Dynamic NURBS with Geometric Constraints for Interactive Sculpting

Demetri Terzopoulos and **Hong Qin**
Department of Computer Science, University of Toronto

Published in *ACM Transactions on Graphics*, **13**(2), April, 1994, 103-136.

Abstract

This article develops a dynamic generalization of the nonuniform rational B-spline (NURBS) model. NURBS have become a *de facto* standard in commercial modeling systems because of their power to represent free-form shapes as well as common analytic shapes. To date, however, they have been viewed as purely geometric primitives that require the user to manually adjust multiple control points and associated weights in order to design shapes. Dynamic NURBS, or D-NURBS, are physics-based models that incorporate mass distributions, internal deformation energies, and other physical quantities into the popular NURBS geometric substrate. Using D-NURBS, a modeler can interactively sculpt curves and surfaces and design complex shapes to required specifications not only in the traditional indirect fashion, by adjusting control points and weights, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. D-NURBS move and deform in a physically intuitive manner in response to the user's direct manipulations. Their dynamic behavior results from the numerical integration of a set of nonlinear differential equations that automatically evolve the control points and weights in response to the applied forces and constraints. To derive these equations, we employ Lagrangian mechanics and a finite-element-like discretization. Our approach supports the trimming of D-NURBS surfaces using D-NURBS curves. We demonstrate D-NURBS models and constraints in applications including the rounding of solids, optimal surface fitting to unstructured data, surface design from cross sections, and free-form deformation. We also introduce a new technique for 2D shape metamorphosis using constrained D-NURBS surfaces.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling — Curve, Surface, solid, and object representations; Physically based modeling; splines; I.3.6 [Computer Graphics]: Methodology and Techniques — Interaction techniques.

General Terms: Algorithms, Design, Theory

Additional Keywords and Phrases: CAGD, NURBS, Deformable Models, Dynamics, Constraints, Finite Elements, Trimming, Solid Rounding, Optimal Curve and Surface Fitting, Cross-Sectional Shape Design, Free-Form Deformation, Shape Metamorphosis.

1 Introduction

In 1975 Versprille [40] 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 [23]. 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. By adjusting the positions of control points and manipulating associated weights, one can design a large variety of shapes using NURBS [10, 12, 24, 21, 22, 23, 39].

Because NURBS are a purely geometric representation, however, their 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, etc.?

- 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 [37] and further developed in [36, 26, 33, 20, 34, 17] are particularly relevant in the context of modeling with NURBS. Important advantages accrue from the deformable model approach [36]:

- 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 [35]. 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.

In this article, 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 curves and surfaces, 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 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.

Section 2 discusses the similarities and distinctive features of D-NURBS relative to prior models. Section 3 briefly reviews NURBS geometry and its properties. In Section 4, we formulate D-NURBS and derive their equations of motion. Section 5 discusses the application of forces and constraints for physics-based design. We discuss the numerical simulation of D-NURBS in Section 6. Section 7 describes our prototype D-NURBS modeling system and presents applications and results. Section 8 concludes the article.

2 Background

Dynamic NURBS are motivated by prior research aimed at applying the deformable modeling approach to shape design. Terzopoulos and Fleischer [36] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [5] developed an interesting prototype system for interactive free-form design based on the finite-element optimization of energy functionals proposed in [36]. Bloor and Wilson [3] developed related models using similar energies and numerical optimization, and in [2] they proposed the use of B-splines for this purpose. Subsequently, Celniker and Welch [6] investigated deformable B-splines with linear constraints. Welch and Witkin [42] extended the approach to trimmed hierarchical B-splines (see also [13]). Thingvold and Cohen [38] proposed a deformable B-spline whose control points are mass points connected by elastic springs and hinges.

In [2, 6, 42] 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 in the case of [6] involve simple linear dynamics with diagonal (arbitrarily lumped) mass and damping matrices (apparently for efficiency).

D-NURBS are a more sophisticated dynamic model. We adopt the approach proposed in [17] for converting arbitrary geometric models into dynamic models using Lagrangian mechanics and finite element analysis. Our approach is systematic. We formulate deformable curves and surfaces and reduce them to algorithms in a principled way, without resorting to any of the *ad hoc* assumptions of prior schemes (c.f. [38]). Because our dynamic models allow fully continuous mass and damping distributions, we obtain banded mass and damping matrices. These are known as *consistent* matrices in the finite element literature [43].

The D-NURBS control points and associated weights become 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,

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

this does not preclude interactive performance on current graphics workstations, at least for the size of surface models that appear in our demonstrations. We prove several mathematical results that enable us to simplify the motion equations and apply numerical quadrature to the underlying NURBS basis functions to compute efficiently the integral expressions for the matrix entries.

3 NURBS Geometry

In this section, we review the formulation of NURBS curves and surfaces. We then describe their analytic and geometric properties. More detailed material can be found in [4, 24, 21, 22, 23, 39].

3.1 Curves

A NURBS curve generalizes the B-spline. It is the combination of a set of piecewise rational functions with $n+1$ control points $\mathbf{p}_i$ and associated weights w_i:

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

where u is the parametric variable and $B_{i,k}(u)$ are B-spline basis functions. Assuming basis functions of degree $k-1$, a NURBS curve has $n+k+1$ knots t_i in nondecreasing sequence: $t_0 \leq t_1 \leq \ldots \leq t_{n+k}$. The basis functions are defined recursively as

$$B_{i,1}(u) = \begin{cases} 1 & \text{for } t_i \leq u < t_{i+1} \\ 0 & \text{otherwise} \end{cases},$$

with

$$B_{i,k}(u) = \frac{u - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(u).$$

The parametric domain is $t_{k-1} \leq u \leq t_{n+1}$. 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 $\mathbf{p}_n$.

3.2 Surfaces

A NURBS surface is the generalization of the tensor-product B-spline surface. It is defined over the parametric variables u and v as

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

A NURBS surface has $(m+1)(n+1)$ control points $\mathbf{p}_{i,j}$ and weights $w_{i,j}$. 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 nondecreasing knot sequence is $t_0 \leq t_1 \leq \ldots \leq t_{m+k}$ along the u-axis and $s_0 \leq s_1 \leq \ldots \leq s_{n+l}$ along the v-axis. The parametric domain is $t_{k-1} \leq u \leq t_{m+1}$ and $s_{l-1} \leq v \leq s_{n+1}$. 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.

3.3 Properties

NURBS generalize the nonrational parametric form. Like nonrational B-splines, the rational basis functions of NURBS sum to unity, they are infinitely smooth in the interior of a knot span provided the denominator

significantly.

is not zero, and at a knot they are at least C^{k-1-r} continuous with knot multiplicity r, which enables them to satisfy different smoothness requirements. They inherit many of the properties of nonrational B-splines, such as the strong convex hull property, variation diminishing property, local support, and invariance under standard geometric transformations (see [11] for more details). Moreover, they have some additional properties:

- 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.
- NURBS include weights as extra degrees of freedom which influence local shape. If a particular weight is zero, then the corresponding rational basis function is also zero and its control point does not effect the NURBS shape. The spline is attracted toward a control point more if the corresponding weight is increased and less if the weight is decreased.

For a more detailed discussion of NURBS properties, see [23, 24, 39, 12, 21, 22].

The most frequently used NURBS design techniques are the specification of a control polygon, or 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. The initial shape is then refined into the final desired shape through interactive adjustment of control points and weights and possibly the addition or deletion of knots. The refinement process is *ad hoc* and often tedious. To ameliorate it, we propose dynamic NURBS.

4 Formulation of D-NURBS

This section formulates our physics-based D-NURBS model. The shape parameters of geometric NURBS, which were described in Section 3, 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.

4.1 Curves

For simplicity, consider first a D-NURBS space curve. The D-NURBS curve is defined as in (1), but it is also a function of the spatial parameter u and time t:

$$\mathbf{c}(u,t) = \frac{\sum_{i=0}^{n} \mathbf{p}_i(t) w_i(t) B_{i,k}(u)}{\sum_{i=0}^{n} w_i(t) B_{i,k}(u)}. \tag{3}$$

The control points $\mathbf{p}_i(t)$ and weights $w_i(t)$, which are now functions of time, comprise the generalized coordinates of D-NURBS. To simplify notation, we concatenate the generalized coordinates into the following vectors:

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

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

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

where $^\top$ denotes transposition. Note that we can express the curve $\mathbf{c}(u,t)$ as $\mathbf{c}(u,\mathbf{p})$ in order to emphasize its dependence on the vector of generalized coordinates $\mathbf{p}$ whose components are functions of time. The velocity of the kinematic spline is

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

where an overstruck dot denotes a time derivative and $\mathbf{J}(u, \mathbf{p})$ is the Jacobian matrix. Because $\mathbf{c}$ is a 3-component vector-valued function and $\mathbf{p}$ is an $4(n+1)$ dimensional vector, $\mathbf{J}$ is a $3 \times 4(n+1)$ matrix which is the concatenation of the vectors $\partial\mathbf{c}/\partial\mathbf{p}_i$ and $\partial\mathbf{c}/\partial w_i$, $i = 0, \ldots, n$. Let us investigate the contents of $\mathbf{J}$. For $i = 0, \ldots, n$, let $\mathbf{B}_i(u, \mathbf{p})$ be a 3×3 diagonal matrix whose diagonal entries are the rational basis functions

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

and let the 3-vector

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

We collect the $\mathbf{B}_i$ into $\mathbf{B}$ and the $\mathbf{w}_i$ into $\mathbf{W}$ as follows

$$\mathbf{B}(u, \mathbf{p}) = \left[\begin{array}{ccc} \mathbf{B}_0 & \cdots & \mathbf{B}_n \end{array} \right],$$

$$\mathbf{W}(u, \mathbf{p}) = \left[\begin{array}{ccc} \mathbf{w}_0 & \cdots & \mathbf{w}_n \end{array} \right].$$

The Jacobian matrix may then be written as

$$\mathbf{J}(u, \mathbf{p}) = \left[\begin{array}{ccccc} \mathbf{B}_0 & \mathbf{w}_0 & \cdots & \mathbf{B}_n & \mathbf{w}_n \end{array} \right].$$

Using the foregoing notation, we can express

$$\mathbf{c} = \mathbf{B}\mathbf{p}_b,$$

Appendix A shows that

$$\mathbf{W}\mathbf{p}_w = \mathbf{0}, \tag{5}$$

so that we can express the D-NURBS as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{c}(u, \mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{6}$$

Another interesting relationship is $\dot{\mathbf{J}}\mathbf{p} = \mathbf{0}$, and it will enable us to simplify the discretized version of the D-NURBS differential equations and arrive at an efficient numerical implementation.

4.2 Surfaces

A D-NURBS surface has a similar structure to the curve. Proceeding analogously from (2), we define

$$\mathbf{s}(u, v, t) = \frac{\sum_{i=0}^m \sum_{j=0}^n \mathbf{p}_{i,j}(t) w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^m \sum_{j=0}^n w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}. \tag{7}$$

Again, the control points and weights comprise the generalized coordinates and are assembled into vectors $\mathbf{p}_b$, $\mathbf{p}_w$, and $\mathbf{p}$. 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.

As before, we can write $\mathbf{s}(u, v, \mathbf{p})$ instead of $\mathbf{s}(u, v, t)$. By analogy to $\mathbf{c}$ in (4) and (6), we obtain for the D-NURBS surface

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

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}. \tag{9}$$

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

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

As before, the $\mathbf{B}_{i,j}$ and $\mathbf{w}_{i,j}$ are assembled into $\mathbf{B}$ and $\mathbf{W}$, respectively. Hence,

$$\mathbf{J}(u,v,\mathbf{p}) = \left[\begin{array}{cccccc} \mathbf{B}_{0,0} & \mathbf{w}_{0,0} & \cdots & \mathbf{B}_{m,n} & \mathbf{w}_{m,n} \end{array}\right].$$

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

4.3 D-NURBS Equations of Motion

The previous two sections presented D-NURBS curve and surface geometry in a unified way. D-NURBS physics are based on the work-energy version of Lagrangian dynamics [14]. In an abstract physical system, let $p_i(t)$ be a set of generalized coordinates. These N functions of time are assembled into the vector $\mathbf{p}$. Let $f_i(t)$ be the generalized applied force that acts on p_i. We assemble the f_i into the vector $\mathbf{f}_p$. We also assume that $\mathbf{J}$ is the concatenation of N vectors $\mathbf{j}_i$.

