{\alpha}\mathbf{a}_{i,x}\mathbf{b}_{j,x} + \alpha\dot{\mathbf{a}}_{i,x}\mathbf{b}_{j,x} + \alpha\mathbf{a}_{i,x}\dot{\mathbf{b}}_{j,x} \\ \dot{\alpha}\mathbf{a}_{i,x}\mathbf{b}_{j,y} + \alpha\dot{\mathbf{a}}_{i,x}\mathbf{b}_{j,y} + \alpha\mathbf{a}_{i,x}\dot{\mathbf{b}}_{j,y} \\ \dot{\mathbf{a}}_{i,z} \end{array}\right] \\ \dot{w}_{i,j} &= \dot{w}_i^a w_j^b + w_i^a \dot{w}_j^b \end{aligned} \tag{22}$$

Using the notations in Section 3, we can rewrite (21) and (22) in the matrix form

$$\dot{\mathbf{q}} = \mathbf{G}\dot{\mathbf{p}}, \qquad \mathbf{q} = \mathbf{B}\mathbf{p}, \tag{23}$$

where $\mathbf{B}$ and $\mathbf{G}$ are $N \times M$ matrices with $M = (4m + 4n + 9)$.

Substituting (23) into (20), we arrive at the equations of motion for the dynamic NURBS swung surface (12), where the $M \times M$ mass, damping, and stiffness matrices are given by

$$\mathbf{M} = \mathbf{G}^{\top}\mathbf{M}_q\mathbf{G},\ \ \mathbf{D} = \mathbf{G}^{\top}\mathbf{D}_q\mathbf{G},\ \ \mathbf{K} = \mathbf{G}^{\top}\mathbf{K}_q\mathbf{B}.$$

The generalized forces with respect to $\mathbf{p}$ are

$$\mathbf{f}_p = \mathbf{G}^{\top}\mathbf{f}_q, \qquad \mathbf{g}_p = \mathbf{G}^{\top}(\mathbf{g}_q - \mathbf{M}_q\dot{\mathbf{G}}\dot{\mathbf{p}}).$$

The constraint reduces the $4(m+1)(n+1)$ generalized coordinates of the D-NURBS surface to the $4m+4n+9$ generalized coordinates of the dynamic NURBS swung surface.

8 Conclusion

We have proposed dynamic NURBS swung surfaces. Like the D-NURBS predecessor [19], the new model is a physics-based generalization of its geometric counterpart. We derived it systematically through Lagrangian mechanics and implemented interactive software using concepts from finite element analysis and iterative numerical methods. We have shown that the dynamic NURBS swung surface model can be formulated in two different ways: (i) constructively from two NURBS profile curves and (ii) since swung surfaces are NURBS surfaces, by applying a nonlinear constraint to a D-NURBS surface. Swung surfaces have $O(m+n)$ degrees of freedom. In the first formulation the surface inherits, by construction, the degrees of freedom of each generator curve $(O(m)+O(n))$. In the second formulation, the nonlinear constraint eliminates most of the $O(mn)$ degrees of freedom of a general D-NURBS surface to produce the swung form, but the surface can revert to its general form if the constraint is removed.

The physics-based model responds to applied simulated forces with natural and predictable dynamics with the geometric parameters, including the control points and weights, varying automatically. Designers can employ force-based "tools" to perform direct manipulation and interactive sculpting. Additional control over the shape is available through the modification of physical parameters. Elastic energy functionals allow the qualitative imposition of fairness criteria through quantitative means. Linear or nonlinear constraints may be imposed either as hard constraints that must not be violated, or as soft constraints to be satisfied approximately in the form of simple forces. In particular, dynamic NURBS swung surfaces may be fit to very sparse, unstructured data. Energy-based shape optimization for surface design and fairing is an automatic consequence of the dynamic model achieving static equilibrium subject to data constraints.

Our prototype interactive modeling system provides a promising approach to advanced surface design problems such as blending and cross-section design. It demonstrates the flexibility of dynamic NURBS swung surfaces in a variety of applications, including constraint-based optimization for surface design and fairing, automatic settings of weights in surface fitting, and interactive sculpting through applied forces. Our results indicate that these non-fixed-topology dynamic surfaces offer broad geometric coverage during design. Since our models are built on the industry-standard NURBS geometric substrate, designers working with them can continue to make use of the existing array of geometric design toolkits.

