D-NURBS:
Dynamic Non-Uniform Rational B-Splines

Hong Qin

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy,
Graduate Department of Computer Science, in the
University of Toronto

D-NURBS: Dynamic Non-Uniform Rational B-Splines

Ph.D. 1995

Hong Qin

Graduate Department of Computer Science

University of Toronto

Abstract

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 some common analytic shapes. To date, 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. Despite modern interactive devices, this conventional shape modification process can be clumsy and laborious when it comes to designing complex, real-world objects.

This thesis paves the way for NURBS to achieve their full potential, by putting the laws of physics on their side. The thesis proposes, develops, and applies dynamic NURBS (D-NURBS), a physics-based generalization of the NURBS representation. D-NURBS unify the features of the industry-standard geometry with the many demonstrated conveniences of interaction through physical dynamics.

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 results from the numerical integration of a set of nonlinear differential equations that automatically evolve the geometric DOFs in response to the applied forces and constraints to produce physically meaningful, hence highly intuitive shape variation. Consequently, a modeler can interactively sculpt complex shapes to required specifications not only in the traditional indirect fashion, by adjusting control points, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. An important advantage of this physics-based framework is that existing geometric toolkits continue to be applicable at the basic geometry level, while it also affords designers new force-based toolkits that support dynamic manipulation and interactive sculpting at the physics level.

The mathematical development consists of four related parts: (i) D-NURBS curves, (ii) tensor product D-NURBS surfaces, (iii) swung D-NURBS surfaces, and (iv) triangular D-NURBS surfaces. We use Lagrangian mechanics to formulate the equations of motion for all four varieties, and finite element analysis to reduce these equations to efficient algorithms that can be simulated at interactive rates using standard numerical techniques.

We implement a prototype modeling environment based on D-NURBS, demonstrating that D-NURBS are effective tools in a wide range of applications in CAD and graphics. We demonstrate shape blending, scattered data fitting, surface trimming, cross-sectional shape design, shape metamorphosis, and free-form deformation with geometric and physical constraints. Thus, D-NURBS provide a systematic and unified approach for a variety of CAD and graphics modeling problems such as constraint-based optimization, variational parametric design, automatic weight selection, shape approximation, user interaction, etc. They also support direct manipulation and interactive sculpting through the use of force-based manipulation tools, the specification of geometric constraints, and the adjustment of physical parameters such as mass, damping, and elasticity.

Acknowledgments

Foremost, I would like to acknowledge my gratitude to my supervisor, Professor Demetri Terzopoulos, for his guidance, inspiration, and encouragement over the past four years. I would like to thank him for accepting me as his student. He has been the best supervisor I could have hoped for, because he is not only an outstanding researcher but also a great advisor.

I would also like to thank the other departmental members of my examination committee, Professors Eugene Fiume, Allan Jepson, and James Stewart for their constructive critique of my thesis. I wish to thank my external committee members, Professor David Gossard of the MIT Mechanical Engineering Department and Professor George Will of the UofT Department of Civil Engineering for their interest in my research and insightful comments. The invaluable suggestions from all of the committee members have greatly improved this thesis.

I have benefited greatly from being a member of the vision and graphics groups at the University of Toronto. I acknowledge all of my fellow colleagues and friends in both the vision and graphics labs. Especially, but not exclusively, I wish to thank Tim McInerney, Sean Culhane, Zhengyan Wang, Xiaoyuan Tu, Radek Grzeszczuk, Yuencheng Lee Victor, David Tonnesen, Luiz Velho, Xuan Ju, Feng Lu, Yiming Ye, Gilbert Verghese, Dave Wilkes, and Richard Mann, for numerous valuable technical discussions as well as many distractions. Thanks also to many others in the department of Computer Science including my office mates: Mike Hutton, Karim Harzallah, Marc Ouellette, and Timothy Veatch.

I gratefully acknowledge financial assistance from the University of Toronto Open Fellowship Office, the Department of Computer Science, and support from the Information Technologies Research Center of Ontario and the Natural Sciences and Engineering Research Council of Canada via grants to Professor Terzopoulos.

I would also like to thank my parents for their love, support, and encouragement. Finally, I wish to thank my wife, Wei Zhao, for her love and sacrifice. She has always supported me, shared the good and bad moments with patience and understanding, and kept me strong and confident throughout the pursuit of my professional goals. I could not envision having completed this thesis and my degree without her.

This work is dedicated to Mom, Dad, and Wei

Contents

List of Figures

Chapter 1

Introduction

Geometric modeling is concerned with the mathematical representation of geometric entities and their application to the design of objects by computer. Geometric modeling is crucial to a variety of fields including computer-aided geometric design (CAGD), computer graphics, virtual reality, medical imaging, and computer vision. During the past several decades, numerous geometric formulations and design techniques have been proposed and developed for a large variety of geometric modeling applications.

In 1975 Versprille proposed Non-Uniform Rational B-Splines (NURBS) (Versprille, 1975). This shape representation for geometric design generalized the B-splines (Riesenfeld, 1973). NURBS quickly gained popularity and were eventually incorporated into several commercial modeling systems (Piegl, 1991). This is primarily because NURBS provide a unified mathematical formulation for representing not only free-form curves and surfaces like B-splines, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution, which NURBS can also represent exactly.

