D-NURBS

Hong Qin
Demetri Terzopoulos

Department of Computer Science, University of Toronto
Toronto, Ontario, Canada, M5S 1A4

ABSTRACT

This paper presents dynamic NURBS, or D-NURBS, a physics-based generalization of non-uniform rational B-splines. NURBS have become a *de facto* standard in commercial modeling systems because of their power to represent both free-form and common analytic shapes. Traditionally, however, NURBS have been viewed as purely geometric primitives, which require the designer to interactively adjust many degrees of freedom (DOFs)—control points and associated weights—to achieve desired shapes. The conventional shape modification process can often be clumsy and laborious. D-NURBS are physics-based models that incorporate mass distributions, internal deformation energies, forces, and other physical quantities into the NURBS geometric substrate. Their dynamic behavior, resulting from the numerical integration of a set of nonlinear differential equations, produces physically meaningful, hence intuitive shape variation. Consequently, a modeler can interactively sculpt complex shapes to required specifications not only in the traditional indirect fashion, by adjusting control points, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. We use Lagrangian mechanics to formulate the equations of motion for D-NURBS curves, tensor-product surfaces, swung surfaces, and triangulated surfaces. We apply finite element analysis to reduce these equations to efficient algorithms that can be simulated at interactive rates using standard numerical techniques. We describe a prototype modeling environment based on D-NURBS, and demonstrate that D-NURBS can be effective tools in a wide range of CAGD applications such as shape blending, scattered data fitting, and interactive sculpting.

1 Introduction

In 1975 Versprille[1] proposed the Non-Uniform Rational B-Splines or NURBS. This shape representation for geometric design generalized Riesenfeld's B-splines. NURBS quickly gained popularity and were incorporated into several commercial modeling systems.[2]

The NURBS representation has several attractive properties. It offers a unified mathematical formulation for representing not only free-form curves and surfaces, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution. The most frequently used NURBS design techniques are the specification of a control polygon, and interpolation or approximation of data points to generate the initial shape. For surfaces or solids, cross-sectional design including skinning, sweeping, and swinging operations is also popular. By adjusting the positions of control points, associated weights, and knots of the initial shape, one can design a large variety of shapes using NURBS.[3–5,2,6] Despite modern interactive devices, however, this conventional refinement process can be clumsy and laborious when it comes to designing complex, real-world objects.

In this paper, we propose *Dynamic* NURBS, or D-NURBS. D-NURBS are physics-based models that incorporate mass distributions, internal deformation energies, and other physical quantities into the NURBS geometric substrate. Time is fundamental to the dynamic formulation. The models are governed by dynamic differential equations which, when integrated numerically through time, continuously evolve the control points and weights in response to applied forces. The D-NURBS formulation supports interactive direct manipulation of NURBS objects, which results in physically meaningful hence intuitively predictable motion and shape variation.

Using D-NURBS, a modeler can interactively sculpt complex shapes not merely by kinematic adjustment of control points and weights, but dynamically as well—by applying simulated forces. Additional control over dynamic sculpting stems from the modification of physical parameters such as mass, damping, and elastic properties. Elastic functionals allow the imposition of qualitative "fairness" criteria through quantitative means. Linear or nonlinear constraints may be imposed either as hard constraints that must not to be violated, or as soft constraints to be satisfied approximately. The latter may be interpreted intuitively as simple forces. Optimal shape design results when D-NURBS are allowed to achieve static equilibrium subject to shape constraints. All of these capabilities are subsumed under an elegant formulation grounded in physics.

2 Motivation

NURBS have offered designers extraordinary flexibility when utilized for geometric design. Nevertheless, traditional design methodology does not exploit the full potential of the underlying geometric formulations whose extraordinary flexibility has some drawbacks:

- The designer is faced with the tedium of indirect shape manipulation through a bewildering variety of geometric parameters; i.e., by repositioning control points, adjusting weights, and modifying knot vectors. Despite the recent prevalence of sophisticated 3D interaction devices, the indirect geometric design process remains clumsy and time consuming in general.

- Shape design to required specifications by manual adjustment of available geometric degrees of freedom is often elusive, because relevant design tolerances are typically

shape-oriented and not control point/weight oriented. The geometric "redundancy" of NURBS tends to make geometric shape refinement *ad hoc* and ambiguous; for instance, to adjust a shape should the designer move a control point, or change a weight, or move two control points,...?

- Typical design requirements may be stated in both quantitative and qualitative terms, such as "a fair and pleasing surface which approximates scattered data and interpolates a cross-section curve." Such requirements impose both local and global constraints on shape. The incremental manipulation of local shape parameters to satisfy complex local and global shape constraints is at best cumbersome and often unproductive.

Physics-based modeling provides a means to overcome these drawbacks. Free-form deformable models, which were introduced to computer graphics in 1987[7] and further developed during the past eight years are particularly relevant in the context of modeling with NURBS. Important advantages accrue from the deformable model approach:[8]

- The behavior of the deformable model is governed by physical laws. Through a computational physics simulation, the model responds dynamically to applied simulated forces in a natural and predictable way. Shapes can be sculpted interactively using a variety of force-based "tools."

- The equilibrium state of the dynamic model is characterized by a minimum of the potential energy of the model subject to imposed constraints.[9] It is possible to formulate potential energy functionals that satisfy local and global design criteria, such as curve or surface (piecewise) smoothness, and to impose geometric constraints relevant to shape design.

- The physical model may be built upon a standard geometric foundation, such as free-form parametric curve and surface representations. This means that while shape design may proceed interactively or automatically at the physical level, existing geometric toolkits are concurrently applicable at the geometric level.

Physics-based shape design can free designers from making nonintuitive decisions such as assigning weights to NURBS. In addition, with physics-based direct manipulation, non-expert users are able to concentrate on visual shape variation without necessarily comprehending the underlying mathematical formulation.