A Structure of Stiffness Matrix

Clearly

$$\mathbf{L}_u = \left[\begin{array}{ccc} \mathbf{A}_u \mathbf{c}_2 & \mathbf{B}_\alpha (\mathbf{J}_1)_u & (\mathbf{A}_\alpha)_u \mathbf{J}_2 \end{array} \right], \ \mathbf{L}_v = \left[\begin{array}{ccc} \mathbf{A}(\mathbf{c}_2)_v & (\mathbf{B}_\alpha)_v \mathbf{J}_1 & \mathbf{A}_\alpha (\mathbf{J}_2)_v \end{array} \right].$$

In addition, since $\mathbf{J}_1$ is not a function of v, we have

$$(\mathbf{H}_4)_u = \left[\begin{array}{ccc} \mathbf{0} & \mathbf{C}(\mathbf{J}_1)_u & \mathbf{0} \end{array} \right], \ (\mathbf{H}_4)_v = \mathbf{0}.$$

We decompose $\mathbf{K}$ into two matrices. Let

$$\begin{aligned} \mathbf{K}_1 &= \frac{1}{3} \iint (\alpha_{1,1} \mathbf{L}_u^\top \mathbf{L}_u + \alpha_{2,2} \mathbf{L}_v^\top \mathbf{L}_v + \beta_{1,1} \mathbf{L}_{uu}^\top \mathbf{L}_{uu} + \\ &\qquad \beta_{1,2} \mathbf{L}_{uv}^\top \mathbf{L}_{uv} + \beta_{2,2} \mathbf{L}_{vv}^\top \mathbf{L}_{vv}) \, du \, dv \end{aligned} \tag{24}$$

and

$$\begin{aligned} \mathbf{K}_2 &= \frac{2}{3} \iint (\alpha_{1,1} \mathbf{L}_u^\top (\mathbf{H}_4)_u + \alpha_{2,2} \mathbf{L}_v^\top (\mathbf{H}_4)_v + \beta_{1,1} \mathbf{L}_{uu}^\top (\mathbf{H}_4)_{uu} + \\ &\qquad \beta_{1,2} \mathbf{L}_{uv}^\top (\mathbf{H}_4)_{uv} + \beta_{2,2} \mathbf{L}_{vv}^\top (\mathbf{H}_4)_{vv}) \, du \, dv \end{aligned} \tag{25}$$

So, in view of (10), it is easy to verify

$$\mathbf{K}\mathbf{p} = (\mathbf{K}_1 + \mathbf{K}_2)\mathbf{p} \tag{26}$$

To examine the structure of $\mathbf{K}_1$ and $\mathbf{K}_2$, we consider without loss of generality only the second cross derivative term for $\mathbf{K}_1$. The entry is the integral of $(\beta_{1,2}/3)\mathbf{L}_{uv}^\top \mathbf{L}_{uv}$ where

$$\mathbf{L}_{uv}^\top \mathbf{L}_{uv} = \left[\begin{array}{ccc} \mathbf{U}_{1,1} & \mathbf{U}_{1,2} & \mathbf{U}_{1,3} \\ & \mathbf{U}_{2,2} & \mathbf{U}_{2,3} \\ \textbf{symmetric} & & \mathbf{U}_{3,3} \end{array} \right], \tag{27}$$

where

$\mathbf{U}_{1,1} = (\mathbf{c}_2)_v^\top \mathbf{A}_u^\top \mathbf{A}_u (\mathbf{c}_2)_v = (\mathbf{X}_1 \mathbf{a})_u^2 \left\| (\mathbf{c}_2)_v \right\|^2$,

$\mathbf{U}_{1,2} = (\mathbf{c}_2)_v^\top \mathbf{A}_u^\top (\mathbf{B}_\alpha)_v (\mathbf{J}_1)_u = \alpha \left\| (\mathbf{c}_2)_v \right\|^2 (\mathbf{X}_1 \mathbf{a})_u (\mathbf{X}_1)_u$,

$\mathbf{U}_{1,3} = (\mathbf{c}_2)_v^\top \mathbf{A}_u^\top (\mathbf{A}_\alpha)_u (\mathbf{J}_2)_v = \alpha (\mathbf{X}_1 \mathbf{a})_u^2 (\mathbf{c}_2)_v^\top (\mathbf{J}_2)_v$,

$\mathbf{U}_{2,2} = (\mathbf{J}_1)_u^\top (\mathbf{B}_\alpha)_v^\top (\mathbf{B}_\alpha)_v (\mathbf{J}_1)_u = \alpha^2 \left\| (\mathbf{c}_2)_v \right\|^2 (\mathbf{X}_1)_u^\top (\mathbf{X}_1)_u + (\mathbf{Z}_1)_u^\top (\mathbf{Z}_1)_u$,