One can design a large variety of shapes using NURBS by adjusting the positions of control points, setting the values of associated weights, and modifying the knot vector distribution (Farin, 1989; Farin, 1992; Piegl and Tiller, 1987; Piegl, 1989a; Piegl, 1989b; Piegl, 1991; Rogers and Adlum, 1990; Tiller, 1983). This is known as indirect geometric manipulation, because the geometric shape is modified indirectly through the modification of its free parameters or degrees of freedom (DOFs). As we will explain momentarily, since NURBS are abstract and extremely flexible geometric representations, it can often be difficult for the designer to exploit their full potential through indirect manipulation.

This thesis proposes and develops *dynamic NURBS*, or D-NURBS, by associating physical dynamics with NURBS geometry. Our new physics-based framework for NURBS imbues standard geometric NURBS with some of the universally familiar behavior of real-world objects. Hence, intuitive physics-based design augments (rather than supersedes) standard geometry and geometric design, offering attractive new advantages. A detailed discussion about physics-based modeling can be found in the following sections.

1.1 Problem Statement

Among various geometric representations, NURBS have become a *de facto* industrial standard in commercial modeling systems because of their many superior properties, such as their ability to unify free-form and common analytic shapes. To date, however, NURBS have been viewed as purely geometric primitives. Because of the extraordinary flexibility of geometric NURBS, the conventional shape modification process becomes problematic:

- Traditional free-form geometric design is a kinematic process. Designers are 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. For example, design refinement with triangular B-splines and NURBS can be especially time-consuming due to the irregularity of control points and knot vectors.

- Shape design to satisfy required specifications by manual adjustment of available geometric degrees of freedom is often elusive, because relevant design tolerances are naturally specified in terms of shape and not in terms of control points and weights. 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 manual 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, or change a weight, or move two control points, or...?

- The design requirements of engineers and stylists can be very different. Whereas engineers focus on technical and functional issues, stylists emphasize aesthetically-driven conceptual design. Therefore, 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 may impose both local and global constraints on shape. The incremental manipulation of local shape parameters to satisfy both functional design and aesthetic criteria simultaneously is at best cumbersome and at worst unproductive.

- Stylists are often interested in geometric "theme variation." This requires geometric entities to be generated quickly and naturally. Unlike engineers, stylists are concerned more with the shape of the geometric object than with its underlying mathematical description. It is apparent that conventional interpolation/approximation techniques, which often generate computerized models from digitized data, may not be suitable to the time-varying requirements of stylists.

1.2 The Physics-Based Approach

Physics-based modeling methodology and finite element techniques provide a means to overcome some disadvantages of conventional geometric design. Free-form deformable models, which were introduced to computer graphics in (Terzopoulos et al., 1987) and further developed in (Terzopoulos and Fleischer, 1988; Platt and Barr, 1988; Szeliski and Terzopoulos, 1989; Pentland and Williams, 1989; Celniker and Gossard, 1991; Szeliski and Tonnesen, 1992; Metaxas and Terzopoulos, 1992) are particularly relevant in the context of modeling with NURBS. Physics-based models have proven to be superior for realistic animation of non-rigid figures such as human bodies, visual motion tracking, etc. Important advantages over conventional geometric design also accrue from the deformable model approach (Terzopoulos and Fleischer, 1988):

- 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 physics-based sculpting is intuitive for shape design and control.
- The equilibrium state of the dynamic model is characterized by a minimum of its potential energy, subject to imposed constraints (Terzopoulos, 1986). 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 is 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, all existing geometric operations and techniques are concurrently applicable at the geometric level.

Physics-based shape design can free designers from the need to make nonintuitive decisions such as assigning weights to NURBS or determining shape parameters in variational splines through the direct and intuitive sculpting of NURBS objects. In addition, with physics-based direct manipulation, non-expert users are able to concentrate on visual shape variation without needing to comprehend the underlying mathematical formulation. Designers are allowed to interactively sculpt shapes in a natural and predictable way using a variety of force-based tools. For example, the D-NURBS swung surfaces (see Chapter 5) in Fig. 1.1 were interactively sculpted in this fashion from the prototype shapes indicated in the caption.

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, time is fundamental to physics-based modeling. Additional advantages can be obtained through the use of real-time dynamics:

- An "instantaneous" optimizer (if such a thing existed) can produce some kinematics 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, non-smooth 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 marries the geometry with time, mass, force, and constraint. Dynamic models produce smooth, natural motions which are familiar and easily controlled.