To proceed with the Lagrangian formulation, we will define kinetic energy T, potential energy U, and Raleigh dissipation energy F which are functions of the generalized coordinates and their derivatives. The Lagrangian equations of motion are then expressed as

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{p}_i} - \frac{\partial T}{\partial p_i} + \frac{\partial F}{\partial \dot{p}_i} + \frac{\partial U}{\partial p_i} = f_i. \tag{10}$$

Variants of this equation have served as the basis for deformable model formulations [36]. Using (10), we can take an arbitrary geometric model, such as a NURBS, introduce appropriate kinetic, potential, and dissipation energies, and systematically formulate a physics-based, dynamic generalization of the model [17].

In the sequel, we will discuss only D-NURBS surfaces (we can consider D-NURBS curves as a special case with a simpler expression in fewer variables). To define energies and derive the D-NURBS equations of motion, let $\mu(u,v)$ be the mass density function defined over the parametric domain of the surface. The kinetic energy of the surface is

$$T = \frac{1}{2}\int\!\!\int \mu \dot{\mathbf{s}}^\top \dot{\mathbf{s}}\, du\, dv = \frac{1}{2}\dot{\mathbf{p}}^\top \mathbf{M}\dot{\mathbf{p}}, \tag{11}$$

where (using (9))

$$\mathbf{M}(\mathbf{p}) = \int\!\!\int \mu \mathbf{J}^\top \mathbf{J}\, du\, dv \tag{12}$$

is an $N \times N$ mass matrix. Similarly, let $\gamma(u,v)$ be the damping density function. The dissipation energy is

$$F = \frac{1}{2}\int\!\!\int \gamma \dot{\mathbf{s}}^\top \dot{\mathbf{s}}\, du\, dv = \frac{1}{2}\dot{\mathbf{p}}^\top \mathbf{D}\dot{\mathbf{p}}, \tag{13}$$

where

$$\mathbf{D}(\mathbf{p}) = \int\!\!\int \gamma \mathbf{J}^\top \mathbf{J}\, du\, dv \tag{14}$$

is the damping matrix.

For the elastic potential energy of D-NURBS, we can adopt the *thin-plate under tension* energy model [35], which was also used in [5, 42] (other energies are possible, including the nonquadratic, curvature-based energies in [36, 19]):

$$U = \frac{1}{2}\iint\left(\alpha_{1,1}\frac{\partial \mathbf{s}^\top}{\partial u}\frac{\partial \mathbf{s}}{\partial u} + \alpha_{2,2}\frac{\partial \mathbf{s}^\top}{\partial v}\frac{\partial \mathbf{s}}{\partial v} + \beta_{1,1}\frac{\partial^2 \mathbf{s}^\top}{\partial u^2}\frac{\partial^2 \mathbf{s}}{\partial u^2} + \beta_{1,2}\frac{\partial^2 \mathbf{s}^\top}{\partial u \partial v}\frac{\partial^2 \mathbf{s}}{\partial u \partial v} + \beta_{2,2}\frac{\partial^2 \mathbf{s}^\top}{\partial v^2}\frac{\partial^2 \mathbf{s}}{\partial v^2}\right) du\, dv = \frac{1}{2}\mathbf{p}^\top \mathbf{K}\mathbf{p}. \tag{15}$$

The $\alpha_{i,j}(u,v)$ and $\beta_{i,j}(u,v)$ are elasticity functions which control local tension and rigidity, respectively, in the two parametric coordinate directions.[2] In view of (8), the $N \times N$ stiffness matrix is

$$\mathbf{K}(\mathbf{p}) = \iint \left(\alpha_{1,1}\mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2}\mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1}\mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2}\mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2}\mathbf{J}_{vv}^\top \mathbf{J}_{vv}\right) du\, dv, \tag{16}$$

where the subscripts on $\mathbf{J}$ denote parametric partial derivatives.

In Appendix B, we show that by applying (10), the D-NURBS equations of motion are given by

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \mathbf{I}\dot{\mathbf{p}}, \tag{17}$$

where the generalized force vector, obtained through the principle of virtual work [14] done by the applied force distribution $\mathbf{f}(u,v,t)$, is

$$\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{J}^\top \mathbf{f}(u,v,t)\, du\, dv, \tag{18}$$

and where

$$\mathbf{I}(\mathbf{p}) = \iint \mu \mathbf{J}^\top \dot{\mathbf{J}}\, du\, dv.$$

5 Forces and Constraints

We have derived the Lagrangian equations of motion for D-NURBS. When working with D-NURBS, a modeler may impose design requirements in terms of energies, forces, and constraints. For instance, the modeler 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 $\mathbf{K}$. By building the physics-based D-NURBS generalization 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. For example, if the shapes of certain cross-sectional curves in a NURBS surface must be circular arcs, the control points associated with these curves must be constrained geometrically to admit only circular arcs. Other constraints include the specification of positions of surface points, the specification of surface normals at surface points, and continuity requirements between adjacent surface patches or curve arcs.

5.1 Applied Forces

In the D-NURBS design scenario, sculpting tools may be implemented as applied forces. The force $\mathbf{f}(u,v,t)$ in the D-NURBS equation of motion represents the net effect of all applied forces. Typical force functions are spring forces, repulsion forces, gravitational forces, inflation forces, etc. [36, 6, 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

[2] 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: $U = \frac{1}{2}\int \alpha(u)\frac{\partial \mathbf{c}^\top}{\partial u}\frac{\partial \mathbf{c}}{\partial u} + \beta(u)\frac{\partial^2 \mathbf{c}^\top}{\partial u^2}\frac{\partial^2 \mathbf{c}}{\partial u^2}\, du$.

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 is 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.

5.2 Linear Constraints

Linear geometric constraints such as point, curve, and surface normal constraints are often useful [6]. To incorporate linear geometric constraints into D-NURBS, we reduce the matrices and vectors in (17) to a minimal unconstrained set of generalized coordinates. Linear constraints are generally expressible as follows:

$$\mathbf{C}(\mathbf{p}) = \mathbf{A}\mathbf{p} + \mathbf{b} = \mathbf{0}, \tag{20}$$

where $\mathbf{A}$ is a matrix of coefficients. If (20) is an underdetermined linear system, we can eliminate variables to express the generalized coordinate vector $\mathbf{p}$ as

$$\mathbf{p} = \mathbf{G}\mathbf{q} + \mathbf{q}_0, \tag{21}$$

where $\mathbf{q}$ is a new generalized coordinate vector with $M < N$ components q_j. Here, $\mathbf{G}$ is an $N \times M$ matrix, which may be computed through Gaussian elimination or other means, and $\mathbf{q}_0$ is a constant vector.

The lower-dimensional generalized coordinate vector $\mathbf{q}$ replaces $\mathbf{p}$ in the linearly constrained D-NURBS model. To derive the equations of motion with constraints, we combine (8) and (9) with (21) as follows:

$$\mathbf{s}(u,v,\mathbf{q}) = \mathbf{J}(\mathbf{G}\mathbf{q} + \mathbf{q}_0) = \mathbf{L}\mathbf{q} + \mathbf{J}\mathbf{q}_0$$

$$\dot{\mathbf{s}}(u,v,\mathbf{q}) = \mathbf{J}(\mathbf{G}\dot{\mathbf{q}} + \dot{\mathbf{p}}_0) = \mathbf{J}\mathbf{G}\dot{\mathbf{q}} = \mathbf{L}\dot{\mathbf{q}},$$

where

$$\mathbf{L} = \mathbf{J}\mathbf{G}$$

is the new Jacobian matrix of $\mathbf{s}$ with respect to $\mathbf{q}$. Note that $\mathbf{L}$ consists of M vectors $\mathbf{l}_j = \partial\mathbf{s}/\partial q_j$, for $j = 1, \ldots, M$. Hence, the energy expressions become

$$T = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}^\top \mathbf{M}\mathbf{G}\dot{\mathbf{q}}$$

$$F = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}^\top \mathbf{D}\mathbf{G}\dot{\mathbf{q}}$$

$$U = \frac{1}{2}(\mathbf{q}^\top \mathbf{G}^\top + \mathbf{q}_0)\mathbf{K}(\mathbf{G}\mathbf{q} + \mathbf{q}_0).$$

We also define the $M \times M$ mass, damping, and stiffness matrices of the constrained D-NURBS:

$$\mathbf{M}_q = \mathbf{G}^\top \mathbf{M}\mathbf{G}$$

$$\mathbf{D}_q = \mathbf{G}^\top \mathbf{D}\mathbf{G}$$

$$\mathbf{K}_q = \mathbf{G}^\top \mathbf{K}\mathbf{G}.$$

In Appendix C we prove several identities that yield the following equations of motion for D-NURBS with linear constraints:

$$\mathbf{M}_q\ddot{\mathbf{q}} + \mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{g}_q - \mathbf{I}_q\dot{\mathbf{q}}, \tag{22}$$

where the generalized forces are

$$\mathbf{f}_q = \iint \mathbf{L}^\top \mathbf{f}(u, v, t)\, du\, dv, \tag{23}$$

and where

$$\mathbf{g}_q = -\mathbf{G}^\top \mathbf{K}\mathbf{q}_0,$$
$$\mathbf{I}_q = \mathbf{G}^\top \mathbf{I}\mathbf{G}.$$

Although (22) looks more complicated than (17), its implementation is surprisingly straightforward in view of the sparseness of $\mathbf{G}$ and the reduced size of $\mathbf{q}$.

5.3 Nonlinear Constraints

It is possible to impose nonlinear geometric (equality) constraints

$$\mathbf{C}(\mathbf{p}) = \mathbf{0}, \tag{24}$$

on D-NURBS through Lagrange multiplier techniques [32]. This approach increases the number of degrees of freedom, hence the computational cost, by adding unknowns λ_i—also known as Lagrange multipliers—which determine the magnitudes of the constraint forces. The method is applied to the B-spline model in [6, 42]. The augmented Lagrangian method [18] combines the Lagrange multipliers with the simpler penalty method [26].