$\mathbf{U}_{2,3} = (\mathbf{J}_1)_u^\top (\mathbf{B}_\alpha)_v^\top (\mathbf{A}_\alpha)_u (\mathbf{J}_2)_v = \alpha^2 (\mathbf{X}_1)_u^\top (\mathbf{X}_1 \mathbf{a})_u (\mathbf{c}_2)_v^\top (\mathbf{J}_2)_v$, and

$\mathbf{U}_{3,3} = (\mathbf{J}_2)_v^\top (\mathbf{A}_\alpha)_u^\top (\mathbf{A}_\alpha)_u (\mathbf{J}_2)_v = \alpha^2 (\mathbf{X}_1 \mathbf{a})_u^2 (\mathbf{J}_2)_v^\top (\mathbf{J}_2)_v$.

Next, we discuss $\mathbf{K}_2$. Because $(\mathbf{H}_4)_v = \mathbf{0}$, (25) can be simplified as

$$\mathbf{K}_2 = \frac{2}{3} \iint (\alpha_{1,1} \mathbf{L}_u^\top (\mathbf{H}_4)_u + \beta_{1,1} \mathbf{L}_{uu}^\top (\mathbf{H}_4)_{uu}) \, du \, dv \tag{28}$$

We consider the first derivative entry

$$\mathbf{L}_u^\top (\mathbf{H}_4)_u = \left[\begin{array}{ccc} \mathbf{0} & \mathbf{U}'_{1,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}'_{2,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}'_{3,2} & \mathbf{0} \end{array} \right] \tag{29}$$

Using the foregoing notations, it is easy to verify that

$\mathbf{U}'_{1,2} = \mathbf{c}_2^\top \mathbf{A}_u^\top \mathbf{C} (\mathbf{J}_1)_u = \mathbf{0}$,

$\mathbf{U}'_{2,2} = (\mathbf{J}_1)_u^\top \mathbf{B}_\alpha^\top \mathbf{C} (\mathbf{J}_1)_u = (\mathbf{Z}_1)_u^\top (\mathbf{Z}_1)_u$, and

$\mathbf{U}'_{3,2} = \mathbf{J}_2^\top (\mathbf{A}_\alpha)_u^\top \mathbf{C} (\mathbf{J}_1)_u = \mathbf{0}$

Thus, $\mathbf{K}_2$ is symmetric. Also, $\mathbf{K}_1$ is obviously symmetric. Therefore, $\mathbf{K}$ is symmetric.

Acknowledgement

We gratefully acknowledge the cooperation of Drs. Marc Rioux, Luc Cournoyer, and Pierre Gariepy of the National Research Council of Canada who provided the geometry and texture data of the clay pot. This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario.

References

[1] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, Jan. 1991.

[2] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485–498, 1987.

[3] W. Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, Sept. 1983.

[4] I.D. Faux and M.J. Pratt. *Computational Geometry for Design and Manufacture*. Ellis Horwood, Chichester,UK, 1979.

[5] C. Woodward. Cross-sectional design of B-spline surfaces. *Computers and Graphics*, 11(2):193–201, 1987.

[6] E. Cobb. *Design of Sculptured Surfaces Using the B-spline Representation*. PhD thesis, University of Utah, 1984.

[7] J. Snyder and J. Kajiya. Generative modeling: A symbolic system for geometric modeling. *Computer Graphics*, 26(2):369–378, 1992.

[8] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[9] J. Platt and A. Barr. Constraints methods for flexible models. *Computer Graphics*, 22(4):279–288, 1988.

[10] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics*, 26(2):309–312, 1992.

[11] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[12] G. Celniker and D. Gossard. Deformable curve and surface finite-element for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991.

[13] M.I.G. Bloor and M.J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.

[14] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992.

[15] J.A. Thingvold and E. Cohen. Physical modeling with B-spline surfaces for interactive design and animation. *Computer Graphics*, 24(2):129–137, 1990. Proceedings, 1990 Symposium on Interactive 3D Graphics.

[16] H.P. Moreton and C.H. Sequin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992.

[17] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics* Proceedings, Annual Conference Series, Proc. ACM Siggraph'93 (Anaheim, CA, Aug., 1993), pages 35–44, 1993.

[18] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.

[19] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[20] G. Farin. *Curves and Surfaces for Computer aided Geometric Design: A Practical Guide*. Academic Press, second edition, 1990.

