D-NURBS: A Physics-Based Framework for Geometric Design

Hong Qin and **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 shapes 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 and setting weights, 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 D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces. We apply finite element analysis to reduce these equations to efficient numerical algorithms computable at interactive rates on common graphics workstations. We implement 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.**

Keywords— **NURBS, Geometric Modeling, Computer-Aided Design, Computer Graphics, Physics-Based Models, Finite Elements, Dynamics.**

I. Introduction

Geometric modeling is crucial to a variety of fields including computer-aided geometric design (CAGD), computer graphics, and scientific visualization. Numerous geometric representations have been developed for a large variety of geometric modeling applications. Among them, non-uniform rational B-splines, or NURBS, [37] have become an industry-standard representation [25]. NURBS are prevalent in commercial modeling systems primarily because they provide a unified mathematical representation for free-form curves and surfaces and for standard analytic shapes such as conics, quadrics, and surfaces of revolution.[1] There are several different types of NURBS representations, including NURBS curves, tensor-product NURBS surfaces, swung NURBS surfaces, and triangular NURBS surfaces. Experienced practitioners can design a large variety of NURBS objects by adjusting the positions of control points, setting the values of associated weights, and modifying the distribution of knots [9], [23], [24], [25], [36]. Despite modern interaction techniques, however, this conventional geometric design process with NURBS can oftentimes be clumsy and laborious.

We propose a physics-based framework for geometric design with NURBS that addresses these problems through a new class of models known as *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 formulation. The models are governed by dynamic differential equations that, when integrated numerically through time, continuously evolve the control points and weights in response to applied forces. We formulate D-NURBS curves, tensor-product D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces.

Within our physical dynamics framework, standard geometric NURBS objects inherit some of the universally familiar behaviors of physical, real-world objects. Thus, physics-based design augments (rather than supersedes) standard geometry and geometric design, offering attractive new advantages. In particular, the elastic energy functionals associated with D-NURBS allow the imposition of global qualitative "fairness" criteria through quantitative means. Linear or nonlinear shape constraints may be imposed either as hard constraints that must not be violated, or as soft constraints to be satisfied approximately. Constraints may be interpreted intuitively as forces and optimal shape design results when D-NURBS are allowed to achieve static equilibrium subject to these forces. More importantly, the D-NURBS formulation supports interactive direct manipulation of NURBS objects, which results in physically meaningful, hence intuitively predictable, deformation and shape variation.

[1] More specifically, 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 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 that 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.

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.

II. The Physics-Based Approach

Although NURBS have offered designers extraordinary flexibility, traditional design methodology does not exploit the full potential of the underlying geometric formulation. Conventional geometric design with NURBS models can be problematic for the following reasons [35]:

- In traditional free-form geometric design, the user is often 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, indirect geometric design of univariate and tensor product splines can be clumsy and laborious when designing complex, real-world objects. It is especially severe for triangular NURBS due to the irregularity of control points and knot vectors. For symmetric surfaces or solids, the labor may be lessened to some degree through the judicious application of cross-sectional design using NURBS swinging operations [25].
- 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. Moreover, a particular shape can often be represented non-uniquely, with different values of knots, control points, and weights. This "geometric redundancy" of NURBS tends to make shape refinement *ad hoc* and ambiguous; it often requires designers to make nonintuitive decisions—for instance, to adjust a shape, should the designer move a control point, change a weight, move two control points, or adjust several weights?
- The design requirements of engineers and stylists can be different. Whereas engineers focus on technical and functional issues, stylists emphasize aesthetically-driven conceptual design. Thus, typical design requirements may be posed in both quantitative and qualitative terms. Therefore, it can be very frustrating to design via the indirect approach, say, a "fair" surface that approximates unorganized 3D data.

To ameliorate the design process for geometric NURBS, we develop a physics-based generalization of the model using Lagrangian mechanics and finite element techniques. Our D-NURBS model unifies the elegant geometric features of NURBS with the many demonstrated conveniences of interaction within a physical dynamics framework. The following are major advantages of physics-based shape design [35]:

- The behavior of D-NURBS 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." Physics-based sculpting is intuitively appealing for shape design and control.
- The equilibrium shape of a D-NURBS object is characterized by a minimum of its potential energy, subject to imposed constraints. 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.
- Since the dynamic model may be built upon the standard geometric NURBS foundation, shape design may proceed interactively or automatically at the physical level, existing geometric toolkits are concurrently applicable at the geometric level (Fig. 1). More importantly, the two types of toolkits are compatible with each other. Designers are free to choose either one or both to achieve design requirements.

Physics-based shape design can free designers from having to make 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. Designers are allowed to interactively sculpt shapes in a natural and predictable way using a variety of force-based tools.

Moreover, geometric design, especially conceptual design, is a time-varying process because designers are often interested in not only the final static equilibrium shape but the intermediate shape variation as well. In contrast to recent variational design approaches [4], [15], time is fundamental to physics-based design. Additional advantages accrue through the use of real-time dynamics:

- An "instantaneous" optimizer (if such a thing existed) can produce some kinematic deformation if it were applied at every interaction step to satisfy constraints. But the motion would be artificial and there would be nothing to prevent sudden, nonsmooth motions (depending on the structure of the constraints), which can be annoying and confusing. By contrast, the dynamic formulation is much more general in that it augments the geometry with time, mass, force, and constraint. Dynamic models produce smooth, natural motions that are familiar and easily controlled.
- Dynamics facilitates interaction, especially direct manipulation and interactive sculpting of complex geometric models for real-time shape variation. The dynamic approach subsumes all of the geometric capabilities in an elegant formulation that grounds shape variation in real-world physics. Despite the fact that incremental optimization may provide a