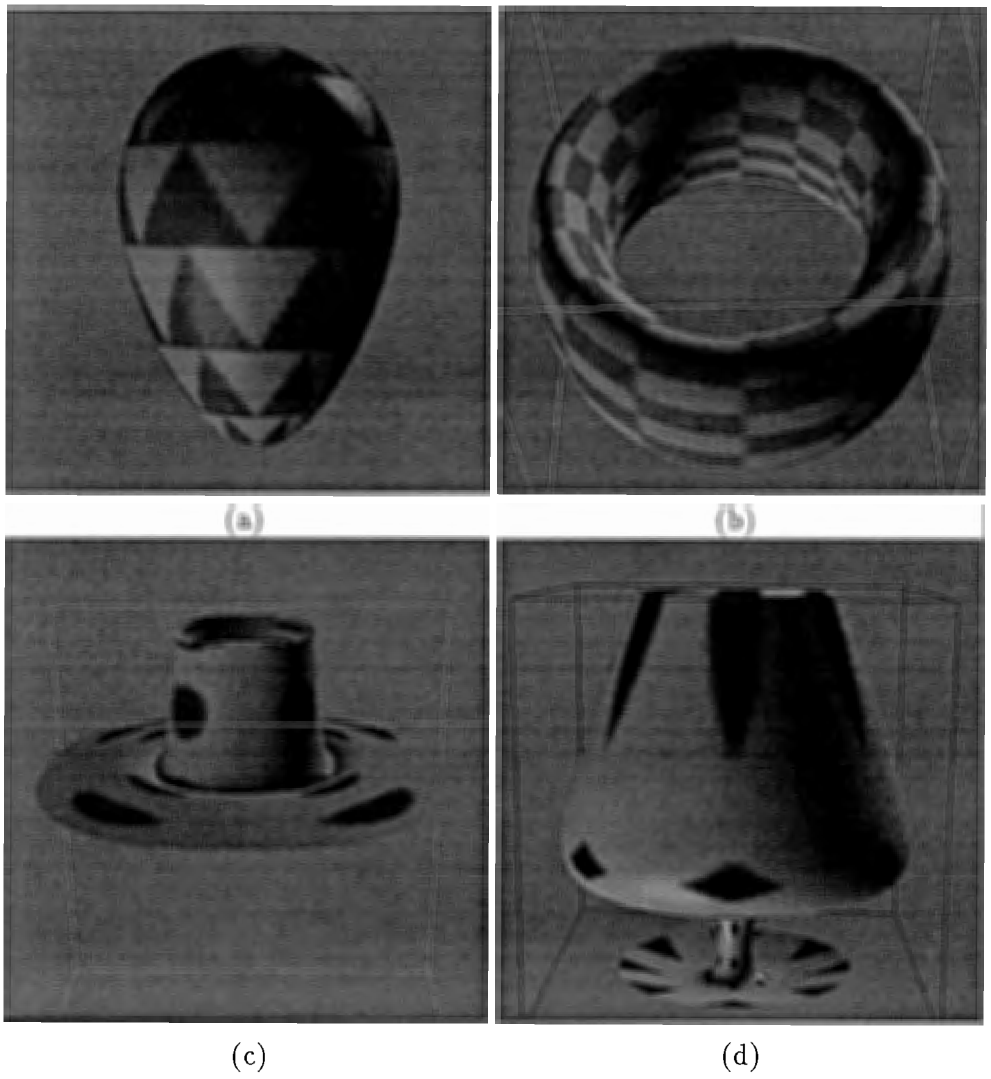


Figure 1.1: Assorted Dynamic NURBS Swung Surfaces. Open and closed surfaces shown were sculpted interactively from prototype shapes noted in parentheses (a) Egg shape (sphere). (b) Deformed toroid (torus). (c) Hat (open surface). (d) Wine glass (cylinder).

- 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 which grounds shape variation in real-world physics. Despite the fact that incremental optimization may provide a 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 applicable throughout the entire design and manufacturing process.

1.3 Contributions

Geometric representation is abstract. Geometry does not have the intuitive behavior of precomputational, physical design media such as modeling clay. This thesis paves the way for NURBS to achieve their full potential, by putting the laws of physics on their side. By associating physical behavior with abstract geometric representation, we take another important step towards bridging the gap between geometry and the requirements of design and manufacturing. Hence, this thesis develops a dynamic design framework supporting both direct manipulation and interactive sculpting. The primary contributions of the research, which have been documented in (Qin and Terzopoulos, 1993; Terzopoulos and Qin, 1994; Qin and Terzopoulos, 1994; Qin and Terzopoulos, 1995b; Qin and Terzopoulos, 1995a; Qin and Terzopoulos, 1995c), will be detailed in the following sections.

1.3.1 The D-NURBS Framework

We propose *dynamic NURBS*, or D-NURBS, a physics-based generalization of the standard NURBS representation. The geometric control points and weights of NURBS become the generalized coordinates of D-NURBS within a Lagrangian mechanics formulation. Mass distributions, internal deformation energies, forces, and other physical quantities augment the NURBS geometric substrate (Fig. 1.2).

1.3.2 Automatic Selection of DOFs

Making meaningful use of weights has proven to be one of the most important and challenging areas of current NURBS research. The dynamics framework of D-NURBS provides a systematic mechanism for automatically determining the values of both control points and weights in accordance with design requirements and direct sculpting forces. In contrast, with traditional geometric design, one often resorts to heuristic guesses to set the unknown weights, which often lacks consistency.