[21] M.E. Mortenson. *Geometric Modeling*. John Wiley and Sons, 1985.

[22] B.R. Gossick. *Hamilton's Principle and Physical Systems*. Academic Press, New York and London, 1967.

[23] H. Kardestuncer. *Finite Element Handbook*. McGraw–Hill, New York, 1987.

[24] W. Press, B. Flanney, S. Teukolsky, and W. Verttering. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1986.

[25] C. de Boor. On calculating with B-Splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

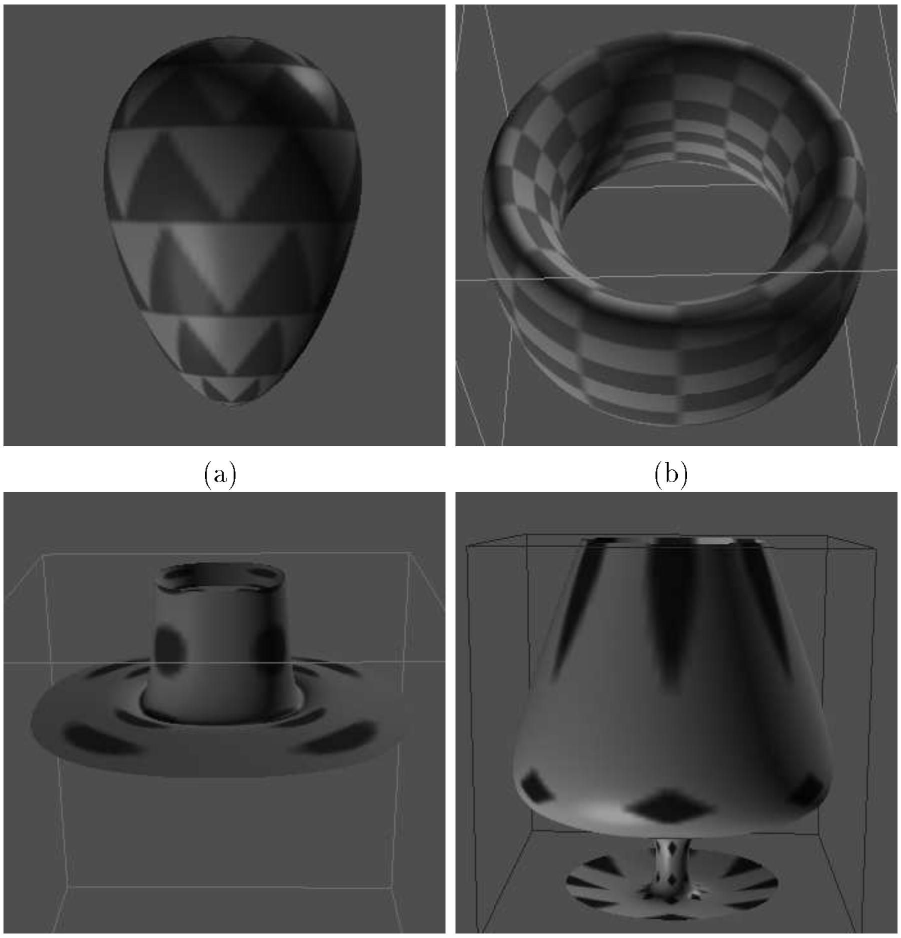
[26] M. Minoux. *Mathematical Programming*. Wiley, New York, 1986.

[27] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, MA, 1986.

[28] J. Platt. A generalization of dynamic constraints. *CVGIP: Graphical Models and Image Processing*, 54(6):516–525, 1992.

[29] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Meth. in Appl. Mech. and Eng.*, 1:1–16, 1972.

[30] G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, PA, 1990.



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

Figure 4: Assorted Dynamic NURBS Swung Surfaces. Open and closed surfaces shown were sculpted interactively from prototype shapes noted in parentheses (a) Egg shape (sphere). (b) Deformed toroid (torus). (c) Hat (open surface). (d) Wine glass (cylinder).