One of the best known techniques for applying constraints to dynamic models is the Baumgarte stabilization method [1] which solves constrained equations of motion through linear feedback control (see also [17, 25]). We augment (17) as follows:

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \mathbf{I}\dot{\mathbf{p}} - \mathbf{C}_\mathbf{p}^\top \boldsymbol{\lambda}, \tag{25}$$

where $-\mathbf{C}_\mathbf{p}^\top \boldsymbol{\lambda}$ are generalized forces stemming from the holonomic constraint equations. The term $\mathbf{C}_\mathbf{p}^\top$ is the transpose of the constraint Jacobian matrix and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$ is a vector of Lagrange multipliers that must be determined. We can obtain the same number of equations as unknowns, by differentiating (24) twice with respect to time: $\ddot{\mathbf{C}}(\mathbf{p}) = \mathbf{0}$. Baumgart's method replaces these additional equations with equations that have similar solutions, but which are asymptotically stable; e.g., the damped second-order differential equations $\ddot{\mathbf{C}} + 2a\dot{\mathbf{C}} + b^2\mathbf{C} = \mathbf{0}$, where a and b are stabilization factors. For a given value of a, we can choose $b = a$ to obtain the critically damped solution $\mathbf{C}(\mathbf{p}, 0)e^{-at}$ which has the quickest asymptotic decay towards constraint satisfaction (24). Taking the second time derivative of (24) and rearranging terms yields

$$\mathbf{C}_\mathbf{p}\ddot{\mathbf{p}} = -\ddot{\mathbf{C}} - (\mathbf{C}_\mathbf{p}\dot{\mathbf{p}})_\mathbf{p}\dot{\mathbf{p}} - 2\dot{\mathbf{C}}_\mathbf{p}\dot{\mathbf{p}} = \boldsymbol{\sigma}. \tag{26}$$

We arrive at the following system of equations for the unknown constrained generalized accelerations and Lagrange multipliers:

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{p}^\top \\ \mathbf{C}_\mathbf{p} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{p}} - \mathbf{K}\mathbf{p} - \mathbf{I}\dot{\mathbf{p}} + \mathbf{f}_p \\ \boldsymbol{\sigma} - 2a\mathbf{C}_\mathbf{p}\dot{\mathbf{p}} - b^2\mathbf{C} \end{bmatrix}. \tag{27}$$

This system can be solved for $\ddot{\mathbf{p}}$ and $\boldsymbol{\lambda}$ using standard direct or iterative techniques (or in the least squares sense when it is overdetermined by conflicting constraints).

5.4 Constraining the Weights

The components of $\mathbf{p}_b$ may take arbitrary finite values in $\Re$, but this is not the case for the weights $\mathbf{p}_w$. Negative components of $\mathbf{p}_w$ may cause the denominator to vanish at some evaluation points, causing the matrices to diverge. Although not forbidden, negative weights are not useful. We include constraints in our D-NURBS which enforce positivity of weight values. Such a constraint is easily implemented by establishing a positive lower bound on the weight values and enforcing it in the numerical solution using a projection method.

Another potential difficulty is that smaller values of $\mathbf{p}_w$ tend to flatten the surface in the vicinity of the control points, which lowers the deformation energy. Consequently, the $\mathbf{p}_w$ will tend to move toward zero. To counteract this tendency, we can associate with the potential energy the penalty term

$$c(\mathbf{p}_w - \mathbf{p}_w^0)^\top (\mathbf{p}_w - \mathbf{p}_w^0)$$

in which $\mathbf{p}_w^0$ are desired weights and c is scaling factor.

We have implemented both techniques. Experiments indicate that the projection scheme works very well. Consequently, we do not make use of the penalty scheme in our current modeling system. It may be useful, however, if the modeler wants to constrain the weights to assume values near certain target values $\mathbf{p}_w^0$.

6 Numerical Implementation

The evolution of the D-NURBS generalized coordinates is determined by the second-order nonlinear differential equations (17) or (22), with time-varying mass, damping, and stiffness matrices. We cannot obtain an analytical solution in general. An efficient numerical implementation of D-NURBS is possible, however, through the use of techniques from finite-element analysis [15].

Standard finite element codes assemble individual element matrices into the global matrices that appear in the discrete equations of motion [43, 15]. Despite the fact that the global matrices are stored using efficient sparse matrix storage schemes (which maintain only the entries needed for matrix factorization), matrix assembly and matrix-vector multiplications quickly become too costly, particularly for D-NURBS surfaces with high dimensional $\mathbf{p}$.

In our implementation, we use an iterative matrix solver that enables us to avoid the costs of assembling the global $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ matrices associated with the whole D-NURBS curve or surface. Rather, we work with the individual D-NURBS element matrices. We construct finite element data structures that contain the information needed to compute all of the element matrices independently and in parallel.

6.1 Data Structures for D-NURBS Finite Elements

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. We define an element data structure which contains the geometric specification of the D-NURBS element along with its physical properties. A complete D-NURBS curve or surface is then implemented as a data structure which consists of an ordered array of D-NURBS curve or surface elements with additional information.

The element structure includes pointers to the associated generalized coordinates (control points and weights). For instance, 9 control points and associated weights are needed to describe a patch of a quadratic D-NURBS surface (the total number of degrees of freedom is 36). The generalized coordinates associated with the entire D-NURBS curve or surface are stored in the global vector $\mathbf{p}$. Note that neighboring elements will share some generalized coordinates. The shared variables will have multiple pointers impinging on them.

We also allocate in each D-NURBS 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.

6.2 Calculation of Element Matrices

We evaluate the integral expressions for the matrices (12), (14), and (16) numerically using Gaussian quadrature [27]. We shall explain the computation of the element stiffness matrix; the computation of the mass and damping matrices follow suit. Assuming the element's parametric domain is $[u_0, u_1] \times [v_0, v_1]$, the expression for entry k_{ij} of the stiffness matrix of a D-NURBS surface element takes the integral form

$$k_{ij} = \int_{u_0}^{u_1} \int_{v_0}^{v_1} f_{ij}(u, v)\, du\, dv, \tag{28}$$

where, according to (16),

$$\begin{aligned} f_{ij}(u,v) = {} & \alpha_{1,1}(u,v)\frac{\partial \mathbf{j}_i^\top}{\partial u}\frac{\partial \mathbf{j}_j}{\partial u} + \alpha_{2,2}(u,v)\frac{\partial \mathbf{j}_i^\top}{\partial v}\frac{\partial \mathbf{j}_j}{\partial v} \\ & + \beta_{1,1}(u,v)\frac{\partial^2 \mathbf{j}_i^\top}{\partial u^2}\frac{\partial^2 \mathbf{j}_j}{\partial u^2} + \beta_{1,2}(u,v)\frac{\partial^2 \mathbf{j}_i^\top}{\partial u \partial v}\frac{\partial^2 \mathbf{j}_j}{\partial u \partial v} + \beta_{2,2}(u,v)\frac{\partial^2 \mathbf{j}_i^\top}{\partial v^2}\frac{\partial^2 \mathbf{j}_j}{\partial v^2}. \end{aligned}$$

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

We apply Gaussian quadrature to compute the above integral approximately. The integral is obtained by applying Gaussian quadrature on the 1-D interval twice. Given integer N_g and N_h, we can find Gauss weights a_g, b_h and abscissas u_g, v_h in two directions of the parametric domain such that k_{ij} can be approximated by ([27])

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

We apply the de Boor algorithm [9] to evaluate $f_{ij}(u_g, v_h)$.[3]

Generally speaking, for integrands that are polynomial of degree $2N-1$ or less, Gaussian quadrature evaluates the integral exactly with N weights and abscissas. For D-NURBS, f_{ij} is not polynomial unless the model is reduced to a B-spline. In our system, we choose N_g and N_h to be integers between 4 and 7. Our experiments reveal that matrices computed in this way lead to stable, convergent solutions.

6.3 Discrete Dynamics Equations

In order to integrate the D-NURBS ordinary differential equations of motion (17) in an interactive modeling environment, it is important to provide the modeler or designer 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 matrices $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ (and $\mathbf{M}_q$, $\mathbf{D}_q$, and $\mathbf{K}_q$) are symmetric, sparse, and banded. Several algorithms are available for the numerical integration of the D-NURBS ordinary differential equations of motion. The suitability of implicit or explicit integration algorithms is dependent on the bandwidth of the matrices, as determined by the dimensionality of the parametric space and the order of the NURBS basis functions. The matrices for a D-NURBS curve have a single band which has a half-bandwidth of $4k$, where k is the order of the NURBS basis. For D-NURBS surfaces, the matrices become block banded, with each block containing n bands similar to those of dynamic curves, where n depends on the order of the NURBS basis in the opposite parametric direction.

[3] The entries of the D-NURBS curve element stiffness matrix are $k_{ij} = \int_{u_0}^{u_1} f_{ij}(u)\, du$, where $f_{ij}(u) = \alpha(u)\frac{\partial \mathbf{j}_i^\top}{\partial u}\frac{\partial \mathbf{j}_j}{\partial u} + \beta(u)\frac{\partial^2 \mathbf{j}_i^\top}{\partial u^2}\frac{\partial^2 \mathbf{j}_j}{\partial u^2}$ Given integer N_g, we can find Gauss quadrature abscissas u_g and weights a_g such that k_{ij} can be approximated as follows: $k_{ij} \approx \sum_{g=1}^{N_g} a_g f_{ij}(u_g)$.

We integrate the differential equations (17) through time by discretizing the derivative of $\mathbf{p}$ over time-steps Δt. The state of the D-NURBS at time $t + \Delta t$ is integrated using prior states at time t and $t - \Delta t$. Depending on the choice of physical parameters, (17) may be a stiff system. We use an implicit time integration method in order to maintain the stability of the integration scheme. The implicit method employs discrete derivatives of $\mathbf{p}$ using backward differences

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

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

Making use of the fact that $\dot{\mathbf{J}}\mathbf{p} = \mathbf{0}$, we obtain the time integration formula

$$\left(4\mathbf{M} + 2\Delta t\mathbf{D} + 4\Delta t^2\mathbf{K}\right)\mathbf{p}^{(t+\Delta t)} = 4\Delta t^2\mathbf{f}_p + 8\mathbf{M}\mathbf{p}^{(t)} - (3\mathbf{M} - 2\Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)} - \iint \mu\mathbf{J}^\top\mathbf{s}^{(t-\Delta t)}\,du\,dv, \quad (29)$$

where the superscripts denote evaluation of the quantities at the indicated times, and where the remaining quantities are evaluated at time $t+\Delta t$. For example, we can extrapolate the mass matrix using the formula

$$\mathbf{M}^{(t+\Delta t)} = \mathbf{M}^{(t)} + \Delta t\dot{\mathbf{M}}^{(t)} = 2\mathbf{M}^{(t)} - \mathbf{M}^{(t-\Delta t)} \quad (30)$$

and likewise for the other matrices and vectors in (29). The simpler, constant extrapolations $\mathbf{M}^{(t+\Delta t)} = \mathbf{M}^{(t)}$, etc., ([15] Section 8.6) also work satisfactorily.

In the interest of efficiency, we do not factorize the matrix expression on the left hand side of (29) in order to solve for $\mathbf{p}^{(t+\Delta t)}$. Instead, we employ the conjugate gradient method to obtain an iterative solution [27, 32]. 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 dramatically during interactive sculpting.

Note that when physical parameter values are chosen such that the equations (17) are not stiff, it is much cheaper to employ an explicit time integration method using forward differences. Appendix D discusses the forward difference approach. Note that the explicit method requires values for the matrices only at time t, hence (30) is not needed.

For the D-NURBS curve, we simply replace $\mathbf{c}$ with $\mathbf{s}$ in (29) and everything proceeds as in the case of surfaces.

In the case of D-NURBS with linear constraints, we discretize the derivatives of $\mathbf{q}$ (rather than $\mathbf{p}$). Analogous to (29), the discrete version of (22) is

$$\begin{aligned}(4\mathbf{M}_q + 2\Delta t\mathbf{D}_q + 4\Delta t^2\mathbf{K}_q)\mathbf{q}^{(t+\Delta t)} &= \\ 4\Delta t^2(\mathbf{f}_q + \mathbf{g}_q) + 8\mathbf{M}_q\mathbf{q}^{(t)} - (3\mathbf{M}_q - 2\Delta t\mathbf{D}_q)\mathbf{q}^{(t-\Delta t)} - \mathbf{G}^\top\mathbf{M}\mathbf{q}_0 &+ \iint \mu\mathbf{L}^\top\mathbf{s}\,du\,dv.\end{aligned} \quad (31)$$

Since there are fewer degrees of freedom in $\mathbf{q}$ than in $\mathbf{p}$, faster numerical implementation of constrained D-NURBS is possible, provided the constraint matrix $\mathbf{G}$ is sparse. Note that since the conjugate gradient algorithm requires only gradient vectors, we need not compute $\mathbf{M}_q$, $\mathbf{D}_q$ and $\mathbf{K}_q$ explicitly. The only extra cost is the computation of $\mathbf{G}\mathbf{q}$ and the multiplication of $\mathbf{G}$ with several vectors in (31).