3 Background

Dynamic NURBS are motivated by prior research aimed at applying the deformable modeling approach to shape design. Terzopoulos and Fleischer[8] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard[10] developed an interesting prototype system for interactive free-form design based on the finite-element optimization of energy functionals.[8] Bloor and Wilson developed related models using

similar energies and numerical optimization, and they proposed the use of B-splines for this purpose.[11] Subsequently, Celniker and Welch[12] investigated deformable B-splines with linear constraints. Welch and Witkin[13] extended the approach to trimmed hierarchical B-splines.

In prior work,[11–13] deformable B-spline curves and surfaces are designed by imposing shape criteria via the minimization of energy functionals subject to hard or soft geometric constraints. These constraints are imposed through Lagrange multipliers or penalty methods, respectively. The same techniques are applicable to D-NURBS. Compared to deformable B-splines, however, D-NURBS are capable of representing a wider variety of free-form shapes, as well as standard analytic shapes. Previous models solve static equilibrium problems, or involve simple linear dynamics with diagonal (arbitrarily lumped) mass and damping matrices.[12]

D-NURBS are a more sophisticated dynamic model derived through the systematic use of Lagrangian mechanics and finite element analysis without resorting to any of the *ad hoc* assumptions of prior schemes. D-NURBS control points and associated weights are generalized coordinates in the Lagrangian equations of motion. From a physics-based modeling point of view, the existence of weights makes the NURBS geometry substantially more challenging than B-spline geometry. Since the NURBS rational basis functions are functionally dependent on the weights, D-NURBS dynamics are generally nonlinear, and the mass, damping, and stiffness matrices must be recomputed at each simulation time step.[1] Fortunately, this does not preclude interactive performance on current graphics workstations, at least for the size of surface models that appear in our demonstrations. Because our dynamic models allow fully continuous mass and damping distributions, we obtain banded mass and damping matrices. We apply numerical quadrature to the underlying NURBS basis functions to compute efficiently the integral expressions for the matrix entries.

4 Formulation of D-NURBS

This section formulates the physics-based D-NURBS model. The shape parameters of geometric NURBS play the role of generalized (physical) coordinates in dynamic NURBS. We introduce time, mass, and deformation energy into the standard NURBS formulation and employ Lagrangian dynamics to arrive at the system of nonlinear ordinary differential equations that govern the shape and motion of D-NURBS. In particular, we formulate four different varieties: D-NURBS curves, tensor-product D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces.

4.1 D-NURBS Curves

NURBS generalize the non-rational parametric form. They inherit many of the properties of non-rational B-splines, such as the strong convex hull property, variation diminishing

[1] Note, however, that for static weights, the matrices become time invariant and the computational cost is reduced significantly.

property, local support, and invariance under standard geometric transformations. Moreover, they have some additional properties. NURBS can be used to satisfy different smoothness requirements. They include weights as extra degrees of freedom which influence local shape. Most importantly, NURBS offer a common mathematical framework for implicit and parametric polynomial forms. In principle, they can represent analytic functions such as conics and quadrics precisely, as well as free-form shapes.

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 basis functions,[14] $\mathbf{p}_i(t)$ 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: $t_0 \leq t_1 \leq \ldots \leq t_{n+k}$. 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

$$\mathbf{p}(t) = \begin{bmatrix} \mathbf{p}_0^\top & w_0 & \cdots & \mathbf{p}_n^\top & w_n \end{bmatrix}^\top,$$

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 the $3 \times 4(n+1)$ matrix

$$\mathbf{J} = \begin{bmatrix} \cdots & \begin{bmatrix} \frac{\partial \mathbf{c}_x}{\partial \mathbf{p}_{i,x}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{c}_y}{\partial \mathbf{p}_{i,y}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{c}_z}{\partial \mathbf{p}_{i,z}} \end{bmatrix} & \frac{\partial \mathbf{c}}{\partial w_i} & \cdots \end{bmatrix} \tag{3}$$

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 subscripts x, y, and z denote the components 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 elsewhere.[15]

4.2 Tensor-Product D-NURBS Surfaces

In analogy to the kinematic curve of (1), a tensor-product D-NURBS surface

$$\mathbf{s}(u,v,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{p}_{i,j}(t)w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}. \tag{5}$$

generalizes the geometric NURBS surface. The $(m+1)(n+1)$ control points $\mathbf{p}_{i,j}(t)$ and weights $w_{i,j}(t)$, which are functions of time, comprise the D-NURBS generalized coordinates. Assuming basis functions along the two parametric axes of degree $k-1$ and $l-1$, respectively, the number of knots is $(m+k+1)(n+l+1)$. The non-decreasing knot sequence is $t_0 \le t_1 \le \ldots \le t_{m+k}$ along the u-axis and $s_0 \le s_1 \le \ldots \le s_{n+l}$ along the v-axis. The parametric domain is $t_{k-1} \le u \le t_{m+1}$ and $s_{l-1} \le v \le s_{n+1}$. If the end knots have multiplicity k and l in the u and v axis respectively, the surface patch will interpolate the four corners of the boundary control points.

We concatenate these $N = 4(m+1)(n+1)$ coordinates into the vector:

$$\mathbf{p}(t) = \begin{bmatrix} \mathbf{p}_{0,0}^{\top} & w_{0,0} & \cdots & \mathbf{p}_{i,j}^{\top} & w_{i,j} & \cdots & \mathbf{p}_{m,n}^{\top} & w_{m,n} \end{bmatrix}^{\top}.$$

Two subscripts are now associated with the generalized coordinates, reflecting the surface parameters u and v. For concreteness, we order the components in these vectors such that the second subscript varies faster than the first, although this convention does not affect the derived results.