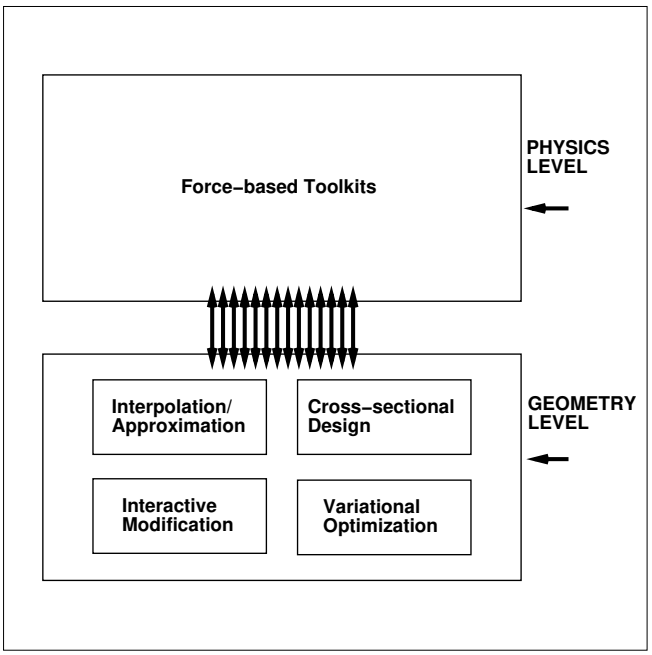


Fig. 1. A Two-Level Physics-Based Design Paradigm.

means of interaction, pure optimization techniques can easily become trapped in the local minima characteristic of non-linear models and/or constraints. In contrast, real-time dynamics can overcome the difficulty of incremental optimization through the incorporation of inertial properties into the model and the interactive use of force-based tools by the designer.

- Practical design processes span conceptual geometric design and the fabrication of mechanical parts. Physics-based modeling techniques and real-time dynamics integrates geometry with physics in a natural and coherent way. The unified formulation is potentially relevant throughout the entire design and manufacturing process.

III. Background

D-NURBS are motivated by prior research aimed at applying the deformable modeling approach to shape design. Terzopoulos and Fleischer demonstrated simple interactive sculpting using viscoelastic and plastic models [33]. Celniker and Gossard developed an interesting prototype system [5] for interactive free-form design based on the finite-element optimization of energy functionals proposed in [33]. The system combines geometric constraints with sculpting operations based on forces and loads to yield fair shapes. However, this approach does not provide interactive mechanisms in dealing with forces and loads. Bloor and Wilson developed related models using similar energies and numerical optimization [3], and in [2] they proposed the use of B-splines for this purpose. Subsequently, Celniker and Welch investigated deformable B-splines with linear constraints [6]. Welch and Witkin extended the approach to trimmed hierarchical B-splines (see also [13]) for interactive modeling of free-form surface with constrained variational optimization [38].

In prior work [2], [6], [38], 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 [35]. 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 [6].

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 [35]. 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.[2] 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 "non-discrete" 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.

IV. 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.

A. D-NURBS Curves

A D-NURBS curve extends the geometric NURBS curve definition by explicitly incorporating time. It can be defined as a function of both the parametric variable

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

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 [10], $\mathbf{p}_i(t)$ 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

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

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 D-NURBS curve 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 a $4(n+1)$ 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 [35].

B. Tensor-Product D-NURBS Surfaces

In analogy to the D-NURBS 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 control points at the four corners of the boundary.

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. By convention, we order the components in these vectors such that the second subscript varies faster than the first.

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

$$\mathbf{w}_{i,j}(u,v,\mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{i,j}} =$$

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

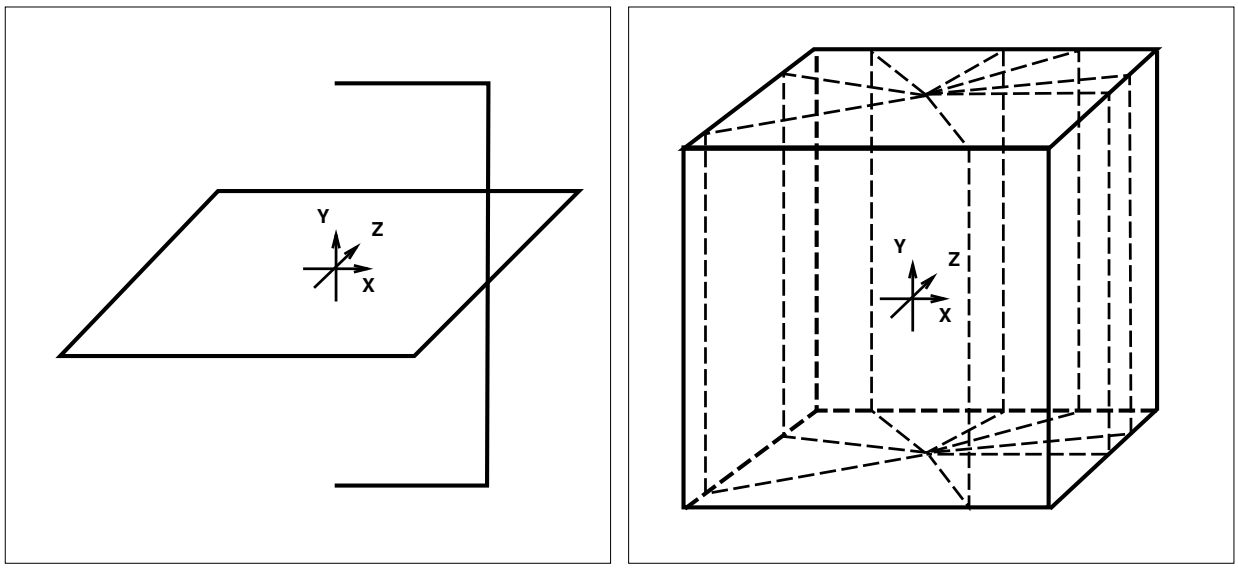
Hence,

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

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

C. 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 [11]. Woodward [39] 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 [25] 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,



(a) (b)

Fig. 2. 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 and a subset of moving profiles.