1.3.3 Formulation of the D-NURBS Models

We develop four related varieties of D-NURBS: (i) D-NURBS curves, (ii) tensor product D-NURBS surfaces, (iii) swung D-NURBS surfaces, and (iv) triangular D-NURBS surfaces. The equations of motion for all four varieties of D-NURBS are formulated systematically through the application of Lagrangian mechanics. The dynamic behavior of D-NURBS re-

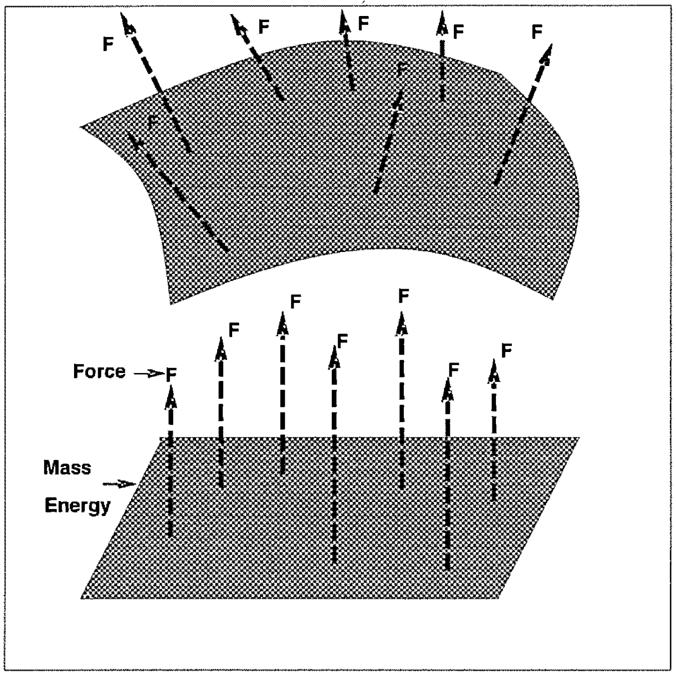


Figure 1.2: Physics-Based D-NURBS Models.

sults from the numerical integration of a set of nonlinear differential equations of motion that automatically evolve the geometric DOFs in response to the applied forces and constraints to produce physically meaningful, hence intuitively predictable motion and shape variation.

1.3.4 Geometric Constraints

We develop techniques for systematically incorporating linear and non-linear geometric constraints into the D-NURBS design framework, either as soft constraints to be satisfied approximately or as hard constraints that must never be violated. Within the Lagrangian dynamics framework, all constraints are viewed uniformly as forces.

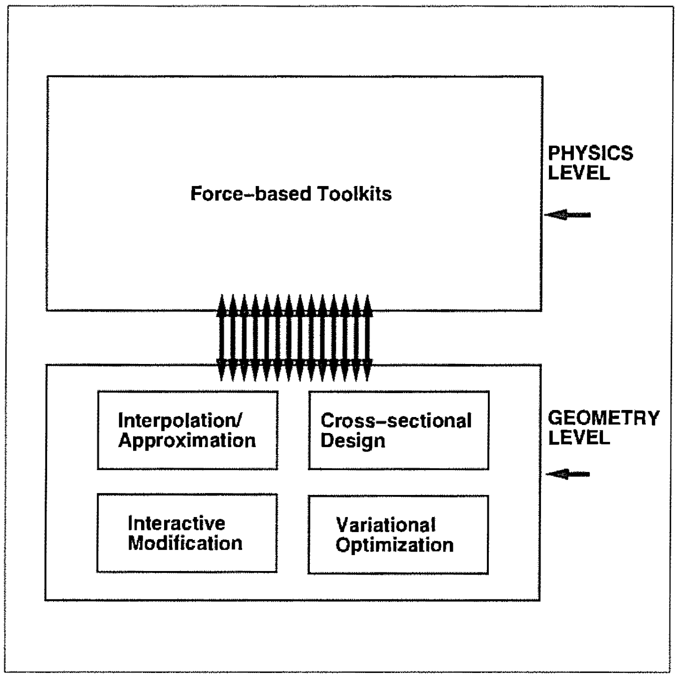


Figure 1.3: A Two-Level Physics-Based Design Paradigm.

1.3.5 Physics-Based Design Paradigm

We present a new physics-based design paradigm based on D-NURBS which generalizes traditional geometric NURBS design methodology. Within the physics-based shape design framework, elastic functionals can be used to allow the qualitative imposition of "fairness" criteria through quantitative means. Optimal shape design (nonlinear shape optimization) results when D-NURBS achieve static equilibrium subject to global or local geometric constraints. Designers can interactively sculpt complex shapes to satisfy required specifications not only in the traditional indirect fashion, by adjusting control points, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. Existing geometric toolkits continue to be applicable at the geometric level, while new force-based toolkits support dynamic manipulation and interactive sculpting at the physics level (Fig. 1.3). More importantly, the two type of toolkits are compatible with each other. Designers are free to choose either one or both to achieve design requirements.

1.3.6 D-NURBS Finite Elements and Numerical Implementation

Because the evolution of the D-NURBS generalized coordinates is determined by second-order nonlinear differential equations with time-varying mass, damping, and stiffness matrices, we cannot obtain an analytical solution in general. To implement the D-NURBS model, we develop an unusual type of finite element—the D-NURBS finite element. Using finite element analysis with this new element we reduce the D-NURBS equations of motion to efficient algorithms that can be simulated at interactive rates using standard numerical techniques.

1.3.7 Prototype Interactive Modeling Environment and Applications

We implement a prototype modeling environment based on D-NURBS. We have used this system to demonstrate that D-NURBS are effective tools in a wide range of applications, including interactive sculpting through force-based direct manipulation tools, shape blending, scattered data fitting, surface trimming, cross-sectional shape design, shape metamorphosis, and free-form deformation with geometric and physical constraints. We show that D-NURBS provide a systematic and unified approach for a variety of CAGD and graphics modeling problems.