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

$$\dot{\mathbf{s}}(u,v,\mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \qquad \mathbf{s}(u,v,\mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{6}$$

where $\mathbf{J}(u,v,\mathbf{p})$ is the $3 \times N$ Jacobian matrix of the D-NURBS surface with respect to $\mathbf{p}$. However, the contents of the Jacobian $\mathbf{J}$ differ from those in the curve case. To arrive at an explicit expression for $\mathbf{J}$, let $\mathbf{B}_{i,j}(u,v,\mathbf{p})$, for $i = 0,\ldots,m$, and $j = 0,\ldots,n$, be a 3×3 diagonal matrix whose entries are

$$N_{i,j}(u,v,\mathbf{p}) = \frac{\partial \mathbf{s}}{\partial \mathbf{p}_{i,j}} = \frac{w_{i,j}B_{i,k}(u)B_{j,l}(v)}{\sum_{c=0}^{m}\sum_{d=0}^{n}w_{c,d}B_{c,k}(u)B_{d,l}(v)}$$

and let the 3-vector

$$\mathbf{w}_{i,j}(u,v,\mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{i,j}} = \frac{\sum_{c=0}^{m}\sum_{d=0}^{n}(\mathbf{p}_{i,j}-\mathbf{p}_{c,d})w_{c,d}B_{c,k}(u)B_{d,l}(v)B_{i,k}(u)B_{j,l}(v)}{(\sum_{c=0}^{m}\sum_{d=0}^{n}w_{c,d}B_{c,k}(u)B_{d,l}(v))^2}.$$

Hence,

$$\mathbf{J}(u,v,\mathbf{p}) = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{w}_{0,0} & \cdots & \mathbf{B}_{m,n} & \mathbf{w}_{m,n} \end{bmatrix}.$$

Note that $\mathbf{J}$ is now a $3 \times 4(m+1)(n+1)$ matrix.

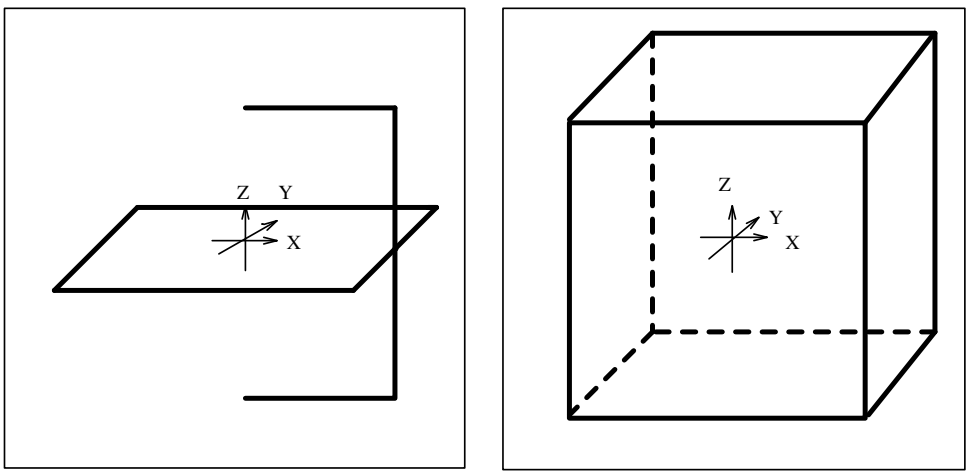

Figure 1. Construction of a cubical NURBS swung surface. (a) NURBS profile curve on x-z plane, NURBS trajectory curve on x-y plane. (b) Cube surface wireframe.

4.3 Swung D-NURBS Surfaces

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.[16] Woodward[17] 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. Piegl[2] 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. 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. 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 include some form of swinging (or sweeping) among their repertoire of techniques.[18]

Geometrically, a dynamic swung surface is generated from two planar kinematic NURBS profile curves through the swinging operation[2] (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

$$\mathbf{s}(u, v, t) = \left[\begin{array}{ccc} \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,x} & \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,y} & \mathbf{c}_{1,z} \end{array} \right]^{\mathsf{T}} \tag{7}$$

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

$$\mathbf{p} = \begin{bmatrix} \alpha & \mathbf{a}^\top & \mathbf{b}^\top \end{bmatrix}^\top.$$

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.

The velocity of the swung surface is

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{L}\dot{\mathbf{p}} \tag{8}$$

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 $\partial\mathbf{s}/\partial\mathbf{b}$. The expression of the $3 \times M$ matrix $\mathbf{L}$ can be explicitly formulated.[19] Unlike $\mathbf{J}$ in (4), $\mathbf{L}$ cannot serve as the basis function matrix of the swung surface. Instead, we have

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

where $\mathbf{H}$ is the $3 \times M$ basis function matrix.[19]

4.4 Triangular D-NURBS Surfaces

The main drawback of tensor-product NURBS is that the surface patches are rectangular. Consequently, the designer is forced to model multisided irregular shapes using degenerate patches with deteriorated inter-patch continuity. Thus, the associated smoothness constraints increase the complexity of the design task in general. In contrast, triangular B-splines[20] and NURBS can represent complex non-rectangular shapes over arbitrary triangulated domains with low degree piecewise polynomials that nonetheless maintain relatively high-order continuity. They can express smooth non-rectangular shapes without degeneracy. They can also model discontinuities by varying the knot distribution.

Let $T = \{\Delta(\mathbf{i}) = [\mathbf{r}, \mathbf{s}, \mathbf{t}] | \mathbf{i} = (i_0, i_1, i_2) \in Z_+^3\}$ be an arbitrary triangulation of the planar parametric domain, where i_0, i_1, and i_2 denote indices of $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$ in the vertex array of the triangulation, respectively. For each vertex $\mathbf{v}$ in the triangulated domain, we associate a knot sequence (also called a cloud of knots) $[\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n]$ (which are inside the shaded circles in Fig. 2). Next, we define a convex hull

$$V_{\mathbf{i},\beta} = \{\mathbf{r}_0, \ldots, \mathbf{r}_{\beta_0}, \mathbf{s}_0, \ldots, \mathbf{s}_{\beta_1}, \mathbf{t}_0, \ldots, \mathbf{t}_{\beta_2}\},$$