Fig. 2 illustrates the design of a cubical NURBS swung surface from two NURBS profile curves. Some of the profiles (dashed lines) are shown to illustrate the swinging operation.

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 [31].

Geometrically, a swung D-NURBS surface is generated from two planar D-NURBS profile curves through the swinging operation [25] (Fig. 2). 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

$$\mathbf{s}(u, v, t) = \begin{bmatrix} \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,x} & \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,y} & \mathbf{c}_{1,z} \end{bmatrix}^\top, \quad (7)$$

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 D-NURBS surface is

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

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 [28]. Unlike $\mathbf{J}$ in (4), $\mathbf{L}$ cannot serve as the basis function matrix of the swung surface because of the fact $\mathbf{s}(u, v, \mathbf{p}) \neq \mathbf{L}\mathbf{p}$ [28]. Instead, we have

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

where $\mathbf{H}$ is the $3 \times M$ basis function matrix whose explicit expression can also be found in [28].

D. 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 [7] 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

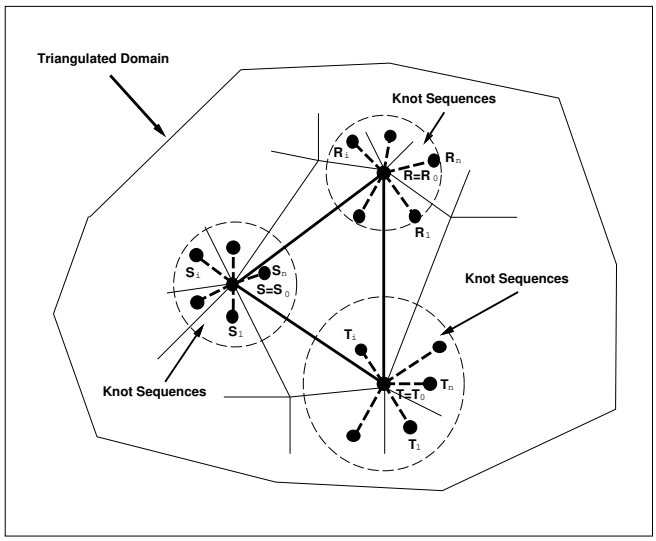


Fig. 3. Knot vectors associated with each triangle in the domain triangulation.

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) $[\mathbf{v} \equiv \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n]$ (which are inside the circles in Fig. 3). Next, we define a convex hull

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

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 [7]), 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 $k = \begin{pmatrix} n+2 \\ 2 \end{pmatrix}$ 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 [35].

E. D-NURBS Equations of Motion

The equations of motion of our dynamic NURBS model are derived from the work-energy version of Lagrangian dynamics [14]. 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 (refer to [35], [28], [29] for the details). The $N \times N$ mass and damping matrices are are

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

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 [32], [5], [38], [16], [35]. This yields the $N \times N$ stiffness matrix

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

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 that control tension and rigidity, respectively, in the two parametric coordinate directions.[3] Other energies are applicable, including the nonquadratic, curvature-based energies [34], [21]. 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 [14] 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 [35], [28]).

V. Implementation of D-NURBS Finite Element

The equations of motion (15) that determine the evolution of $\mathbf{p}$ cannot be solved analytically in general. Instead, we pursue an efficient numerical implementation using finite-element techniques [17].

Standard finite element codes explicitly assemble the global matrices that appear in the discrete equations of motion [17]. 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.

A. Element Data Structures

We consider a D-NURBS curve arc or surface patch defined by consecutive knots in the parametric domain to be a type of finite element. For instance, Fig. 4 illustrates a typical finite element of a cubic triangular B-spline surface, along with its local degrees of freedom.[4]

We define an element data structure which contains the geometric specification of the surface patch (or curve arc) element along with its physical properties, as is illustrated in Fig. 5. A complete D-NURBS curve or surface is then implemented as a data structure that consists of an ordered array of D-NURBS elements with additional

[3] In the case of the D-NURBS curve, there are only two terms and two weighting functions in the potential energy form because of the single spatial parameter u: $U = \frac{1}{2}\int \alpha(u)\frac{\partial \mathbf{c}^\top}{\partial u}\frac{\partial \mathbf{c}}{\partial u} + \beta(u)\frac{\partial^2 \mathbf{c}^\top}{\partial u^2}\frac{\partial^2 \mathbf{c}}{\partial u^2}\, du$.

[4] The degrees of freedom of this element consist of all control points whose basis functions are non-zero over the current triangle in the parametric domain. Because of the irregular knot distribution of triangular NURBS, we can not display all the degrees of freedom of this element; only 10 indexed control points are shown in Fig. 4.