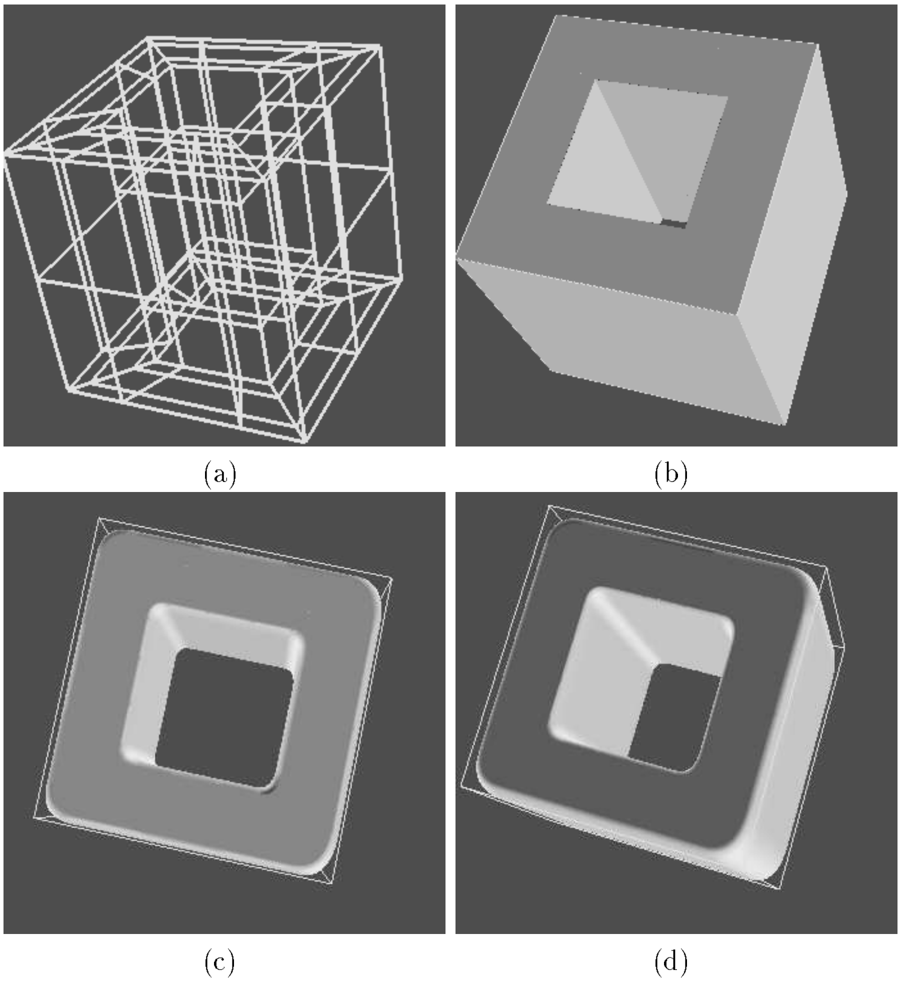
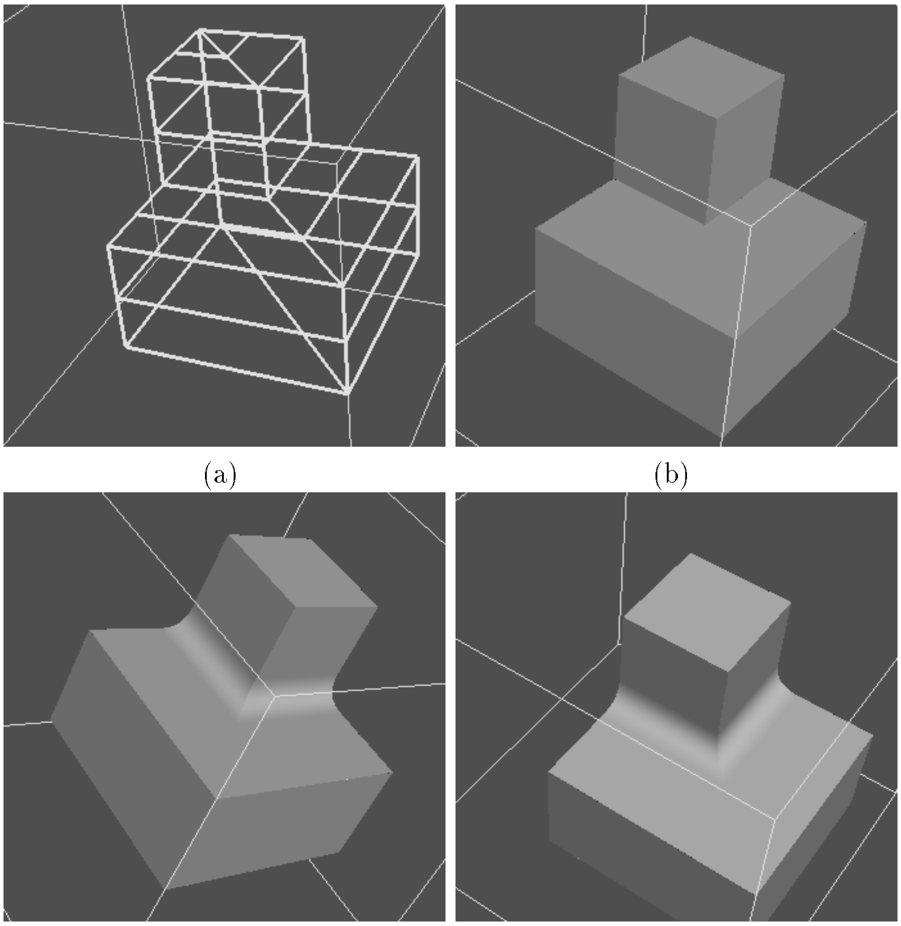