For nonlinear constraints, at each time step we can apply the conjugate gradient algorithm to solve (27) for the Lagrange multipliers $\boldsymbol{\lambda}$ and the constrained generalized accelerations $\ddot{\mathbf{p}}$ (given known $\mathbf{p}$ and $\dot{\mathbf{p}}$). We then integrate $\ddot{\mathbf{p}}$ and $\dot{\mathbf{p}}$ from t to $t + \Delta t$ to obtain the constrained generalized velocities $\dot{\mathbf{p}}$ and coordinates $\mathbf{p}$ (e.g., using the simple Euler method $\dot{\mathbf{p}}^{(t+\Delta t)} = \dot{\mathbf{p}}^{(t)} + \Delta t\ \ddot{\mathbf{p}}^{(t)}$; $\mathbf{p}^{(t+\Delta t)} = \mathbf{p}^{(t)} + \Delta t\ \dot{\mathbf{p}}^{(t+\Delta t)}$).

6.4 Simplifications

The above implementation strategy permits real-time simulation of the general D-NURBS model on midrange graphics workstations. Lengthy curves can be simulated at interactive rates, as can quadratic and cubic surfaces on the order of 10×10 control points. It is possible to make simplifications that further reduce the computational expense of (29) and (31), making it practical to work with larger D-NURBS surfaces.

First, it is seldom necessary to simulate the fully general D-NURBS model throughout an entire sculpting session. Once we freeze the values of the weights $\mathbf{p}_w$, all of the matrices in (17) and (22) are constant and their entries need no longer be recomputed at each time step. With this restricted rational generalization of the B-splines, interactive rates are readily obtained for much larger surfaces with up to an order of magnitude more degrees of freedom. Note that D-NURBS reduce to dynamic B-splines if all components of the frozen vector $\mathbf{p}_w$ are, in addition, equal to 1.

Second, a full implementation of (17) is appropriate if the models must respond with realistic dynamics. However, in certain CAGD and surface-fitting applications where the modeler is interested only in the final equilibrium configuration of the model, it makes sense to simplify (17) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish. This economizes on storage and makes the algorithm more efficient. With zero mass density, (17) reduces to

$$\mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p, \tag{32}$$

while (22) reduces to

$$\mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{h}_q. \tag{33}$$

Discretizing the derivatives of $\mathbf{p}$ and $\mathbf{q}$ in (32) and (33) with backward differences, we obtain the integration formulas

$$(\mathbf{D} + \Delta t\mathbf{K})\,\mathbf{p}^{(t+\Delta t)} = \Delta t\mathbf{f}_p + \mathbf{D}\mathbf{p}^{(t)} \tag{34}$$

and

$$(\mathbf{D}_q + \Delta t\mathbf{K}_q)\,\mathbf{q}^{(t+\Delta t)} = \Delta t(\mathbf{f}_q + \mathbf{h}_q) + \mathbf{D}_q\mathbf{q}^{(t)} \tag{35}$$

respectively.

7 Modeling Environment and Applications

This section describes our D-NURBS modeling environment and presents several applications of D-NURBS relating to trimming, solid rounding, optimal curve and surface fitting, cross-sectional design of shapes, free-form deformation, and shape metamorphosis.

7.1 Interactive Modeling Environment

We have developed a prototype modeling environment based on the D-NURBS model. The system is written in C and it currently runs under Iris Explorer on Silicon Graphics workstations. Our parallelized iterative numerical algorithm takes advantage of a 4D/340VGX 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. They may be combined with existing Explorer modules for data input and object rendering.

Using our system, designers can sculpt shapes in conventional geometric ways, such as by sketching control polygons, repositioning control points, and adjusting associated weights. They can also satisfy design requirements by adjusting the D-NURBS internal physical parameters, various applied-force terms, and constraints. Physical parameters such as the mass, damping, and stiffness densities, and force gain factors are interactively adjustable through Explorer control panels.[4]

Our D-NURBS system also implements, as special cases, rational B-splines (fixed weights values) and ordinary B-splines (unit weights), hence it encompasses three related free-form modeling schemes into one unified physics-based implementation. The following sections describe several applications.

7.2 Trimming Curves and Surfaces

The physical basis of the D-NURBS model and our numerical quadrature approach to computing the mass, damping, and stiffness matrices (Section 6.2) suggests a straightforward technique for trimming D-NURBS curves and surfaces. Surfaces may be trimmed with arbitrary curves defined in the parametric domain, including D-NURBS curves. The trimming of D-NURBS is directly analogous to the trimming of excess material from real-world deformable wires and sheets.

Consider a D-NURBS patch that is intersected by a trimming curve. The values of material properties—mass, damping, elasticity densities—over the portion of the patch that extends outside the trimming curve should not affect the dynamics of the trimmed model and are set to zero. The Gauss quadrature proceeds normally, but abscissas that sample zero physical parameters make no contribution to the summation. Of course, a patch may be disregarded if it falls completely outside the trimming boundary. Note that the integrands are discontinuous at the boundary due to the sudden transition of the the physical parameter values. While this does not destroy the correctness of Gauss quadrature, we can expect reduced accuracy since the integrand is not smooth. There is no easy way around this potential problem for arbitrary boundary curves, other than to use Monte Carlo integration and pay the penalty of slow asymptotic convergence [27]. Fortunately, in practice, the D-NURBS model appears tolerant of the reduced integration accuracy in boundary elements.

Fig. 1 illustrates the trimming of D-NURBS surfaces using D-NURBS trimming curves in the parametric domain. Fig. 1(a) shows the creation of a triangular surface with three linear curves each with 4 control points. Fig. 1(b) shows a trimmed annular surface defined by two circular trimming curves each with 25 control points. Snapshots are shown of the trimmed surfaces undergoing dynamic deformations in response to applied forces.

7.3 Solid Rounding

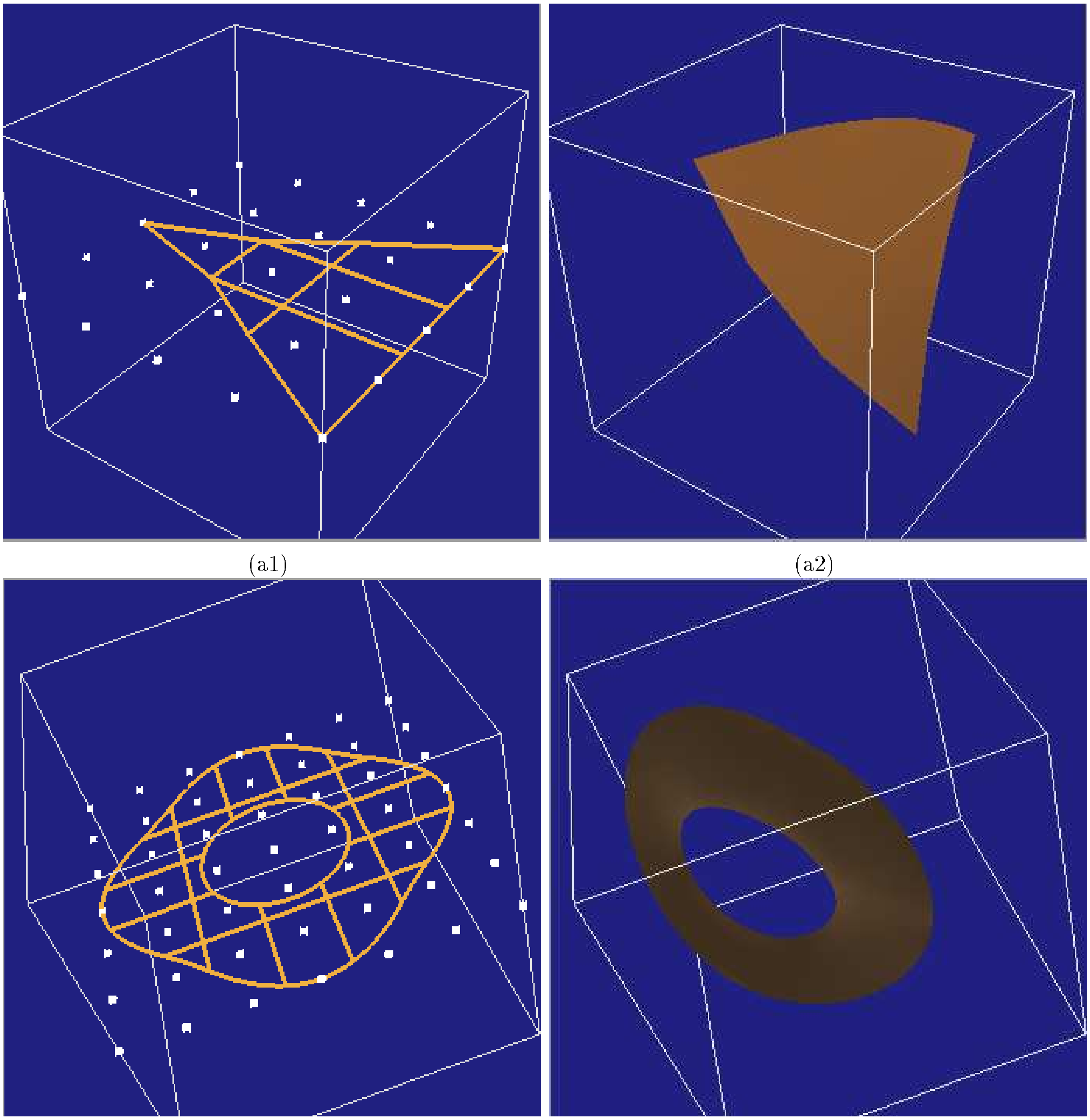
The rounding of solids is a common operation for the design of mechanical parts. A goal of this operation is to construct a fillet surface that smooths by interpolating between two or more surfaces. In geometric modeling, this is usually done by enforcing parametric or geometric continuity requirements on the fillet.

D-NURBS provide a natural solution to the solid rounding problem. In contrast to the geometric approach, the D-NURBS can produce a smooth fillet with the proper continuity requirements by minimizing its internal deformation energy. Additional position and normal constraints may be imposed across the boundary of the surface. The dynamic simulation automatically produces the desired final shape.

Fig. 2 demonstrates edge rounding using D-NURBS surfaces. In Fig. 2(a1), we round an edge at the intersection of two planar faces. The faces are formed using quadratic D-NURBS patches with 8×5 control points. Multiple control points are used to produce the sharp corner. We free the control points near the corner and fix the remaining control points at the far boundaries to impose position and surface normal constraints. After initiating the physical simulation, the D-NURBS rounds the corner as it achieves the minimal energy equilibrium state shown in Fig. 2(a2).