1.4 Thesis Organization

The organization of the remainder of the thesis is as follows:

In Chapter 2, we review geometric modeling methods which have proven to be extremely useful to CAGD and graphics. Surveying a diversity of shape representations, we show that, among various frequently used representations, NURBS have become an industrial standard for free-form modeling primarily because of their unified formulation for both free-form and common analytic shapes.

In Chapter 3, we first summarize several geometric design techniques including interpolation, approximation, interactive modification and variational optimization. We discuss some of the limitations of geometric design and how variational design, the precursor of physics-based geometric design, overcomes some of these limitations. We then review prior work on physics-based modeling applied to shape design. The advantages of the physics-based modeling approach will serve to motivate D-NURBS which will be developed in Chapters 4, 5, and 6.

In Chapter 4, we formulate D-NURBS curves and tensor-product surfaces based on their geometric counterparts. The shape parameters (control points and associated weights) of geometric NURBS become the generalized (physical) coordinates of D-NURBS. We introduce time, mass, and deformation energy into the standard NURBS formulation and employ Lagrangian dynamics to arrive at a system of nonlinear ordinary differential equations that govern the shape and motion of D-NURBS.

In Chapter 5, we develop the physics-based generalization of geometric NURBS swung surfaces which are formed by swinging one planar NURBS profile curve in the x-z plane along a second NURBS trajectory curve in the x-y plane. These topologically variable surfaces retain considerable geometric coverage. They can represent a broad class of shapes with essentially as few degrees of freedom as it takes to specify the two generator curves. We formulate swung D-NURBS in terms of the degrees of freedom of two generator D-NURBS curves. This allows us to derive equations of motion for swung D-NURBS with mass, damping, and deformation energy distributions.

In Chapter 6, we propose triangular NURBS, the rational generalization of triangular B-splines, with weights as extra degrees of freedom to increase the power of the modeling scheme. Triangular NURBS can represent complex non-rectangular shapes over arbitrary triangulated planar domains with low degree piecewise polynomials that nonetheless maintain relatively high-order continuity. Furthermore, we develop triangular D-NURBS, the physics-based generalization of triangular NURBS. We introduce time, mass, deformation energy into triangular D-NURBS and employ Lagrangian dynamics to derive their equations of motion.

In Chapter 7, we incorporate geometric constraints into the D-NURBS framework. Linear constraints allow us to reduce the generalized coordinates vector and matrices to a minimal unconstrained set of generalized coordinates. We also formulate nonlinear constraints on D-NURBS through Lagrange multiplier techniques. Furthermore, we show that swung D-NURBS may also be derived as tensor-product D-NURBS that have been subjected to a dimensionality-reducing nonlinear constraint.

In Chapter 8, we propose and formulate the D-NURBS finite element. Each D-NURBS patch (span for the D-NURBS curve) is treated as a finite element. We compute the individual D-NURBS element matrices using Gaussian quadrature. An efficient numerical implementation of D-NURBS is obtained through the use of finite element analysis. The discrete D-NURBS equations of motion are integrated through time using either explicit or implicit Euler methods.

In Chapter 9, we describe a prototype modeling environment and present our physics-based design paradigm based on D-NURBS. We illustrate with examples that when working within the D-NURBS framework, design requirements may be satisfied through the use of energies, forces, and constraints. We demonstrate a wide range of applications including shape blending, scattered data fitting, surface trimming, cross-sectional shape design, shape metamorphosis, and free-form deformation with geometric and physical constraints.

Chapter 10 summarizes the thesis and suggests future research directions.

Chapter 2

Survey of Geometric Modeling

Geometric modeling has been extensively explored during the past three decades. It is crucial to a variety of fields from computer aided geometric design to computer graphics, virtual reality, medical imaging, and computer vision. Numerous geometric representations have been developed for free-form parametric modeling, ranging from polynomials to NURBS. The latter have become a standard representation in commercial modeling systems. In this chapter, we review the range of geometric representations and their properties.

2.1 Parametric Representation

Geometric computation requires that shapes be represented in a precise and unambiguous manner and that the mathematical formulation be flexible for free-form design. Numerous geometric formulations have been proposed for a large variety of geometric modeling applications. Geometric objects can be represented with simple parametric polynomials,

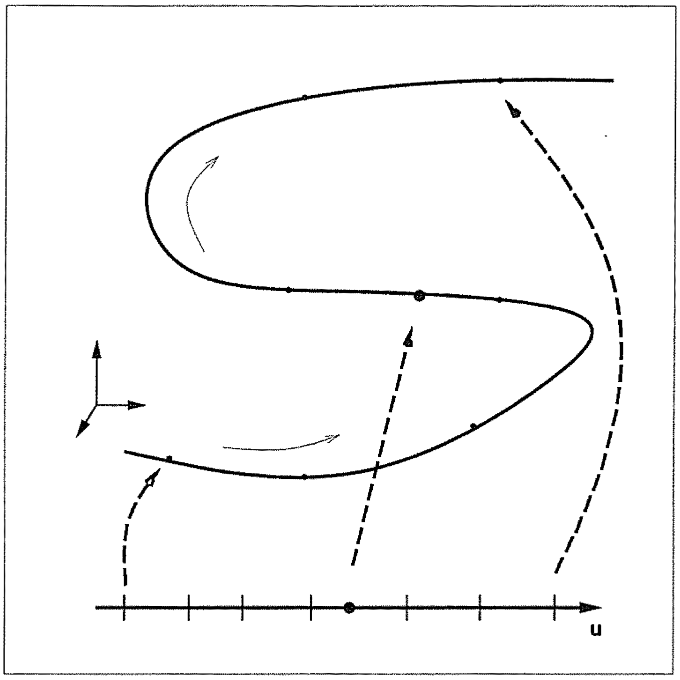


Figure 2.1: The three-dimensional parametric curve and its parametric domain.

piecewise non-rational or rational splines, domain-less recursive subdivision surfaces, and implicit functions. Free-form parametric splines have been frequently used in a large variety of modeling systems primarily because of their representational power and flexibility. In this section, we discuss frequently used parametric curves and surfaces such as B-splines and transfinite surfaces. Other geometric formulations including subdivision forms, algebraic functions, and variational splines can be found in Appendix A.