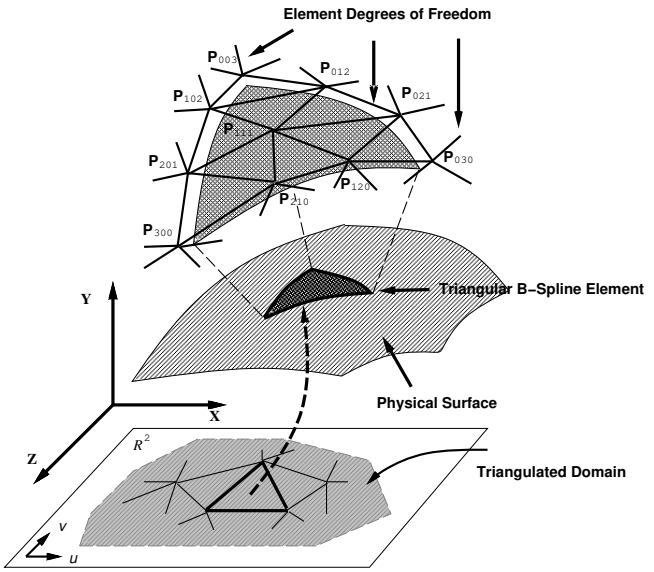


Fig. 4. One finite element and its degrees of freedom of a triangular B-Spline surface.

information. The element structure includes pointers to appropriate components of the global vector $\mathbf{p}$ (control points and weights). Note that neighboring elements will share some generalized coordinates (see Fig. 5). 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.

B. Calculation of Element Matrices

The integral expressions for the mass, damping, and stiffness matrices associated with each element are evaluated numerically using Gaussian quadrature [27]. 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, according to (16),

$$f_{ij}(u,v) = \mathbf{j}_i^\top \mathbf{j}_j.$$

Here, the $\mathbf{j}_i$ and $\mathbf{j}_j$ are the columns of the Jacobian matrix for the D-NURBS surface element. 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 [27]

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

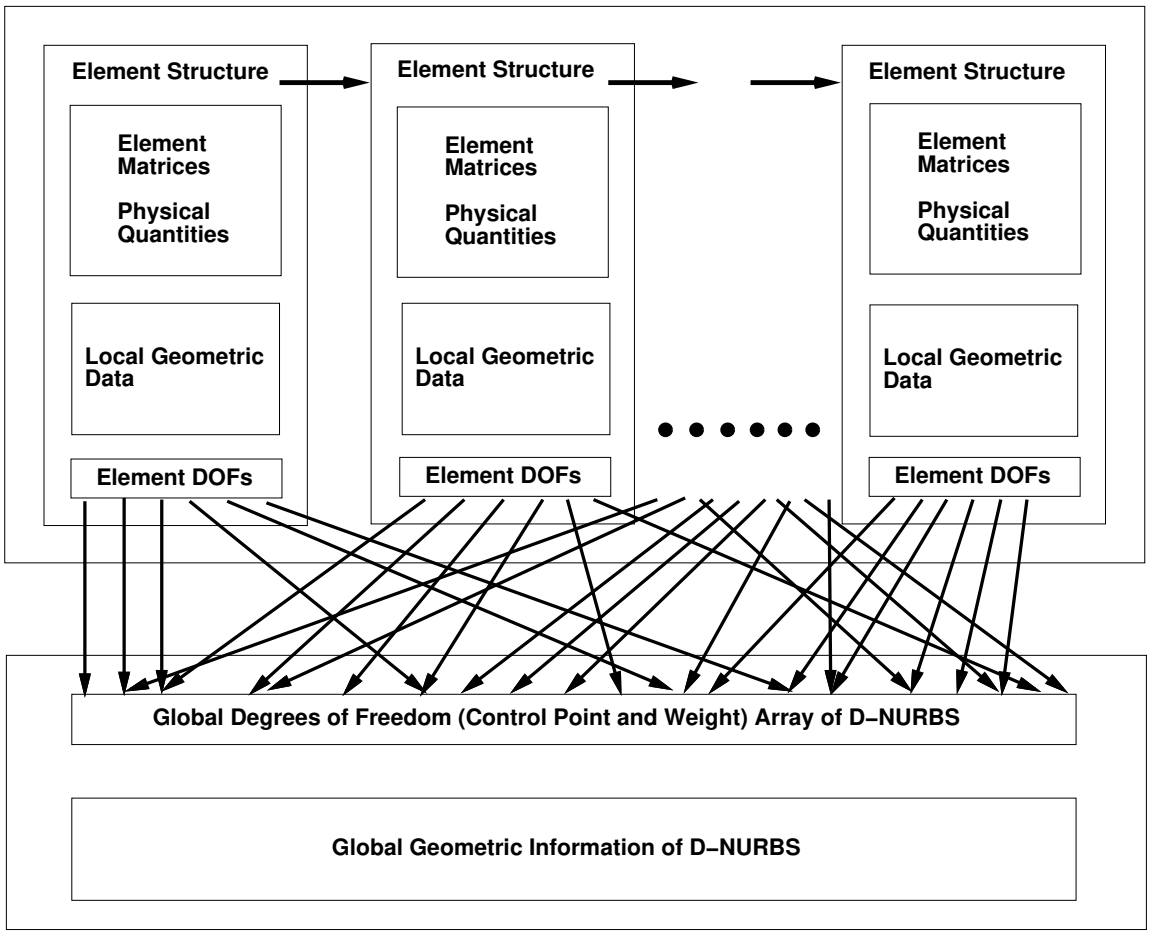


Fig. 5. Element data structure of D-NURBS.

We apply the de Boor algorithm [8] or the recursive algorithm of multivariate simplex B-splines [19] to evaluate $f_{ij}(u_g, v_g)$.[5] 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 [29].