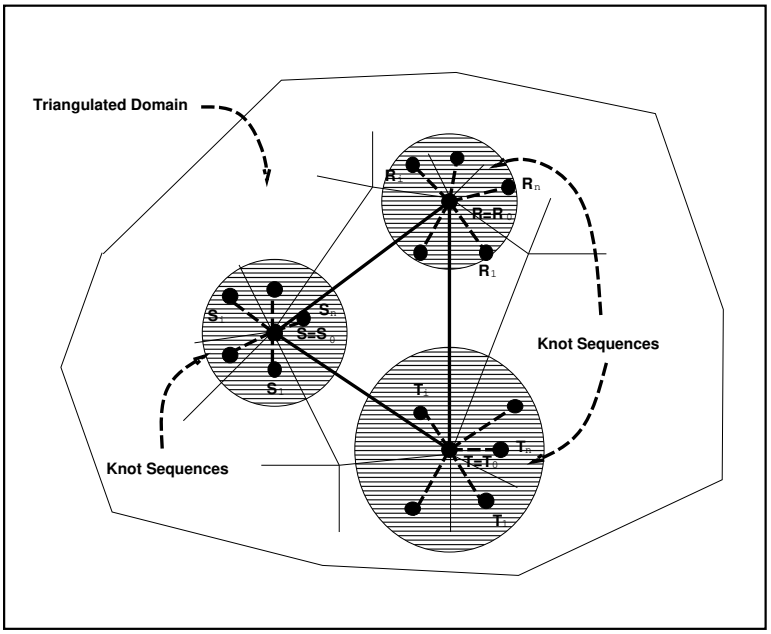

Figure 2. Knot vectors associated with each triangle in the domain triangulation.

where subscript $\mathbf{i}$ is a triangle index, and $\beta = (\beta_0, \beta_1, \beta_2)$ is a triplet such that $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$. The bivariate simplex spline $M(\mathbf{u}|V_{\mathbf{i},\beta})$ with degree n over $V_{\mathbf{i},\beta}$ can be defined recursively (the details are found elsewhere[20]), where $\mathbf{u} = (u, v)$ defines the triangulated parametric domain of the surface. We then define a bivariate B-spline basis function as

$$N_{\mathbf{i},\beta}(\mathbf{u}) = d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2}) M(\mathbf{u}|V_{\mathbf{i},\beta}), \tag{10}$$

where $d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$ is twice the area of $\Delta(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$. Like the ordinary tensor-product D-NURBS, we define triangular D-NURBS as the combination of a set of piecewise rational functions by explicitly incorporating time and physical behavior. The surface is a function of both the parametric variable $\mathbf{u}$ and time t:

$$\mathbf{s}(\mathbf{u}, t) = \frac{\sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta}(t) w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{i}} \sum_{|\beta|=n} w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}. \tag{11}$$

We define the vector of generalized coordinates (control points) $\mathbf{p}_{\mathbf{i},\beta}$ and (weights) $w_{\mathbf{i},\beta}$ as

$$\mathbf{p} = [\ldots, \mathbf{p}_{\mathbf{i},\beta}^{\top}, w_{\mathbf{i},\beta}, \ldots]^{\top}.$$

We then express (11) as $\mathbf{s}(\mathbf{u}, \mathbf{p})$ in order to emphasize its dependence on $\mathbf{p}$ whose components are functions of time.

Thus, the velocity of the triangular D-NURBS is

$$\dot{\mathbf{s}}(\mathbf{u}, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{12}$$

where the overstruck dot denotes a time derivative and the Jacobian matrix $\mathbf{J}(\mathbf{u},\mathbf{p})$ is the concatenation of the vectors $\partial \mathbf{s}/\partial \mathbf{p}_{\mathbf{i},\beta}$ and $\partial \mathbf{s}/\partial w_{\mathbf{i},\beta}$. Assuming m triangles in the parametric domain, β traverses $k = (n+2)!/(n!2!)$ possible triplets whose components sum to n. Because $\mathbf{s}$ is a 3-vector and $\mathbf{p}$ is an $M = 4mk$ dimensional vector, $\mathbf{J}$ is a $3 \times M$ matrix, which may be written as

$$\mathbf{J} = \begin{bmatrix} \ldots, \begin{bmatrix} R_{\mathbf{i},\beta} & 0 & 0 \\ 0 & R_{\mathbf{i},\beta} & 0 \\ 0 & 0 & R_{\mathbf{i},\beta} \end{bmatrix}, \mathbf{w}_{\mathbf{i},\beta}, \ldots \end{bmatrix} \tag{13}$$

where

$$R_{\mathbf{i},\beta}(\mathbf{u},\mathbf{p}) = \frac{\partial \mathbf{s}_x}{\partial \mathbf{p}_{\mathbf{i},\beta,x}} = \frac{\partial \mathbf{s}_y}{\partial \mathbf{p}_{\mathbf{i},\beta,y}} = \frac{\partial \mathbf{s}_z}{\partial \mathbf{p}_{\mathbf{i},\beta,z}} = \frac{w_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{j}} \sum_{|\alpha|=n} w_{\mathbf{j},\alpha} N_{\mathbf{j},\alpha}(\mathbf{u})}$$

and

$$\mathbf{w}_{\mathbf{i},\beta}(\mathbf{u},\mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{\mathbf{i},\beta}} = \frac{(\mathbf{p}_{\mathbf{i},\beta} - \mathbf{s}) N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{j}} \sum_{|\alpha|=n} w_{\mathbf{j},\alpha} N_{\mathbf{j},\alpha}(\mathbf{u})}$$

The subscripts x, y, and z denote derivatives of the components of a 3-vector. Moreover, we can express the surface as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{s}(\mathbf{u},\mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{14}$$

The proof of (14) is the same as that for the tensor-product D-NURBS.[15]

4.5 D-NURBS Equations of Motion

The equations of motion of our D-NURBS are derived from the work-energy version of Lagrangian dynamics.[21] Applying the Lagrangian formulation to D-NURBS curves, tensor-product surfaces, swung surfaces, and triangulated surfaces, we obtain the second-order nonlinear equations of motion