2.1.1 Parametric Curves

A parametric curve is defined as a function of one parameter u over an interval domain I: $\mathbf{c} = \mathbf{c}(u)$ (see Fig. 2.1). The parametric formulation supports straightforward geometric computation. It can be easily extended to higher dimensions.

Polynomial Forms

From the computational viewpoint, finite-degree polynomials are ideal for representing and approximating geometric entities because polynomials are closed under differentiation, integration, and arithmetic operations. The best known polynomial representation is the monomial form of degree n:

$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \ldots + \mathbf{a}_n u^n. \tag{2.1}$$

Despite its simplicity, the computational advantage of (2.1) is counteracted by the fact that the $\mathbf{a}_i$ in (2.1) do not provide any intuitive insight into the curve shape beyond the point $u = 0$.

In addition, a polynomial curve can be defined by a set of $n+1$ coefficients along with a knot sequence $t_0, \ldots, t_n$

$$\mathbf{f}(u) = \mathbf{a}_0 L_0^n(u) + \ldots + \mathbf{a}_n L_n^n(u) \tag{2.2}$$

where $L_i^n(u)$ are Lagrange polynomials of degree n satisfying $L_i^n(u_j) = \delta_{ij}$, and δ is the Kronecker delta. It is apparent that the polynomial curve interpolates all the $\mathbf{a}_i$. However, the interpolating curve exhibits unwanted oscillation, especially for high-order polynomials. Therefore, both the monomial and the Lagrangian forms are not suitable for the construction of piecewise smooth curves.

Alternatively, a modeler can use Hermite polynomials $H_i^n(u)$

$$\begin{aligned}\mathbf{f}(u) = {} & \mathbf{D}_0\mathbf{a}_0 H_0^n(u) + \mathbf{D}_1\mathbf{a}_0 H_1^n(u) + \ldots + \mathbf{D}_{(n-1)/2}\mathbf{a}_0 H_{(n-1)/2}^n(u) \\ & + \mathbf{D}_{(n-1)/2}\mathbf{a}_1 H_{(n+1)/2}^n(u) + \ldots + \mathbf{D}_1\mathbf{a}_1 H_{(n-1)}^n(u) + \mathbf{D}_0\mathbf{a}_1 H_n^n(u)\end{aligned} \tag{2.3}$$

where $\mathbf{D}_j\mathbf{a}_i = \mathbf{f}^{(j)}(i)$, $i = 0, 1$, $j = 0, \ldots, (n-1)/2$, and n is odd. Even-degree Hermite polynomials require a different treatment (see (Farouki and Hinds, 1985) for the details). Although (2.3) is geometrically intuitive, the Hermite interpolant has a severe drawback—designers must be provided with high-order derivative information in order to achieve the desired order of continuity across the adjacent curve spans. Unfortunately, derivative information is neither available nor easily derived in most applications.

To overcome prior difficulties, de Casteljau and Bezier independently developed a curve formulation, now called Bezier curves, based on Bernstein polynomials $B_i^n(u)$:

$$\mathbf{f}(u) = \mathbf{a}_0 B_0^n(u) + \ldots + \mathbf{a}_n B_n^n(u) \tag{2.4}$$

where

$$B_i^n(u) = \begin{pmatrix} n \\ i \end{pmatrix} (1-u)^{n-i} u^i, i = 0, 1, \ldots, n.$$

Bezier points $\mathbf{a}_0, \ldots, \mathbf{a}_n$ form the Bezier polygon. Unlike the coefficients in (2.1)–(2.3), Bezier control points are viewed as design tools to generate a free-form curve. The control polygon is both simple and easy to use mainly because Bernstein basis functions have many important features such as: (i) partition of unity, (ii) positivity, and (iii) recursive evaluation. Feature (i) ensures Bezier points are invariant under affine transformation. Features (i) and (ii) guarantee that the curve segment resides within the convex hull or Bezier polygon and that the Bezier curve has a variation diminishing property. Feature (iii) serves as a basis for the well known de Casteljau algorithm which is used to evaluate a curve through repeated linear interpolation. This recursive evaluation has proven to be both computationally efficient and stable (note that it is slower than Horner's rule). Other important properties of Bernstein polynomials include degree elevation and subdivision (see also (Bohm, Farin and Kahmann, 1984) for the details about the differentiation and integration of Bernstein polynomials and Bezier curves).