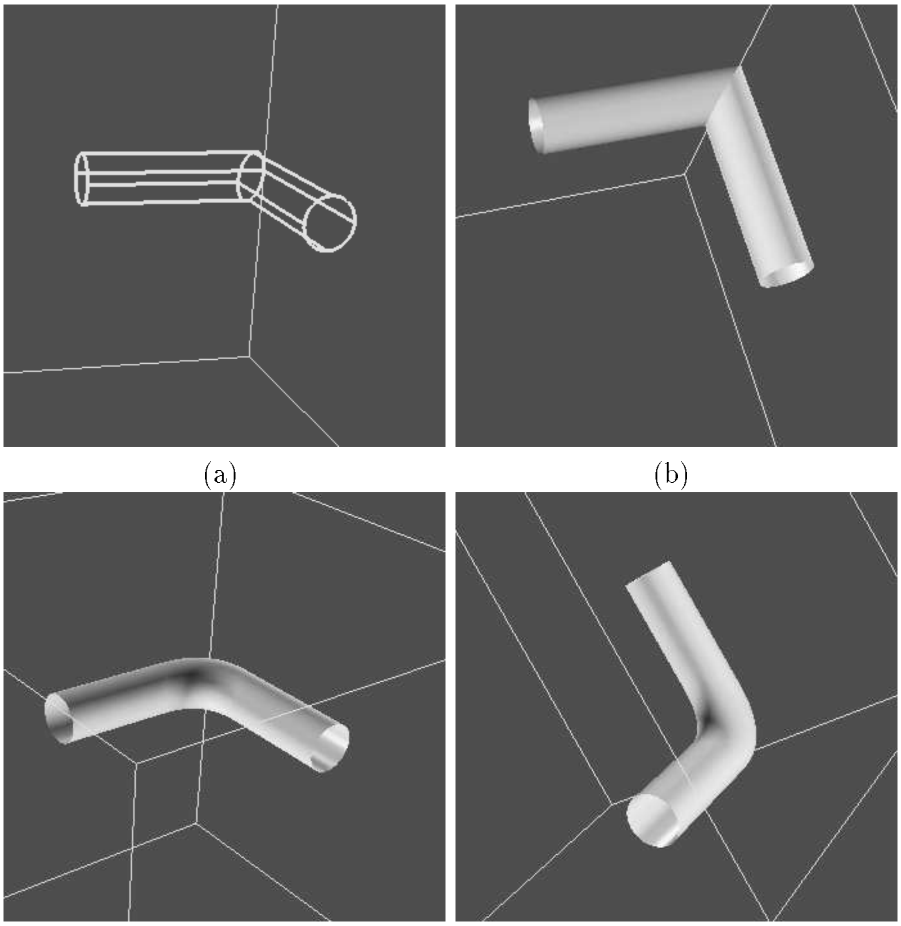