Fig. 2(b1) illustrates the rounding of a trihedral corner of a cube. The corner is represented using a quadratic D-NURBS surface with 6×5 control points. The corner is rounded with position and normal

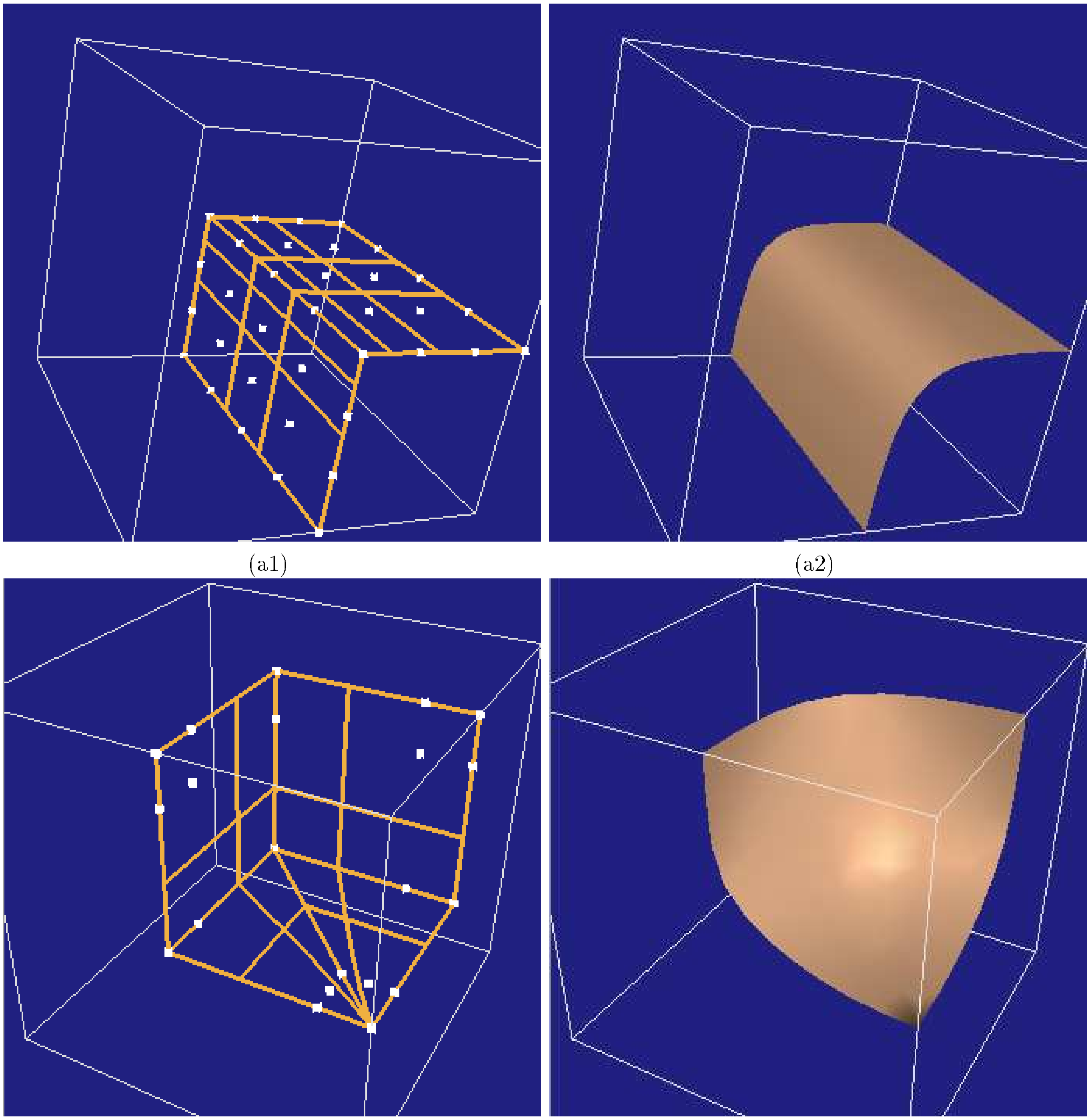
[4]At present, our software assumes uniform mass, damping, and elasticity densities over the parametric domain, except across trimming boundaries (see Section 7.2). This is straightforwardly generalizable to accommodate the nonuniform density functions in our formulation, although our user interface would have to be extended to afford the user full control in specifying these functions.



(a1) (a2)

(b1) (b2)

Figure 1: Trimming D-NURBS surfaces: (a) triangular D-NURBS surface; (b) annular D-NURBS surface. (a1) Patch outlines and control points (white) with linear trimming curves. (a2) Interactive dynamic deformation of trimmed triangular surface. (b1) Patch outlines and control points with concentric trimming curves. (b2) Interactive dynamic deformation of annular surface.



(a1) (a2) (b1) (b2)

Figure 2: Solid rounding: (a) rounding an edge between polyhedral faces; (b) rounding a trihedral vertex. (a1) Initial configuration of control points and patches. (a2) Rounded D-NURBS surface in static equilibrium. (b1) Initial configuration of control points and patches. (b2) Rounded D-NURBS surface. In both examples, the control points along edges have multiplicity 2.

constraints along the far boundaries of the faces (Fig. 2(b2)).

The above rounding technique is easily extensible to any number of surfaces meeting at arbitrary angles. To round a complete solid, we can apply the technique to all of its edges, corners, etc.

7.4 Optimal Surface Fitting

D-NURBS are applicable to the optimal fitting of regular or scattered data [28]. 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 [35, 33]. 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 $\mathbf{p}_w$ are fixed, and it is amenable to the constraint techniques presented in Section 5.2. 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.[5]

We present three examples of surface fitting using D-NURBS coupled 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 D-NURBS surface with 25 control points.

7.5 Cross-Sectional Design

Cross-sectional design is a common approach to shaping surfaces and solids using cross-sectional curves. Our modeling system provides the modeler with D-NURBS generator curves along with the most useful surface generator operators—sweeping and swinging [23]—for generating common surfaces such as extruded surfaces, natural quadrics, general quadrics, ruled surfaces, and surfaces of revolution. In our current implementation, the modeler can indirectly sculpt the composite surfaces by direct dynamic manipulation of the D-NURBS generator curves subject to constraints. Geometric constraints such as positions and normals may be associated with D-NURBS curves.

We present three examples in the cross-sectional design of surfaces. First, Fig. 4 shows a generalized cylinder with 30 control points created by sweeping a green closed curve with 6 control points along the red curve with 5 control points (Fig. 4(a)). The generalized cylinder is interactively sculpted into various shapes by applying spring forces on the green and red cubic D-NURBS curves (Fig. 4(b-d)). Second, Fig. 5 shows a torus with 49 control points generated by swinging the green curve over the red curve (Fig. 5(a)). Both generators are closed cubic D-NURBS curves with 7 control points. In Fig. 5(b-d), the torus is deformed interactively by applying a spring force. Third, Fig. 6 shows a 35 control point "wine glass" shape obtained by sweeping the green generator curve on the red generator curve in Fig. 6(a). The red closed D-NURBS curve has 7 control points and the green open D-NURBS curve has 5 control points. The glass is interactively sculpted into different swept shapes using spring forces (Fig. 6(b-d)).

[5] Cross-validation [41] provides a principled approach to choosing the relevant physical parameters—typically the ratio of data force spring constants to surface stiffnesses—for given data sets. For the special case of zero-mean Gaussian data errors, optimal approximation in the least squares residual sense results when c is proportional to the inverse variance of data errors [35].

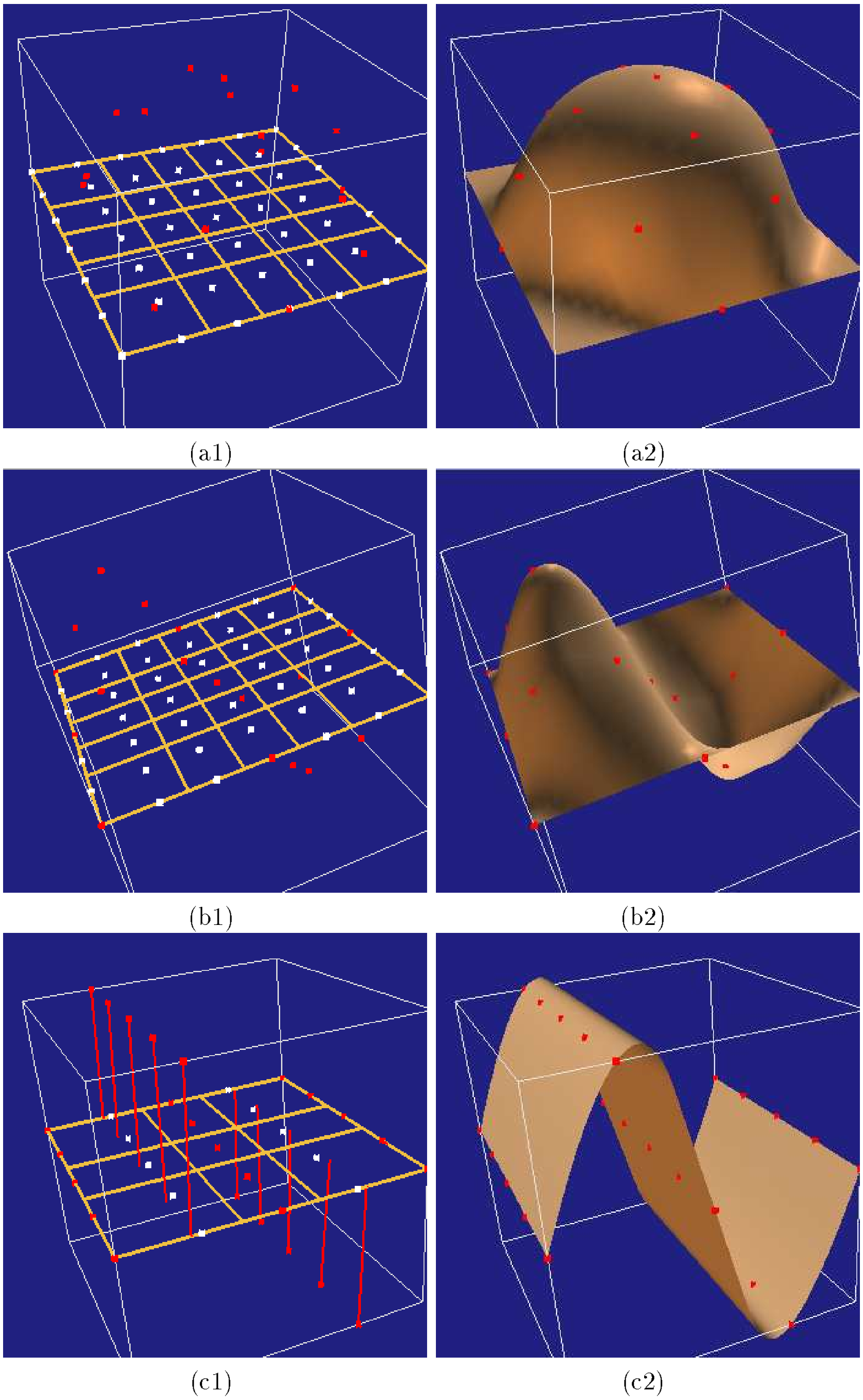


Figure 3: Optimal surface fitting: D-NURBS surfaces fit to sampled data from (a) a hemisphere, (b) a convex/concave surface, (c) a sinusoidal surface. (a–c1) D-NURBS patch outline with control points (white) and data points (red) shown. (a–c2) D-NURBS surface at equilibrium fitted to scattered data points.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