C. 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 [27]. 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

[5] The entries of the D-NURBS curve element mass matrix are $m_{ij} = \int_{u_0}^{u_1} \mu(u) f_{ij}(u)\, du$, where $f_{ij}(u) = \mathbf{j}_i^{\mathsf{T}}\mathbf{j}_j$. Given integer N_g, we can find Gauss quadrature abscissas u_g and weights a_g such that m_{ij} can be approximated as follows: $m_{ij} \approx \sum_{g=1}^{N_g} a_g \mu(u_g) f_{ij}(u_g)$.

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 (19) 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]

VI. Physics-Based Shape Design

The physics-based shape design approach allows modeling requirements to be expressed and 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. Our physics-based shape design approach [35], [28], [29] which utilizes energies, forces, and constraints has proven to be simpler and more intuitive than conventional geometric design methods (e.g., the manipulation and adjustment of control points and weights). This approach is especially attractive for triangular NURBS, because of the complexity and irregularity of their control point and knot vectors.

A. Applied Forces

In the D-NURBS design scenario, 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. [34].

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

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

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. More advanced force tools are readily implemented to intuitively manipulate intrinsic geometric quantities such as normals and curvatures anywhere over D-NURBS objects.

[6] By also setting the damping density function $\gamma(u,v)$ to zero, designers reduce a dynamic model to a conventional nonlinear shape optimizer.

B. 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 [20]. 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 [20] combines the Lagrange multipliers with the simpler penalty method [26]. The Baumgarte stabilization method [1] solves constrained equations of motion through linear feedback control [18], [35]. 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 $\mathbf{p}$ [35].

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.

VII. Modeling Applications

This section describes our D-NURBS modeling environment and presents several applications.

A. Interactive Modeling Environment

We have developed a prototype modeling environment based on the curve, 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 one (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 [12]. 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.

Using our system, designers 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. At present, our software assumes uniform mass, damping, and elasticity densities over the parametric domain. This is straightforwardly generalizable to accommodate the non-uniform density functions in our formulation, although our user interface would have to be extended to afford the user full control in specifying these functions. In general, the qualitative effects of these parameter adjustments on D-NURBS can be intuitively comprehended through the principles of physical dynamics. For example, increasing the magnitude of α has a tendency to reduce the curve arc-length or surface area of D-NURBS, while increasing β tends to make objects more rigid. The damping density γ affects how fast an object converges to its equilibrium state. However, the quantitative effects of how the parameter setting influences the optimal shape of D-NURBS in geometric design remains an open problem. It should be explored in future research.

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. In the following sections we demonstrate applications of D-NURBS to interactive sculpting, solid rounding, and scattered data fitting. Table I specifies the physical parameters used in the subsequent experiments.

B. Optimal Fitting

D-NURBS are applicable to the optimal fitting of regular or scattered data [30]. The most general and often most useful case occurs with scattered data, when there are fewer or more 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 [32]. The surfaces are optimal in the sense that they provide the smoothest curve or surface (as measured by the deformation energy) that interpolates or approximates the data [4], [22].

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 VI-B. 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.

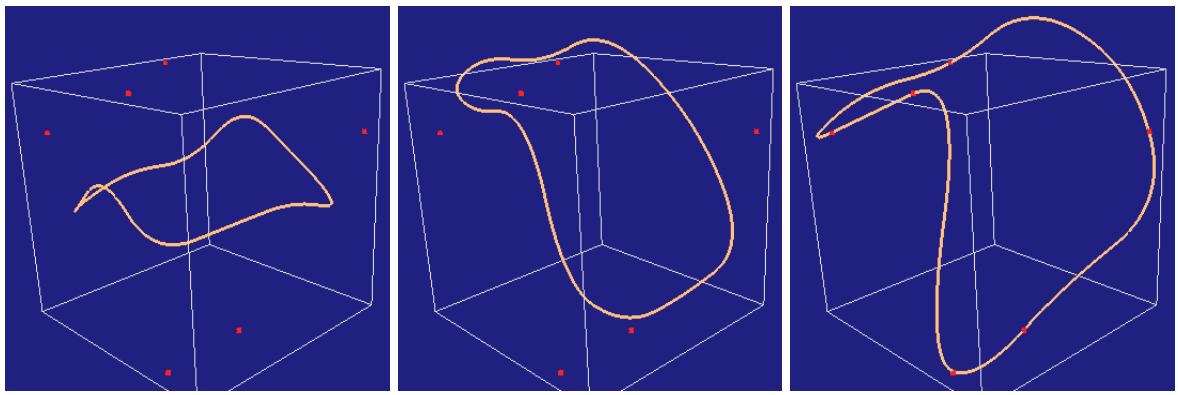
We first present an experiment of D-NURBS curve fitting coupled to data points through spring forces. Fig. 6(a) shows 6 sampled data points and an initial closed cubic D-NURBS curve with 11 control points. Note that the spring forces initiated by the data points are always applied to the nearest points on the curve. The spring attachments to the curve may vary during the fitting process, they are not explicitly illustrated in Fig. 6. One intermediate view of the fitting process and the final fitted curve are displayed in Fig. 6(b) and Fig. 6(c), respectively.

Next, we present three examples of surface fitting using tensor-product D-NURBS [35]. Fig. 7(a) shows 19 data points sampled from a hemisphere and their interpolation with a quadratic D-NURBS surface with 49 control points. Fig. 7(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. 7(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.