Parametric Splines

The above modeling techniques are relatively simple and robust. The only available means for modeling complex shapes, however, is to increase the degree of the basis functions. Higher order polynomials afford greater flexibility at the expense of more complex geometric computation, but they are not quite satisfactory because they tend to introduce spurious undulations between interpolating points. Such oscillations are not implied by the data points themselves. Moreover, designers have to enforce extra continuity requirements where two adjacent polynomials meet. This, in general, complicates the design task tremendously. One alternative is to use piecewise polynomials, or splines.

The mathematical theory of splines was originated by Schoenberg in 1946 (Schoenberg, 1946). A spline curve is a piecewise univariate function satisfying a set of continuity constraints. The spline can be categorized in terms of interpolation/approximation schemes or global/local schemes. In 1972 de Boor proposed B-splines from the standpoint of approximation theory (de Boor, 1972). During 1973 and 1974, Riesenfeld and Gordon proposed B-splines as a powerful and proper generalization of Bernstein polynomials for free-form curve design (Riesenfeld, 1973; Gordon and Riesenfeld, 1974).

A B-spline curve is the combination of a set of piecewise polynomial functions with $n+1$ control points $\mathbf{p}_i$

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(u) \tag{2.5}$$

where u is the parametric variable and $B_{i,k}(u)$ are B-spline basis functions. Assuming basis functions of degree $k-1$, a B-spline curve has $n+k+1$ knots t_i in a non-decreasing sequence:

$t_0 \le t_1 \le \ldots \le t_{n+k}$. B-spline basis functions are defined recursively as

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

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 $t_{k-1} \le u \le t_{n+1}$. Similar to Bezier points in (2.4), de Boor points $\mathbf{p}_i$ form the control polygon. Many attractive and important properties follow immediately including the following:

- Partition of unity, positivity, and recursive evaluation for B-spline basis functions.
- B-splines intrinsically provide various smoothness criteria. The curve is C^{k-j} continuous at a j-fold knot.
- B-splines include Bezier curves as their subset. Composite Bezier curves equivalent to a B-spline curve can be easily obtained by means of multiple knot insertion at every old knot until all knots are of multiplicity k.
- In contrast to Bezier curves, B-splines have a local control property—the adjustment of one control point affects the curve only locally.
- Invariance under linear transformation.
- Strong convex hull and variation diminishing properties.

A recursive evaluation algorithm that can efficiently evaluate B-spline curves was invented by Mansfield, de Boor, and Cox (de Boor, 1972; Cox, 1972). B-spline derivative and integration formulas are discussed in detail in (Bohm, Farin and Kahmann, 1984). One important advantage of B-splines over Bezier polynomials is that, in order to increase the

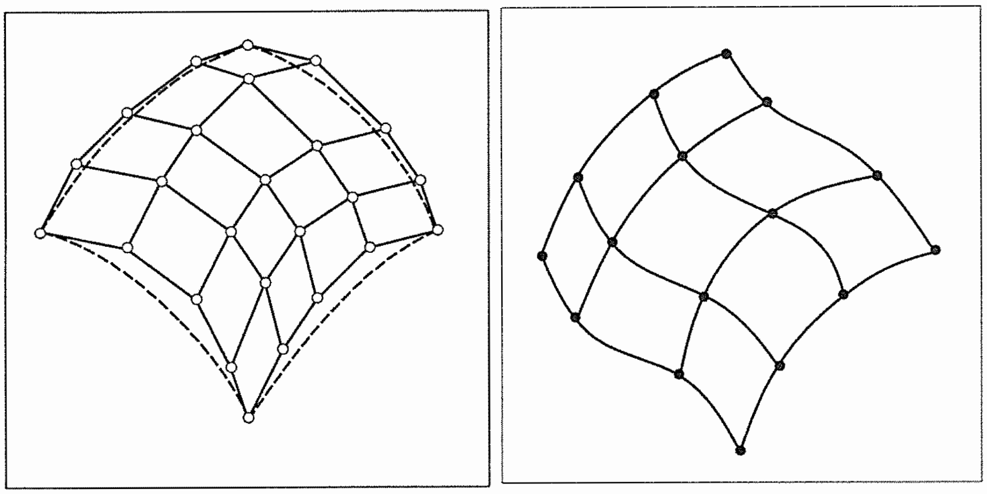

(a) (b)

Figure 2.2: Control point based and transfinite representations.

potential flexibility of B-splines, a modeler need not raise the degree of the basis functions. Instead, this goal can easily be achieved through knot insertion with which the number of B-spline DOFs are increased while the order of the basis function remains unchanged.

2.1.2 Parametric Surfaces

Since the pioneering work by Coons and Bezier in the late 50's, CAGD has been dominated by the theory of rectangular surface patches. Two predominant modeling schemes are control point based (Fig. 2.2(a)) and transfinite based (Fig. 2.2(b)) representations. Recently, many different approaches have been derived for representing smooth surfaces. Today, parametric surfaces can be categorized as follows:

Directrix-Generator The surface is swept out by sliding a generator curve across a set of directrices. Three essential constituents include directrices (longitudinal curves), correspondence rules, and generators. To make the generated surface satisfy continuity requirements, the constituents must be sufficiently smooth. Typical examples are conic lofting surfaces and surfaces of revolution.