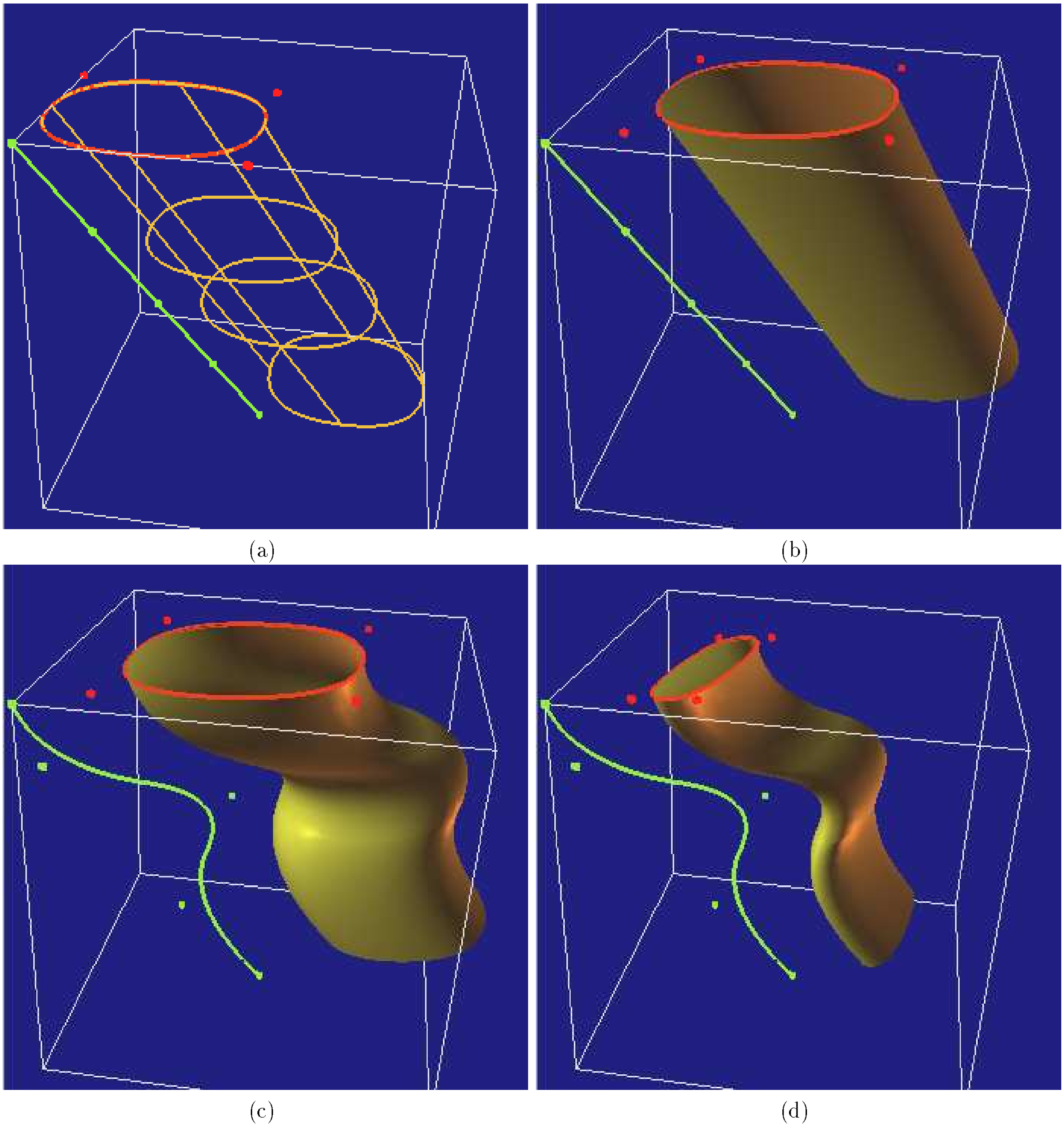
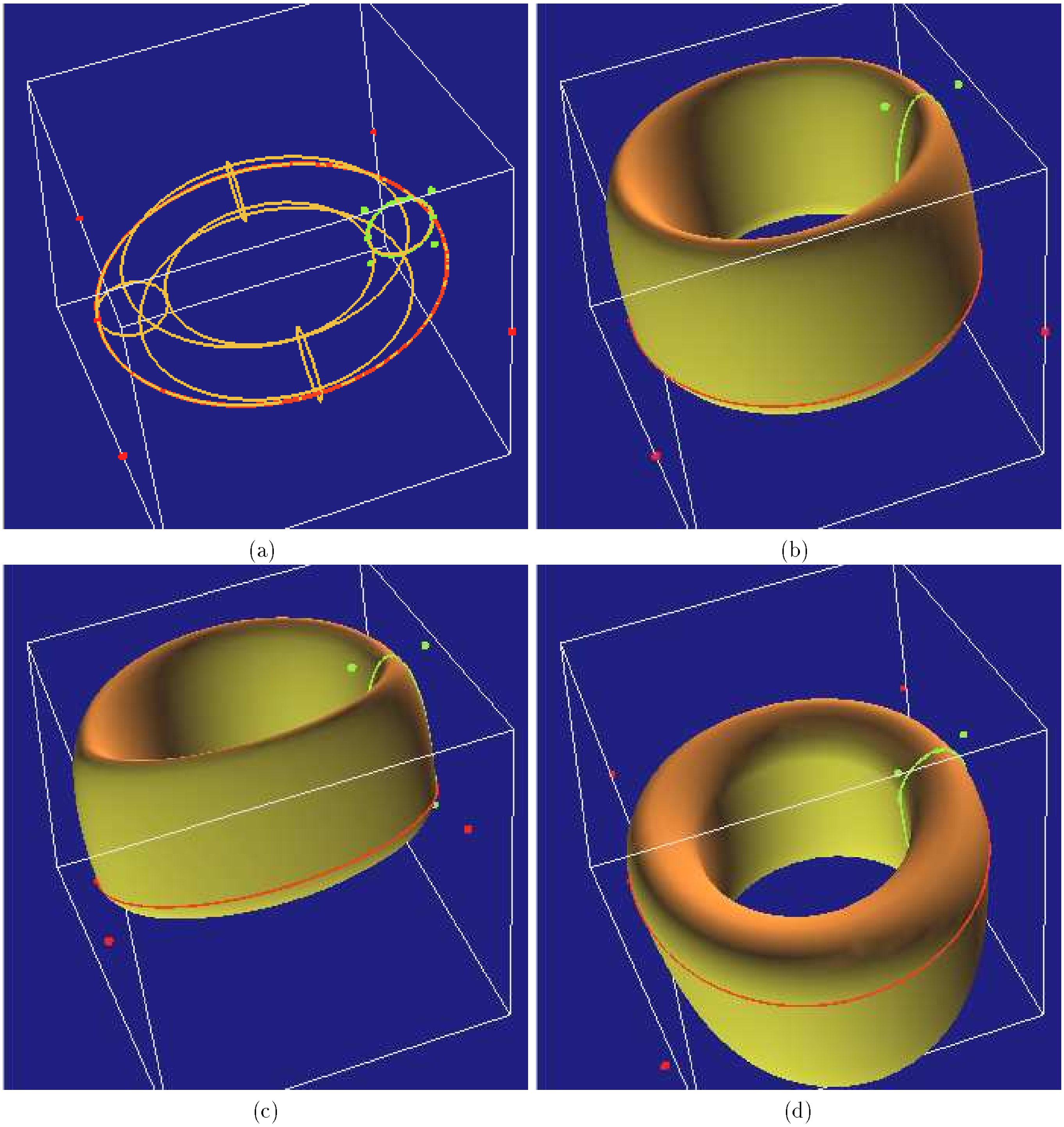
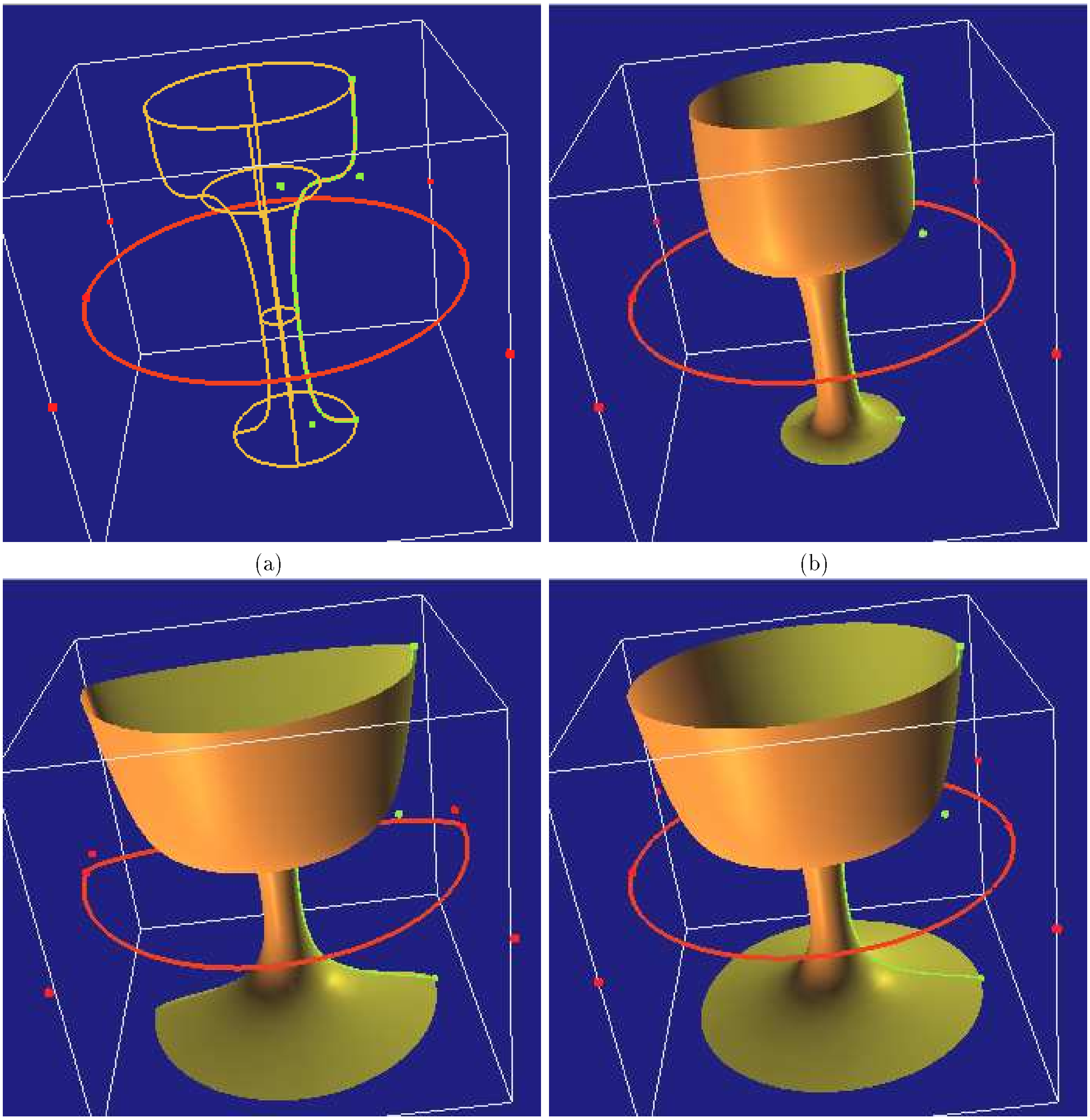


Figure 4: Interactive deformation of generalized cylinder. (a) Patch outline of generalized cylinder created from two D-NURBS generating curves (control points shown) using sweep operation. (b–d) Interactive dynamic deformation of either generating curve causes global deformation of generalized cylinder.



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

Figure 5: Interactive deformation of torus. (a) Two D-NURBS generating curves with control points shown and patch outline of torus generated by swing operation. (b–d) Interactive dynamic deformation of either generating curve causes global deformation of torus.



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

Figure 6: Creation and deformation of "wine glass." (a) Two D-NURBS generating curves with control points and patch outlines of glass formed by swing operation. (b–d) Deformation of glass caused by interactive dynamic deformation of D-NURBS generators.

7.6 Shape Metamorphosis

Metamorphosis is the blending of one shape into another. Work on 3D shape blending includes [8, 16]. The blending of 2D shapes has widespread application in illustration, animation, etc., and simple (e.g., linear) interpolation techniques usually produce unsatisfactory results [29]. Shinagawa and Kunii [31] propose an method which interpolates differential properties of the 2D shape using the elastic surfaces of [37, 36]. Motivated by their approach, we propose a new technique which exploits the properties of D-NURBS surfaces. D-NURBS provide minimal-energy blends which are more general than linear interpolants and which may be controlled through various additional constraints specific to the NURBS geometry. For example, since NURBS can represent conics, we can exploit their ability to generate helical surfaces in order to represent rotational components of shape metamorphoses.