Figure 5: Rounding of polyhedral toroid. (a) Wireframe. (b) Shaded object. (c-d) Final rounded shape.



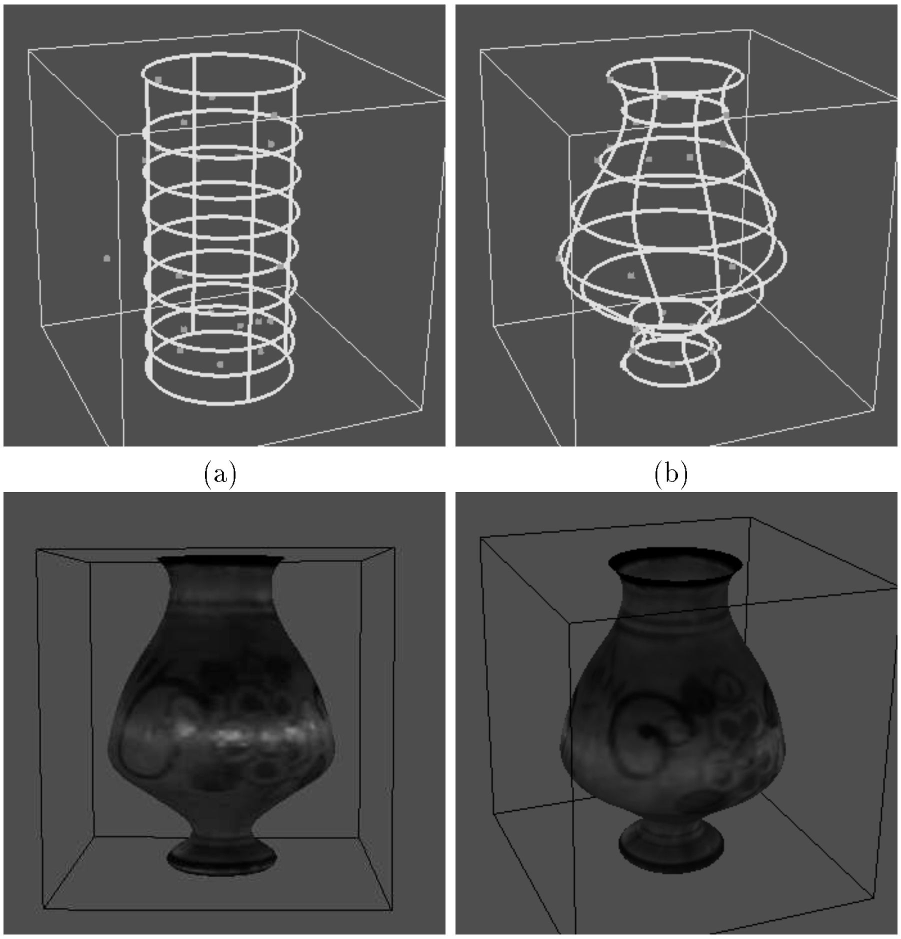
(a) (b) (c) (d)

Figure 6: Rounding of cubical solid. (a) Wireframe. (b) Shaded object. (c-d) Final rounded shape.



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

Figure 7: Surface blending of pipe. (a) Wireframe. (b) Shaded object. (c-d) Final smooth blend.



(a) (b)

(c) (d)

Figure 8: Fitting of 3D laser scanner data. (a) Original cylinder wireframe. (b) Reconstructed pot wireframe. (c-d) Textured pot.