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \tag{15}$$

where the mass matrix $\mathbf{M}(\mathbf{p})$, the damping matrix $\mathbf{D}(\mathbf{p})$, and the stiffness matrix $\mathbf{K}(\mathbf{p})$ can all be formulated explicitly.[15,19,22] The $N \times N$ mass and damping matrices are are

$$\mathbf{M}(\mathbf{p}) = \iint \mu \mathbf{J}^\top \mathbf{J}\, du\, dv; \qquad \mathbf{D}(\mathbf{p}) = \iint \gamma \mathbf{J}^\top \mathbf{J}\, du\, dv \tag{16}$$

where $\mu(u,v)$ is the prescribed mass density function over the parametric domain of the surface and $\gamma(u,v)$ is the prescribed damping density function. To define an elastic potential energy for the surface, we adopt the *thin-plate under tension* energy model.[9,10,13,23,15] This yields the $N \times N$ stiffness matrix

$$\mathbf{K}(\mathbf{p}) = \iint \left(\alpha_{1,1} \mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2} \mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1} \mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2} \mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2} \mathbf{J}_{vv}^\top \mathbf{J}_{vv} \right) du\, dv, \tag{17}$$

where the subscripts on $\mathbf{J}$ denote parametric partial derivatives. 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.[7,24] The generalized force $\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{J}^\top \mathbf{f}(u,v,t)\,du\,dv$ is obtained through the principle of virtual work[21] done by the applied force distribution $\mathbf{f}(u,v,t)$. Because of the geometric nonlinearity, generalized inertial forces $\mathbf{g}_p(\mathbf{p})$ are also associated with the models (see our journal articles for the details[15,19]).

5 Finite Element Implementation

The evolution of $\mathbf{p}$, determined by (15) with time-varying matrices, cannot be solved analytically in general. Instead, we pursue an efficient numerical implementation using finite-element techniques.[25]

Standard finite element codes explicitly assemble the global matrices that appear in the discrete equations of motion.[25] 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.

5.1 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 D-NURBS 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.

5.2 Calculation of Element Matrices

The integral expressions for the mass, damping, and stiffness matrices associated with each element are evaluated numerically using Gaussian quadrature.[26] 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 Ω, the expression for entry m_{ij} of the mass matrix takes the integral form

$$m_{ij} = \int_\Omega \mu(u,v) f_{ij}(u,v)\,du\,dv,$$

where f_{ij} are entries of the mass matrix. Given integers N_g, we can find Gauss weights a_g, and abscissas u_g, v_g in the two parametric directions of Ω such that m_{ij} can be approximated by[26]

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

We apply the de Boor algorithm[27] or the recursive algorithm of multivariate simplex B-splines[28] to evaluate $f_{ij}(u_g, v_g)$. 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 to be integers between 4 and 7. Our experiments indicate that matrices computed in this way lead to stable, convergent solutions. Note that because of the irregular knot distribution for the case of triangular D-NURBS, many f_{ij}'s are zero over the triangular subdomains of Ω. We can further subdivide the subdomains in order to decrease the numerical quadrature error.[22]

5.3 Discrete Dynamics Equations

To integrate (15) 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 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 the time integration formula

$$\left(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K}\right)\mathbf{p}^{(t+\Delta t)} = 2\Delta t^2(\mathbf{f}_p + \mathbf{g}_p) + 4\mathbf{M}\mathbf{p}^{(t)} - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)} \quad (18)$$

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.[26] 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 NURBS surfaces on common graphics workstations.

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 (18) 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 (15) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish.

6 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 D-NURBS upon the standard NURBS geometry, 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.

6.1 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.[7]

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

$$\mathbf{f}(u, v, t) = \iint k(\mathbf{d}_0 - \mathbf{s}(u, v, t))\delta(u - u_0, v - v_0)\, du\, dv, \tag{19}$$

where the δ is the unit delta function. Equation (19) 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.

6.2 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.[29] 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[29] combines the Lagrange multipliers with the simpler penalty method.[30] The Baumgarte stabilization method[31] solves constrained equations of motion through linear feedback control.[32,15] These techniques are appropriate for D-NURBS with nonlinear 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 (15) to a minimal unconstrained set of generalized coordinates. They can then be implemented by applying the same numerical solver on an unconstrained subset of $\mathbf{p}$.[15]

D-NURBS 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. Alternatively, we can 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.

7 Modeling Applications

This section describes our D-NURBS modeling environment and presents several applications relating to solid rounding, optimal surface fitting, and interactive sculpting.

7.1 Interactive Modeling Environment

We have developed a prototype modeling environment based on the tensor-product and swung D-NURBS model. The system is written in C and it currently runs under Iris Explorer 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. To date, our D-NURBS modules implement 3D curve and surface objects with basis function orders of 2, 3, or 4 (i.e., from linear to cubic D-NURBS) with linear geometric constraints.

We have also developed prototype modeling software based on dynamic triangular B-splines which is a special case of triangular D-NURBS by fixing all weights to be unity (an advanced system based upon dynamic triangular NURBS is under construction). We have adopted the data structure, file, and rendering formats of existing geometric triangular B-spline software.[33] To implement the Lagrangian dynamics model on top of this software, we have had to implement a new algorithm for simultaneously evaluating non-zero basis functions and their derivatives up to second order at arbitrary domain points for finite element assembly and dynamic simulation.

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. Linear constraints such as the freezing of control points have been associated with physics-based toolkits in our prototype system. Local geometric constraints can be used to achieve real-time local manipulation for interactive sculpting of complex objects.