C. Rounding

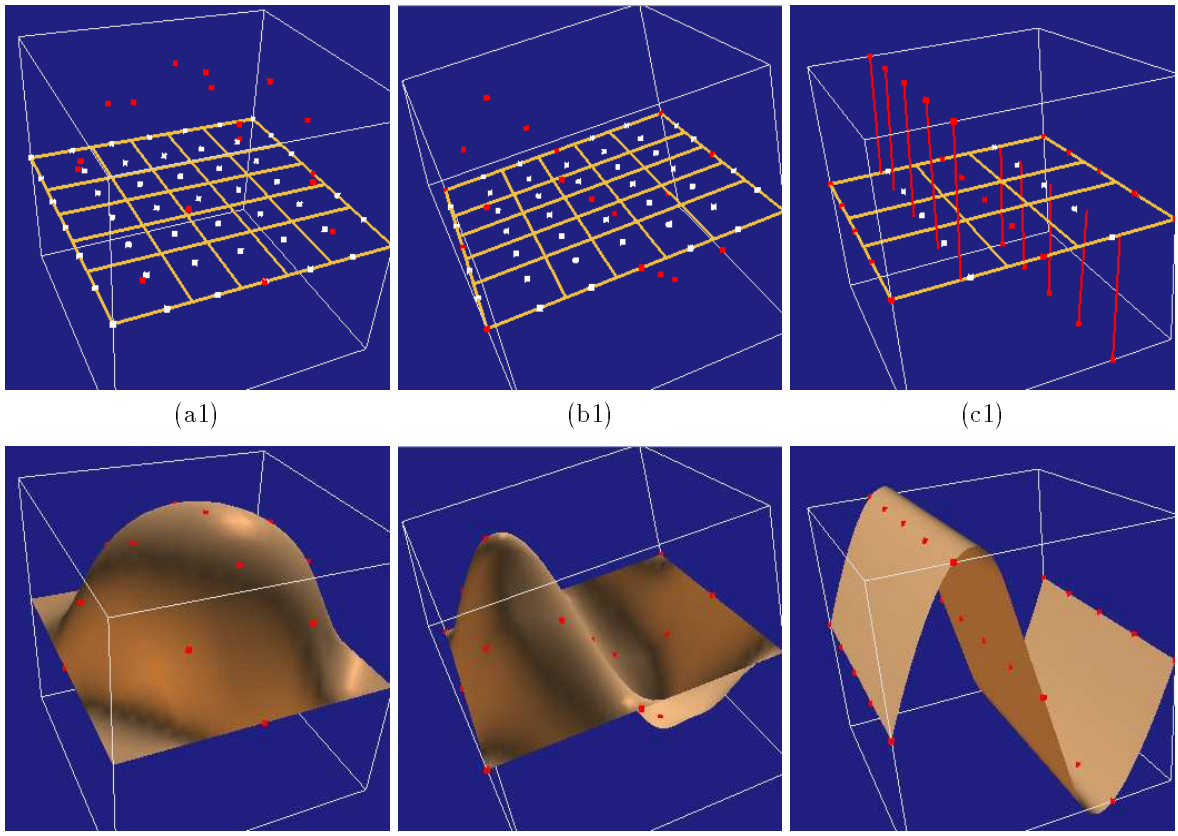
The rounding operation is usually attempted geometrically by enforcing continuity requirements on the fillet that interpolates between two or more surfaces. By contrast, the D-NURBS can produce a smooth fillet by minimizing its internal deformation energy subject to position and normal constraints. The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

Fig. 8(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 or with multiple knots. By restricting the control polygon to be a contin-



(a) (b) (c)

Fig. 6. D-NURBS curve fitting: (a) data points and the initial cubic curve, (b) intermediate scene, (c) the final fitted curve.



(a1) (b1) (c1)

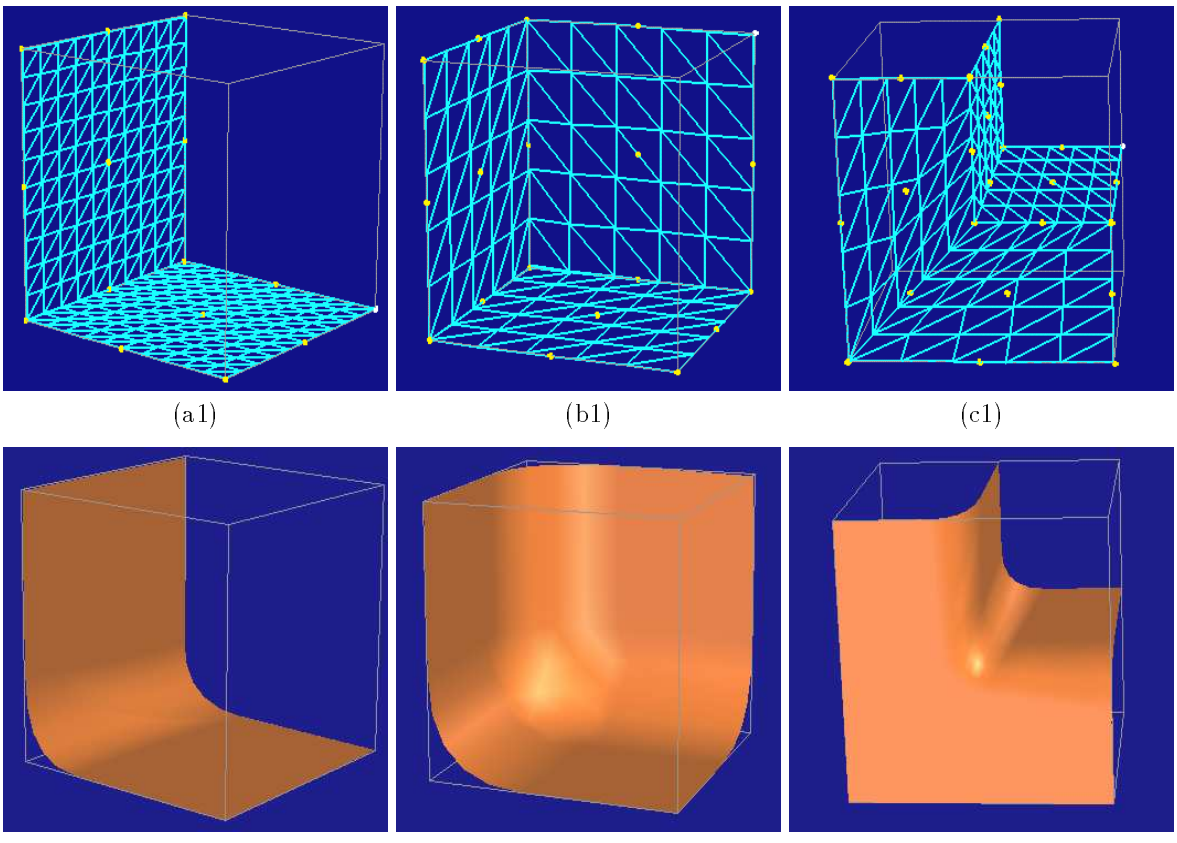
(a2) (b2) (c2)