Our technique interpolates a D-NURBS generalized cylinder between two or more planar shapes with known correspondence. The interpolant is a constrained skinned surface between the two end curves. We interpret the parametric coordinate along the length of the surface, say u, as the (temporal) shape blending parameter. The u coordinates of the control points are fixed, while the v coordinates are subject to the D-NURBS deformation energy and additional constraints. We obtain intermediate shapes by evaluating cylinder cross sections at arbitrary values of u.

Some examples will help to explain our technique in more detail. Fig. 7 shows minimal-energy D-NURBS surfaces with 3×6 control points (3 control points along u) interpolating between two closed elliptical curves. Fig. 7(b) shows a linear generalized cylinder obtained with high surface tension in the u direction: $\alpha_{1,1} = 1000$ and $\alpha_{2,2} = \beta_{i,j} = 0$. Note that the morphing ellipse shrinks as it rotates, a typical artifact of linear interpolation [29]. The rotational component can be preserved, however, by imposing a geometric constraint on the D-NURBS which creates a helical surface in the u direction of the cylinder, as shown in Fig. 7(c). Here the only nonzero deformation energy parameter is the rigidity $\beta_{1,1} = 1000$. Note that the interpolating surface now bulges outside the convex hull between the two ellipses. As a consequence the interpolated ellipses rotate instead of shrinking (Fig. 7(d)). In general, we can obtain a family of blending surfaces between these two extremes by using intermediate values of tension $\alpha_{1,1}$ and rigidity $\beta_{1,1}$ parameters. Fig. 8 illustrates the morphing between two planar polygonal shapes. The D-NURBS interpolant is a 3×7 surface. The parts of this figure are similar to those of the previous one.

7.7 Free-Form Deformation

Bezier introduced the idea of globally deforming a shape through a $\Re^n \to \Re^n$ mapping implemented as a free-form (tensor product) spline. The shape is embedded in the spline and deformed by manipulating the spline's control points. Sederberg and Parry [30] popularized this concept of free-form deformation (FFD) in the graphics literature.

We can arrive at a physics-based version of the FFD in which the object to be deformed is embedded in the D-NURBS "material" and deforms along with the deforming D-NURBS. The physics-based deformation is similar in motivation to the one devised in [7], but it offers fully continuous dynamics by virtue of the continuous nature of D-NURBS. In particular, we can apply forces at arbitrary points in the D-NURBS space to control the deformation directly (rather than through indirect manipulation via control points).

8 Conclusion

We have developed dynamic NURBS, a physics-based generalization of the well-known geometric NURBS curves and surfaces. D-NURBS were derived systematically through the application of Lagrangian mechanics and implemented using concepts from finite element analysis and efficient numerical methods. We generalized our D-NURBS formulation to incorporate geometric constraints. The formulation extends naturally to solids, albeit at proportionately greater computational cost.

We described a prototype interactive modeling system based on D-NURBS and demonstrated the

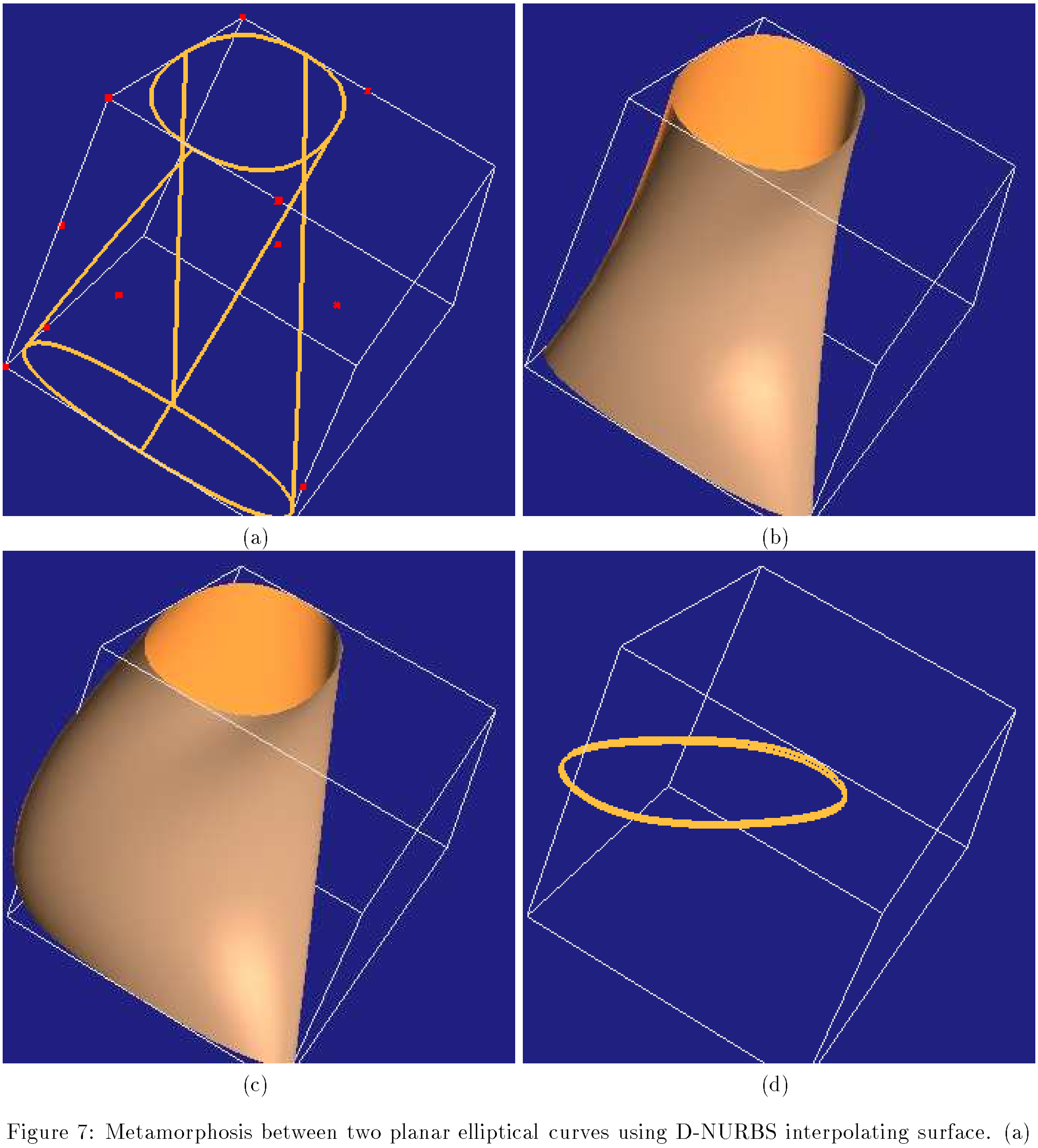


Figure 7: Metamorphosis between two planar elliptical curves using D-NURBS interpolating surface. (a) Control points and patch outline of cylindrical surface terminated by the two planar curves. (b) Linear interpolating surface. (c) Constrained nonlinear interpolating surface combines rigid rotation with nonrigid deformation. (d) An intermediate morphed curve obtained as cross section of surface in (c).

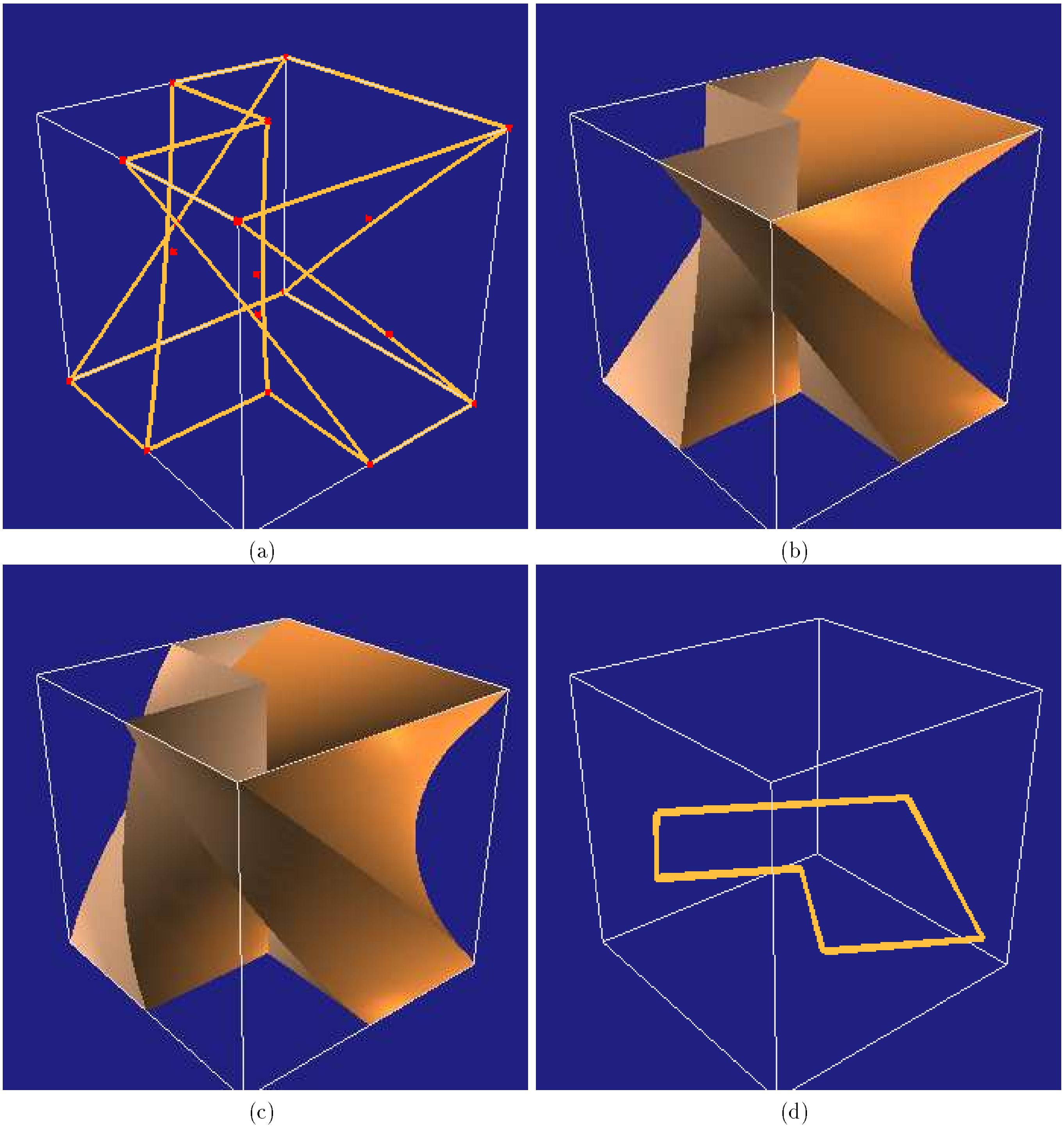


Figure 8: Metamorphosis between two planar polygonal curves using D-NURBS interpolating surface. (a) Control points and patch outlines of cylindrical surface terminated by the two planar curves. (b) Linear interpolating surface. (c) Constrained nonlinear interpolating surface combines rigid rotation with nonrigid deformation. (d) An intermediate morphed curve obtained as cross section of surface in (c).

flexibility of our models in a variety of applications. When working with D-NURBS, a designer need not manipulate the individual degrees of freedom of an object. Instead, the designer can work with sculpting tools that are implemented in terms of forces and geometric constraints. Sculpting forces may be applied interactively to move the object or refine its shape. The interactive response of the D-NURBS may be modified by varying its mass and damping distributions. Global design requirements may also be achieved by varying physical parameters such as elastic energies.

Because NURBS have been assimilated into such industry standards such as IGES, PHIGS+, and OpenGL, our dynamic NURBS model promises to forge stronger links between established CAGD methodologies and new techniques in physics-based modeling.