7.2 Optimal Surface Fitting

D-NURBS are applicable to the optimal fitting of regular or scattered data.[34] The most general and often most useful case occurs with scattered data, when there are fewer or more

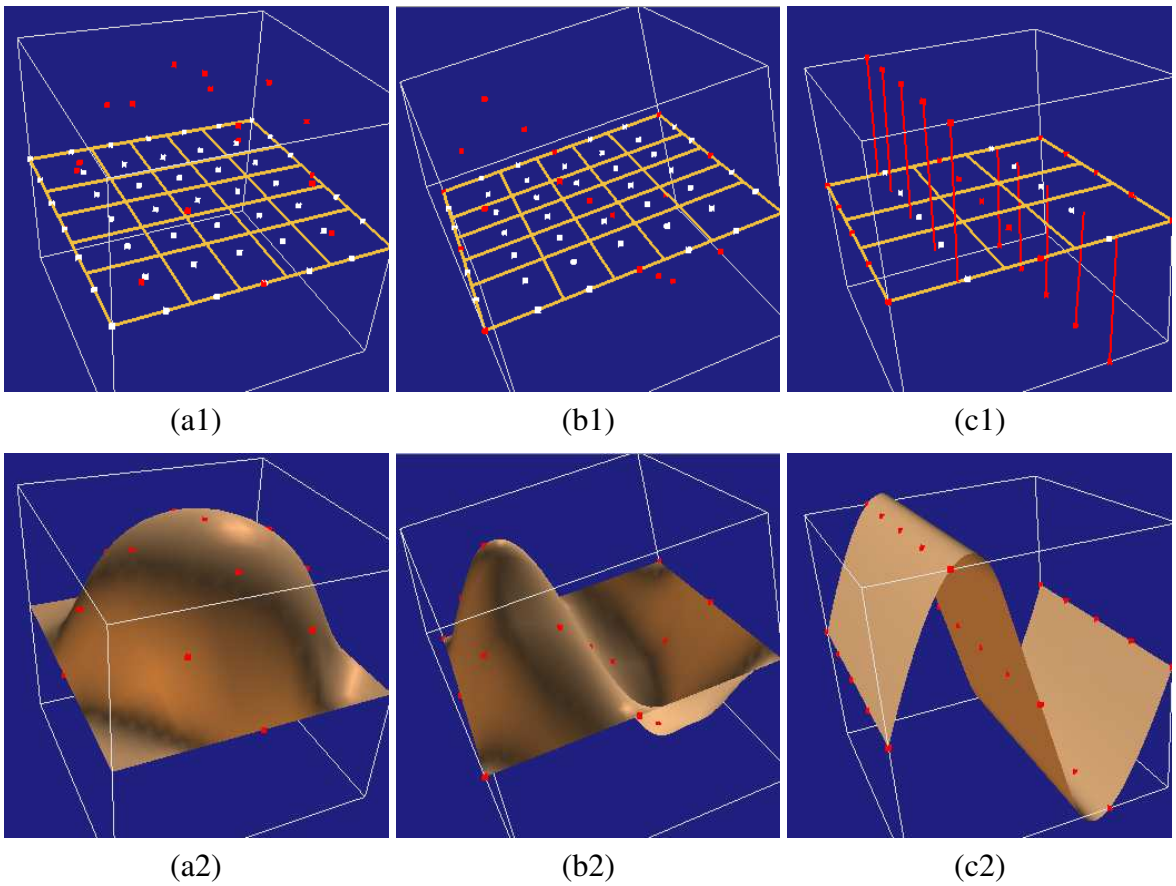

Figure 3. Optimal surface fitting: D-NURBS surfaces fit to sampled data from (a) a hemisphere, (b) a convex/concave surface, (c) a sinusoidal surface. (a–c1) D-NURBS patch outline with control points (white) and data points (red) shown. (a–c2) D-NURBS surface at equilibrium fitted to scattered data points. Red line segments in (c2) represent springs with fixed attachment points on surface.

data points than unknowns—i.e., when the solution is underdetermined or overdetermined by the data. In this case, D-NURBS can yield "optimal" solutions by minimizing the thin-plate under tension deformation energy.[9] The surfaces are optimal in the sense that they provide the smoothest curve or surface (as measured by the deformation energy) which interpolates or approximates the data.

The data point interpolation problem amounts to a linear constraint problem when the weights are fixed, and it is amenable to the constraint techniques presented in Section 6.2. The optimal approximation problem can be approached in physical terms, by coupling the D-NURBS to the data through Hookean spring forces (19). We interpret $\mathbf{d}_0$ in (19) as the data point (generally in $\Re^3$) and (u_0, v_0) as the D-NURBS parametric coordinates associated with the data point (which may be the nearest material point to the data point). The spring constant c determines the closeness of fit to the data point.

We present three examples of surface fitting using tensor-product D-NURBS coupled