Fig. 7. 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.

Applications	*Physical Parameters*								
	μ	γ	$\alpha_{1,1}$	$\alpha_{2,2}$	$\beta_{1,1}$	$\beta_{1,2}$	$\beta_{2,2}$	Δt	k
Curve fitting	0.0	50.0	$\alpha = 20.0$		$\beta = 60.0$			0.001	1000.0
Hemisphere fitting	0.0	50.0	10.0	10.0	5.0	5.0	5.0	0.02	5000.0
Convex/Concave fitting	0.0	50.0	10.0	10.0	5.0	5.0	5.0	0.03	2000.0
Sinusoidal fitting	0.0	100.0	2.0	2.0	10.0	10.0	10.0	0.02	3000.0
Edge rounding	0.0	500.0	1000.0	0.0	1.0	0.0	0.0	0.04	0.0
Corner rounding	0.0	50.0	1000.0	1000.0	10.0	10.0	10.0	0.04	0.0
Bevel rounding	0.0	25.0	100.0	100.0	0.0	0.0	0.0	0.04	0.0
Egg sculpting	1.0	25.0	100.0	100.0	20.0	20.0	20.0	0.002	3000.0
Toroid sculpting	1.0	100.0	25.0	25.0	10.0	10.0	10.0	0.003	4000.0
Hat sculpting	2.0	100.0	5.0	5.0	1.0	1.0	1.0	0.004	5000.0
Glass sculpting	2.0	50.0	15.0	15.0	4.0	4.0	4.0	0.001	2000.0

TABLE I

PHYSICAL PARAMETERS USED IN THE EXAMPLES. PARAMETER k DENOTES THE STIFFNESS OF THE SPRING FORCE.



(a1) (b1) (c1)

(a2) (b2) (c2)

Fig. 8. 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.

uous net, we reduced the number of control points to 21. The initial wireframe surface is shown in Fig. 8(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. 8(a2).

Fig. 8(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. 8(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. 8(b2).

Fig. 8(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. 8(c1–2). Note that, unlike Fig. 7(a1-c1), the wireframes appeared in Fig. 8(a1-c1) do not illustrate the boundaries of triangular patches.

D. 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. 9(a) illustrates the results of four interactive sculpting sessions using "texture-mapped" 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. 9(a). A torus whose two profile curves are quadratic with 7 and 7 control points, respectively, has been deformed into the shape in Fig. 9(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 9(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 9(c).

VIII. Conclusion

We have presented 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. 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 Centre of Ontario.

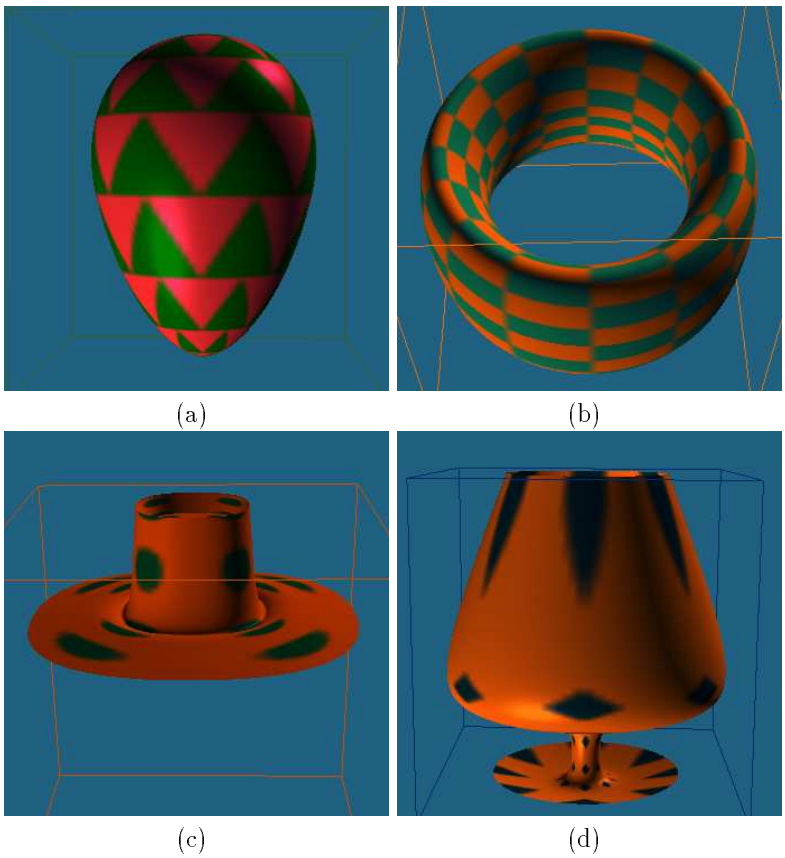


Fig. 9. Interactive Sculpting of Swung D-NURBS Surfaces. Open and closed surfaces shown were sculpted interactively from prototype shapes noted in parentheses (a) Egg shape (sphere). (b) Deformed toroid (torus). (c) Hat (open surface). (d) Wine glass (cylinder).