Acknowledgements

Funding for this research was provided in part by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario.

A Position Equation

Clearly,

$$\mathbf{J}\mathbf{p} = \mathbf{B}\mathbf{p}_b + \mathbf{W}\mathbf{p}_w$$

and

$$\mathbf{c}(\mathbf{p}, u) = \mathbf{B}\mathbf{p}_b.$$

To prove (6), we must show that (5) holds true. By definition,

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

Exchanging the summation order and indexes, we have

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

which proves (5), hence (6).

Moreover, taking the time derivative of (6) yields

$$\dot{\mathbf{c}}(u, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}} + \dot{\mathbf{J}}\mathbf{p}.$$

Given (4), it follows that $\dot{\mathbf{J}}\mathbf{p} = \mathbf{0}$.

B Simplification of Motion Equations

Applying (10), the D-NURBS motion equations are

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{array} \right]^\top - \dot{\mathbf{M}}\dot{\mathbf{p}} - \left[\begin{array}{ccc} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial p_i}\mathbf{p} & \cdots \end{array} \right]^\top. \tag{36}$$

The two vectors involving $\mathbf{M}$ on the right side of (36) may be be combined into a single vector:

$$\dot{\mathbf{M}}\dot{\mathbf{p}} - \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{array} \right]^\top = \mathbf{I}\dot{\mathbf{p}}. \tag{37}$$

Using the product rule of differentiation, we have $\dot{\mathbf{M}} = \mathbf{I} + \mathbf{I}^\top$. For (37) to hold, we must have

$$\mathbf{I}^\top \dot{\mathbf{p}} = \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{array} \right]^\top. \tag{38}$$

It is obvious from (11) that the two sides of (38) are integrals of the two vectors, respectively. The two vectors in (38) are equal when, for $i = 1, \ldots, N$,

$$\dot{\mathbf{j}}_i^\top \mathbf{J}\dot{\mathbf{p}} = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial(\mathbf{J}^\top \mathbf{J})}{\partial p_i}\dot{\mathbf{p}}. \tag{39}$$

We now prove (39). The right side is represented as R. Based on the product rule of differentiation and the property of the Jacobian matrix, we obtain the simpler expression

$$R = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{J}^\top}{\partial p_i}\mathbf{J}\dot{\mathbf{p}} + \frac{1}{2}\dot{\mathbf{p}}^\top \mathbf{J}^\top \frac{\partial \mathbf{J}}{\partial p_i}\dot{\mathbf{p}}.$$

Furthermore, according to the property of the Jacobian matrix and the observation that we can interchange the order of the cross derivatives

$$\frac{\partial \mathbf{J}}{\partial p_i}\dot{\mathbf{p}} = \dot{\mathbf{j}}_i.$$

Combining the above two expressions we obtain

$$R = \frac{1}{2}\dot{\mathbf{j}}_i^\top \mathbf{J}\dot{\mathbf{p}} + \frac{1}{2}\left(\dot{\mathbf{j}}_i^\top \mathbf{J}\dot{\mathbf{p}}\right)^\top.$$

Since R is a scalar, (39) is proved.

Next, we derive another mathematical identity:

$$\left[\begin{array}{ccc} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial p_i}\mathbf{p} & \cdots \end{array}\right]^\top = \mathbf{0}. \tag{40}$$

The left side of (40) is the integral of the summation of the five terms of (16). Each of these five vectors is the zero vector. To see this, note that for $i = 1, \ldots, N$, we have

$$\frac{\partial \mathbf{s}}{\partial p_i} = \frac{\partial \mathbf{J}}{\partial p_i}\mathbf{p} + \mathbf{J}\frac{\partial \mathbf{p}}{\partial p_i} = \frac{\partial \mathbf{J}}{\partial p_i}\mathbf{p} + \mathbf{j}_i$$

According to the definition of the Jacobian matrix, the left hand side is $\mathbf{j}_i$, $i = 1, \ldots, N$. Thus, we have

$$\frac{\partial \mathbf{J}}{\partial p_i}\mathbf{p} = \mathbf{0}.$$

The order of the second cross derivative with respect to the variables p_i and u is irrelevant, so we further have

$$\mathbf{p}^\top \frac{\partial}{\partial p_i}\left(\frac{\partial \mathbf{J}^\top}{\partial u}\frac{\partial \mathbf{J}}{\partial u}\right)\mathbf{p} = 0. \tag{41}$$

Now, (41) is the ith component of the first vector on the left side of (40). Similarly, the other four vectors inside the integral operator on the left hand side of (40) are zero.

C Simplification of Motion Equations with Linear Constraints

Applying (10), the D-NURBS motion equations with linear constraints are

$$\mathbf{M}_q\ddot{\mathbf{q}} + \mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} - \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{array}\right]^\top + \left[\begin{array}{ccc} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial q_j}\mathbf{p} & \cdots \end{array}\right]^\top = \mathbf{f}_q - \mathbf{G}^\top(\dot{\mathbf{M}}\dot{\mathbf{p}} + \mathbf{K}\mathbf{q}_0). \tag{42}$$

To simplify (42) we first show that it reduces to the following

$$\mathbf{G}^\top\dot{\mathbf{M}}\dot{\mathbf{p}} - \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{array}\right]^\top = \mathbf{G}^\top\mathbf{I}\dot{\mathbf{p}}. \tag{43}$$

As in Appendix B, $\dot{\mathbf{M}} = \mathbf{I} + \mathbf{I}^\top$. Hence, (43) is also expressed as

$$\mathbf{G}^\top\mathbf{I}^\top\dot{\mathbf{p}} = \left[\begin{array}{ccc} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{array}\right]^\top. \tag{44}$$

Similar to (38), the two sides of (44) are integrals of two vectors, respectively. Hence, (44) holds if corresponding components of the two vectors are equal; i.e., for $j = 1, \ldots, M$,

$$\dot{\mathbf{l}}_j^\top \mathbf{J}\dot{\mathbf{p}} = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial(\mathbf{J}^\top\mathbf{J})}{\partial q_j}\dot{\mathbf{p}}. \tag{45}$$

We now prove (45). Denoting the right side as R, we further expand it using the product rule of differentiation

$$R = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{J}^\top}{\partial q_j}\mathbf{J}\dot{\mathbf{p}} + \frac{1}{2}\dot{\mathbf{p}}^\top \mathbf{J}^\top \frac{\partial \mathbf{J}}{\partial q_j}\dot{\mathbf{p}}.$$

Furthermore, according to the property of the Jacobian matrix and the irrelevance of the order of differentiation, we have

$$\frac{\partial \mathbf{J}}{\partial q_j}\dot{\mathbf{p}} = \frac{\partial}{\partial q_j}\begin{bmatrix} \cdots & \frac{\partial \mathbf{s}}{\partial p_i} & \cdots \end{bmatrix}\dot{\mathbf{p}} = \dot{\mathbf{l}}_j.$$

Combining the above two equations, we have

$$R = \frac{1}{2}\dot{\mathbf{l}}_j^\top \mathbf{J}\dot{\mathbf{p}} + \frac{1}{2}\left(\dot{\mathbf{l}}_j^\top \mathbf{J}\dot{\mathbf{p}}\right)^\top.$$

Since R is a scalar, (45) follows.

The proof of

$$\begin{bmatrix} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial q_j}\mathbf{p} & \cdots \end{bmatrix}^\top = \mathbf{0} \tag{46}$$

parallels that in Appendix B, with q_j replacing p_j and $\mathbf{l}_j$ replacing $\mathbf{j}_i$.

D Explicit Time Integration

We discretize the motion equations using the following finite differences in $\mathbf{p}$ ($\mathbf{q}$ in the case of geometric constraints):

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

$$\dot{\mathbf{p}}^{(t)} = \frac{\mathbf{p}^{(t)} - \mathbf{p}^{(t-\Delta t)}}{\Delta t}.$$

We obtain the discrete form of (17) as

$$\begin{aligned}\mathbf{M}\mathbf{p}^{(t+\Delta t)} &= \Delta t^2(\mathbf{f}_p - \mathbf{K}\mathbf{p}^{(t)}) + 2\mathbf{M}\mathbf{p}^{(t)} - \Delta t\mathbf{D}\mathbf{p}^{(t)} \\ &+\Delta t\mathbf{D}\mathbf{p}^{(t-\Delta t)} - \iint \mu \mathbf{J}^\top \mathbf{s}^{(t-\Delta t)}\, du\, dv.\end{aligned} \tag{47}$$

In this and the following explicit time integration schemes, all the matrices are evaluated at time t (instead of time $t + \Delta t$ as in the implicit schemes).

For D-NURBS surfaces with linear geometric constraints, (22) is discretized as

$$\begin{aligned}\mathbf{M}_q\mathbf{q}^{(t+\Delta t)} &= \Delta t^2(\mathbf{f}_q + \mathbf{g}_q - \mathbf{K}_q\mathbf{q}^{(t)}) + 2\mathbf{M}_q\mathbf{q}^{(t)} - \Delta t\mathbf{D}_q\mathbf{q}^{(t)} + \\ &\Delta t\mathbf{D}_q\mathbf{q}^{(t-\Delta t)} + \mathbf{G}^\top\mathbf{M}\mathbf{q}_0^{(t-\Delta t)} - \iint \mu \mathbf{L}^\top \mathbf{s}^{(t-\Delta t)}\, du\, dv.\end{aligned} \tag{48}$$

For the D-NURBS curve, we substitute $\mathbf{c}$ with $\mathbf{s}$ in (47) and (48).

The discretized forms of the simplified first order equations of motion (32) and (33) are

$$\mathbf{D}\mathbf{p}^{(t+\Delta t)} = \Delta t(\mathbf{f}_p - \mathbf{K}\mathbf{p}^{(t)}) + \mathbf{D}\mathbf{p}^{(t)} \tag{49}$$

and

$$\mathbf{D}_q\mathbf{q}^{(t+\Delta t)} = \Delta t(\mathbf{f}_q + \mathbf{h}_q - \mathbf{K}_q\mathbf{q}^{(t)}) + \mathbf{D}_q\mathbf{q}^{(t)}. \tag{50}$$

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] W. Bohm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

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

[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] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered construction for animated deformable characters. *Computer Graphics*, 23(3):243–252, 1989.

[8] S.E. Chen and R. Parent. Shape averaging and its applications to industrial design. *IEEE Computer Graphics and Applications*, 9(1):47–54, 1989.

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

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

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

[12] G. Farin. From conics to NURBS: A tutorial and survey. *IEEE Computer Graphics and Applications*, 12(5):78–86, Sept. 1992.

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

[16] A. Kaul and J. Rossignac. Solid interpolating deformations: Construction and animation of PIPs. In F.H. Post and W. Barth, editors, *Proc. Eurographics '91*, pages 493–505, Amsterdam, 1991. North Holland.

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

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

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

[20] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, 1989.

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

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

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

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

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

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

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

[28] 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.

[29] T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2-D shape blending: An intrinsic solution to the vertex path problem. In *Computer Graphics* Proceedings, Annual Conference Series, Proc. ACM Siggraph'93 (Anaheim, CA, Aug., 1993), pages 15–18, 1993.

[30] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric primitives. *Computer Graphics*, 20(4):151–160, 1986.

[31] Y. Shinagawa and T.L. Kunii. The differential model: A model for animating transformation of objects using differential information. In T.L. Kunii, editor, *Modeling in Computer Graphics*, pages 6–15. Springer-Verlag, Tokyo, 1991.

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

[33] R. Szeliski and D. Terzopoulos. From splines to fractals. *Computer Graphics*, 23(3):51–60, 1989.

[34] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, 1992.

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

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

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

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

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

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

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

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

[43] O.C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, London, third edition, 1977.