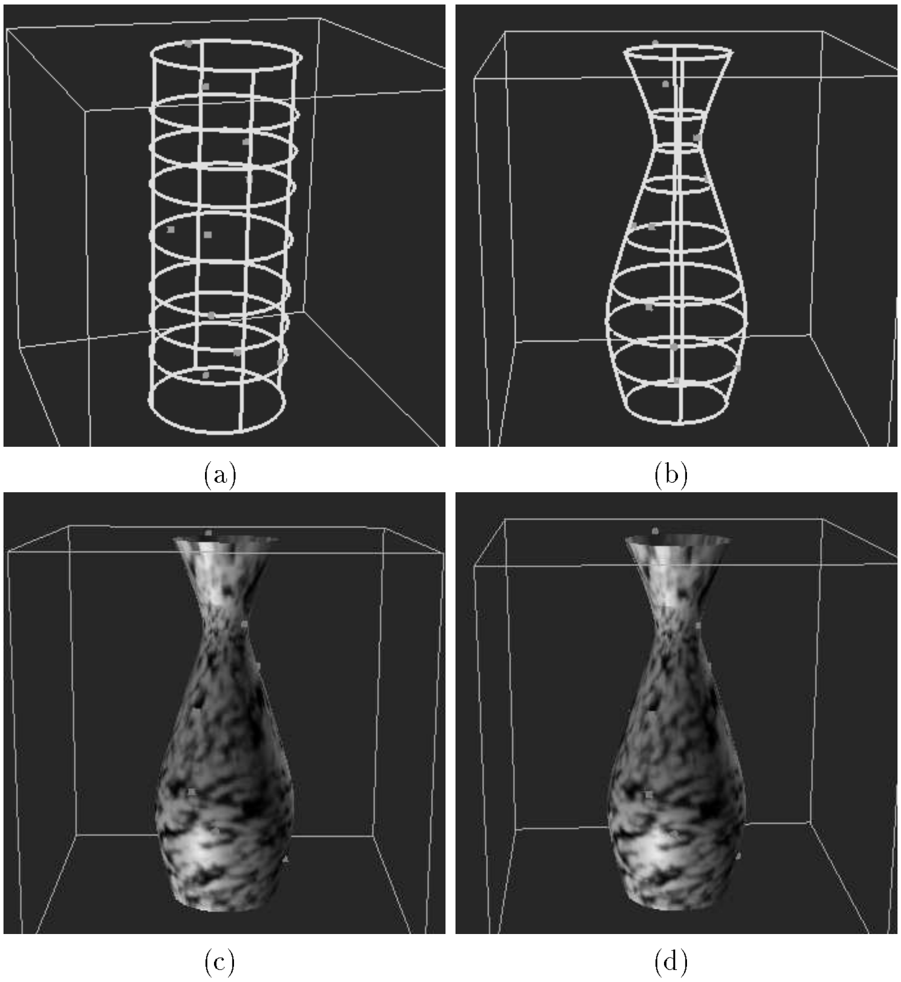


Figure 9: Fitting of synthetic data. (a) cylinder wireframe. (b) Reconstructed vase wireframe. (c-d) Textured vase.

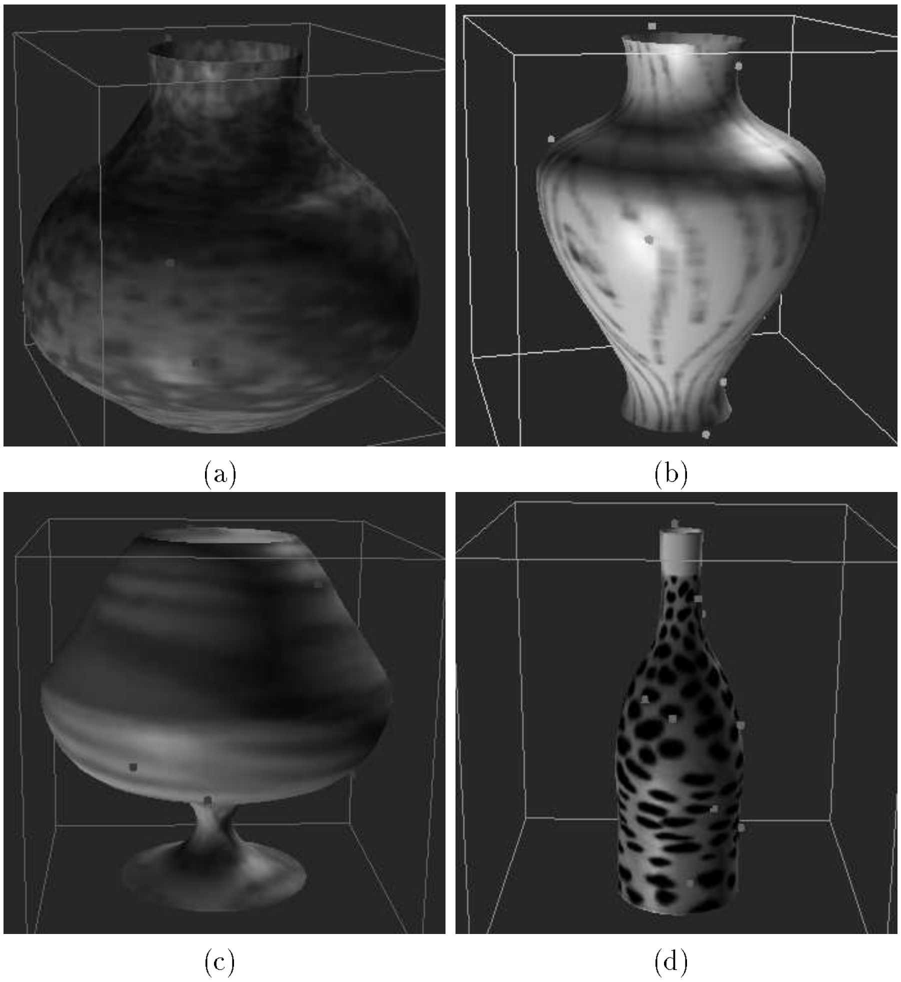


Figure 10: Four fitted shapes. (a) Pot. (b) Vase. (c) Glass. (d) Bottle.