References

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

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

[3] M.I.G. Bloor and M.J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer-Aided Design*, 22(4):202–212, 1990.

[4] B. Brunnett, H. Hagen, and P. Santarelli. Variational design of curves and surfaces. *Surveys on Mathematics for Industry*, 3:1–27, 1993.

[5] 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).

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

[7] 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.

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

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

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

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

[12] 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.

[13] D.R. Forsey and R.H. Bartels. Hierarchical B-spline refinement. *Computer Graphics*, 22(4):205–212, 1988.

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

[15] G. Greiner. Variational design and fairing of spline surfaces. In *Proceedings of EUROGRAPHICS'94*, pages 143–154, Blackwell, 1994. Eurographics Association.

[16] 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.

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

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

[19] 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.

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

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

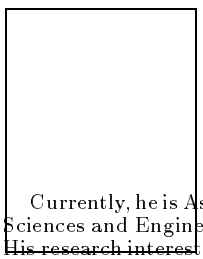
[22] R. Pfeifle and H.-P. Seidel. Fitting triangular B-Splines to functional scattered data. In *Proceedings of Graphics Interface'95*, pages 26–33, Morgan Kaufmann, 1995. Canadian Human-Computer Communications Society.

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

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

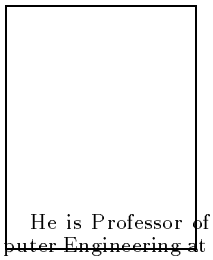
[25] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, Jan. 1991.

[26] J. Platt. A generalization of dynamic constraints. *CVGIP: Graphical Models and Image Processing*, 54(6):516–525, 1992.
[27] W. Press, B. Flanney, S. Teukolsky, and W. Verttering. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1986.
[28] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based shape design. *Computer Aided Design*, 27(2):111–127, 1995.
[29] H. Qin and D. Terzopoulos. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design*, 1996. in press.
[30] 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.
[31] J. Snyder and J. Kajiya. Generative modeling: A symbolic system for geometric modeling. *Computer Graphics*, 26(2):369–378, 1992.
[32] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.
[33] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.
[34] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.
[35] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.
[36] W. Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, Sept. 1983.
[37] K.J. Versprille. *Computer-Aided Design Applications of the Rational B-Spline Approximation form*. PhD thesis, Syracuse University, 1975.
[38] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992. (Proc. ACM Siggraph'92).
[39] C. Woodward. Cross-sectional design of B-spline surfaces. *Computers and Graphics*, 11(2):193–201, 1987.



Hong Qin (S'92) was born in Beijing, China. He received the B.S. degree and the M.S. degree in Computer Science from Peking University, Beijing, China, in 1986 and 1989, respectively. He obtained the Ph.D. degree in Computer Science from the University of Toronto, Toronto, Canada, in 1995.

Currently, he is Assistant Professor of Computer and Information Sciences and Engineering at the University of Florida, Gainesville. His research interests include Computer Graphics, Geometric Modeling, Computer Aided Design, and Scientific Visualization. He is a member of ACM, IEEE, and SIAM.



Demetri Terzopoulos (S'78, M'85) was born in Greece. He received the B.Eng. degree with distinction in Honours Electrical Engineering and the M.Eng. degree in Electrical Engineering from McGill University, Montreal, Canada, in 1978, and 1980, respectively, and the Ph.D. degree in Artificial Intelligence from the Massachusetts Institute of Technology, Cambridge, MA, in 1984.

He is Professor of Computer Science and Electrical and Computer Engineering at the University of Toronto, where he leads the Visual Modeling Group, and is a Fellow of the Canadian Institute for Advanced Research. From 1985-92 he was affiliated with Schlumberger, Inc., serving as Program Leader at research labs in Palo Alto, CA, and Austin, TX. During 1984-85 he was a research scientist at the MIT Artificial Intelligence Lab, Cambridge, MA. He has been a consultant to Digital, Hughes, NEC, Ontario Hydro, and Schlumberger.

His published works include more than 150 scientific articles, primarily in computer vision and graphics, and also in computer-aided design, medical imaging, artificial intelligence, and artificial life, including the recent edited volumes "Real-Time Computer Vision" (Cambridge Univ. Press '94) and "Animation and Simulation" (Springer-Verlag '95). His contributions have been recognized with several awards. In 1996 he was awarded the prestigious Natural Sciences and Engineering Research Council of Canada E.W.R. Steacie Memorial Fellowship. His other awards include three university Excellence Awards, an award from the American Association for Artificial Intelligence in 1987 for his work on deformable models in vision, an award from the IEEE in 1987 for his work on active contours ("snakes"), and awards from the Canadian Academy of Multimedia Arts and Sciences in 1994 and from Ars Electronica in 1995 for his work on artificial animals. He serves on the editorial boards of the journals *Medical Image Analysis*, *Graphical Models and Image Processing*, and the *Journal of Visualization and Computer Animation*. He has served on ARPA and NIH advisory committees and is a member of the New York Academy of Sciences and Sigma Xi.