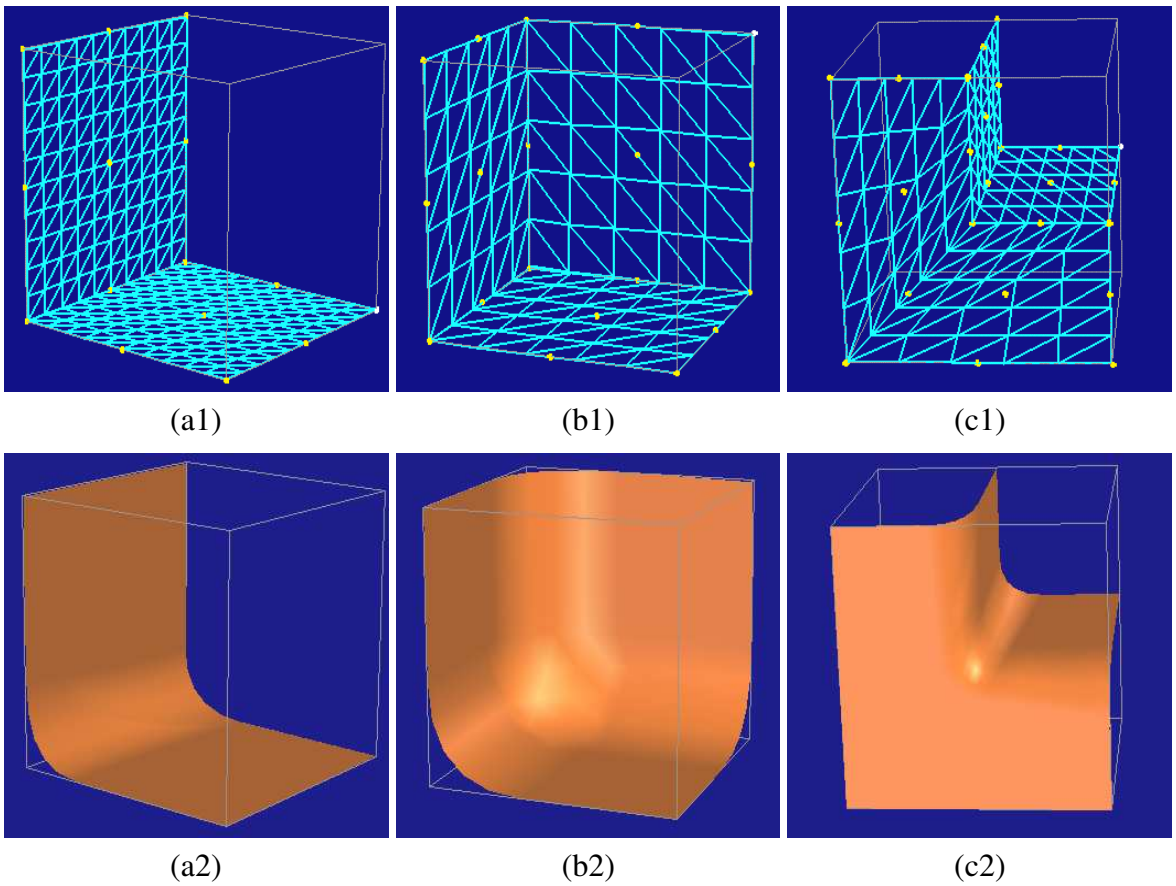

Figure 4. Solid rounding with triangular D-NURBS: Rounding of (a) an edge, (b) a trihedral corner, (c) a bevel joint. (a1-c1) Initial wireframe surfaces. (a2-c2) Final rounded, shaded surfaces.

to data points through spring forces. Fig. 3(a) shows 19 data points sampled from a hemisphere and their interpolation with a quadratic D-NURBS surface with 49 control points. Fig. 3(b) shows 19 data points and the reconstruction of the implied convex/concave surface by a quadratic D-NURBS with 49 control points. The spring forces associated with the data points are applied to the nearest points on the surface. In Fig. 3(c) we reconstruct a wave shape from 25 sample points using springs with fixed attachments to a quadratic tensor-product D-NURBS surface with 25 control points.

7.3 Rounding

The rounding operation is usually attempted geometrically by enforcing continuity requirements on the fillet which interpolates between two or more surfaces. By contrast, the D-NURBS 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 static equilibrium.

Fig. 4(a) demonstrates the rounding of a sharp edge represented by a quadratic triangular

D-NURBS surface with 36 control points. The sharp edge can be represented exactly with multiple control points. By restricting the control polygon to be a continuous net, we reduced the number of control points to 21. The initial wireframe surface is shown in Fig. 4(a1). After initiating the physical simulation, the sharp edges are rounded as the final shape equilibrates into the minimal energy state shown by the shaded surface in Fig. 4(a2).

Fig. 4(b) illustrates the rounding of a trihedral corner of a cube. The corner is represented using a quadratic triangular D-NURBS with 78 control points. The initial wireframe is shown in Fig. 4(b1). The rounding operation is applied in the vicinity of three sharp edges. The sharp edges and corner are rounded with position and normal constraints along the far boundaries of the faces of the shaded surface shown in Fig. 4(b2).

Fig. 4(c) shows a rounding example involving a bevel joint. The bevel joint is a quadratic triangular D-NURBS with 108 control points. The initial right-angle joint and the final rounded surface are shown in Fig. 4(c1–2).

7.4 Interactive Sculpting

In the physics-based modeling approach, not only can designers manipulate the individual degrees of freedom with conventional geometric methods, but they 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. 5(a) illustrates the results of four interactive sculpting sessions using swung D-NURBS surfaces and simple 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. 5(a). A torus whose two profile curves are quadratic with 7 and 7 control points, respectively, has been deformed into the shape in Fig. 5(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 5(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 5(c).

8 Conclusion

We have describe D-NURBS, a dynamic generalization of geometric NURBS. D-NURBS were derived systematically through the application of Lagrangian mechanics and implemented using concepts from finite element analysis and efficient numerical methods. The mathematical development comprised four varieties: D-NURBS curves, tensor-product D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces.

We also presented a new physics-based design paradigm based on D-NURBS which generalizes well established geometric design. This paradigm was the basis of a D-NURBS interactive modeling environment. The physics-based framework furnishes designers not only the standard geometric toolkits but powerful force-based sculpting tools as well.

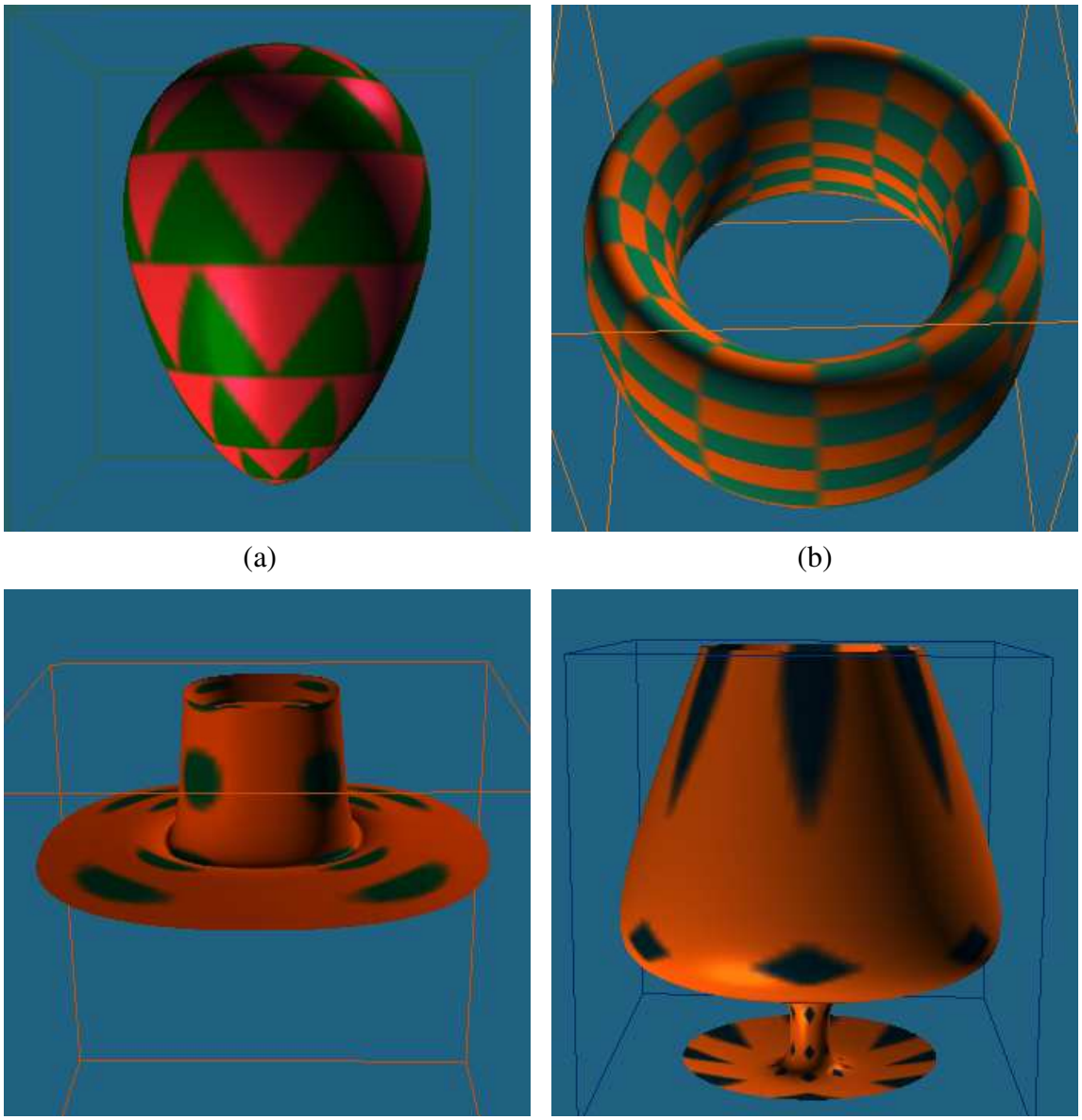

Figure 5. Interactive Sculpting of D-NURBS Swung Surfaces. Open and closed surfaces shown were sculpted interactively from prototype shapes noted in parentheses (a) Egg shape (sphere). (b) Deformed toroid (torus). (c) Hat (open surface). (d) Wine glass (cylinder).

It provides mechanisms for automatically adjusting unknown parameters to support user manipulation and satisfy design requirements.

Since D-NURBS 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. With the advent of high-performance graphics systems, however, the physics-based framework is poised for incorporation into commercial design systems to interactively model and sculpt complex shapes in real-time. Thus, D-NURBS can unify the features of the industry-standard geometry with the many demonstrated conveniences of interaction through physical dynamics.

Acknowledgements

We would like to thank Professor Hans-Peter Seidel for kindly making available the software for triangular B-spline surfaces that he developed with Philip Fong. This research was made possible by grants from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario. DT is a fellow of the Canadian Institute for Advanced Research.

References

[1] K.J. Versprille. *Computer-Aided Design Applications of the Rational B-Spline Approximation form*. PhD thesis, Syracuse University, 1975.

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

[3] G. Farin. Trends in curve and surface design. *Computer-Aided Design*, 21(5):293–296, 1989.

[4] L. Piegl. Modifying the shape of rational B-splines. part 1:curves. *Computer-Aided Design*, 21(8):509–518, 1989.

[5] L. Piegl. Modifying the shape of rational B-splines. part 2:surfaces. *Computer-Aided Design*, 21(9):538–546, 1989.

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

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

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

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

[10] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991. (Proc. ACM Siggraph'91).

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

[12] G. Celniker and W. Welch. Linear constraints for deformable B-spline surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, pages 165–170, 1992.

[13] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992. (Proc. ACM Siggraph'92).

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

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

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

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

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

[19] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based shape design. *Computer Aided Design*, 27(2):111–127, 1995.

[20] W. Dahmen, C. Micchelli, and H.-P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 59(199):97–115, 1992.

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

[22] H. Qin and D. Terzopoulos. Dynamic manipulation of triangular B-splines. In *Proceedings of Third ACM/IEEE Symposium on Solid Modeling and Applications*, Salt Lake City, May 1995. ACM Press. in press.

[23] 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.

[24] H.P. Moreton and C.H. Sequin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992. (Proc. ACM Siggraph'92).

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

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

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

[28] C.A. Micchelli. On a numerically efficient method for computing with multivariate B-splines. In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory*, pages 211–248. Birkhauser, Basel, 1979.

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

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

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

[32] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics*, 26(2):309–312, 1992. (Proc. ACM Siggraph'92).

[33] P. Fong and H.-P. Seidel. An implementation of triangular B-spline surfaces over arbitrary triangulations. *Computer Aided Geometric Design*, 3-4(10):267–275, 1993.

[34] L.L. Schumaker. Fitting surfaces to scattered data. In G.G. Lorentz, C.K. Chui, and L.L. Schumaker, editors, *Approximation Theory II*, pages 203–267. Academic Press, New York, 1976.