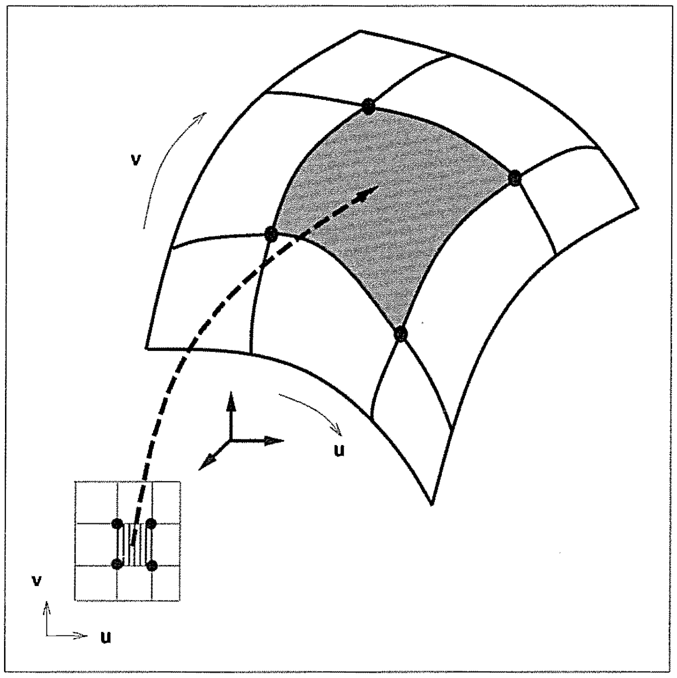


Figure 2.3: Rectangular surface and its parametric domain.

Transfinite Patch The blending surface interpolates a network of given curves (e.g., the Coons patch).

Multiple Patch The surface is a collection of "smaller" patches (e.g., the Bezier patch, and n-sided patches).

Carpet Method The surface is a set of multiple patches, and adjacent patches are constrained to maintain certain continuity conditions (e.g., B-splines).

New Representations Typical examples are recursive subdivision surfaces, irregular B-spline-like surfaces, multivariate B-splines, and algebraic patches.

Tensor-Product Surfaces

Among various surface forms, the tensor-product surface is the simplest and most popular scheme because it is a straightforward generalization of its curve predecessor. Given two sets of basis functions $\{F_i(u)\}$ and $\{G_j(v)\}$ for curves, a general tensor-product surface can be constructed as a linear combination of the composite set of basis functions $\{F_i(u)G_j(v)\}$ as follows:

$$\mathbf{f}(u,v) = \sum_i \sum_j \mathbf{c}_{i,j} F_i(u) G_j(v). \tag{2.6}$$

Fig. 2.3 illustrates a tensor-product surface and its parametric domain. Because of the special construction of (2.6), the properties of tensor-product surfaces are usually determined by those of the underlying curve schemes. Tensor-product surfaces of monomial and Lagrangian forms can easily be derived in light of (2.1) and (2.2), respectively. Analogous to (2.4), the tensor-product Bezier surface can be defined as

$$\mathbf{f}(u,v) = \sum_i \sum_j \mathbf{b}_{i,j} B_i^m(u) B_j^n(v). \tag{2.7}$$

Like Bezier curves, the many nice properties of Bezier surfaces include the following:

- All isoparametric curves are Bezier curves.
- Partial derivatives along the boundary can be formulated explicitly.
- The strong convex hull property.
- Degree elevation and subdivision are carried out along all rows or all columns.
- Affine invariance.

Bezier surface evaluation can be implemented by successively applying the de Casteljau algorithm on relevant rows and columns (note that the evaluation order is irrelevant). In analogy to Bezier curves, complex surfaces can be constructed by stitching together a number of Bezier patches smoothly.

Likewise, a tensor-product B-spline surface is defined over the parametric variables u and v as

$$s(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{i,j} w_{i,j} B_{i,k}(u) B_{j,l}(v). \tag{2.8}$$

A B-spline surface has $(m+1)(n+1)$ control points $\mathbf{p}_{i,j}$. 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 non-decreasing 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.

B-spline surfaces generalize Bezier patches. Their primary properties include the following:

- All isoparametric curves are B-spline curves.
- Local control.
- Knot insertion increases the DOFs while keeping the degree of basis functions unchanged.
- Strong convex hull.
- B-splines include Bezier surfaces as their subset. The piecewise Bezier representation of B-splines can be obtained through multiple knot insertion.
- The B-spline control polygon converges to the B-spline surface as the number of knots increases.
- B-spline control points permit interactive manipulation.

Transfinite Surfaces

Unlike the previous control point based schemes, the transfinite approach allows surfaces to be determined by a set of prescribed boundary curves. Before the advent of computers, transfinite-based surfaces were generated with the lofting technique. One of the most popular transfinite surfaces is the Coons patch which can be obtained by blending four boundary curves either bilinearly or bicubically. The bilinearly blended Coons patch can be concisely formulated with the Boolean sum

$$(P)\mathbf{f} = (P_1 \oplus P_2)\mathbf{f} = (P_1 + P_2 - P_1P_2)\mathbf{f} \tag{2.9}$$

where two operators P_1, and P_2 are defined in terms of Lagrange polynomials in (2.2) and four boundary curves

$$(P_1)\mathbf{f} = \mathbf{f}(0,v)L_0^1(u) + \mathbf{f}(1,v)L_1^1(u)$$

$$(P_2)\mathbf{f} = \mathbf{f}(u,0)L_0^1(v) + \mathbf{f}(u,1)L_1^1(v)$$

Fig. 2.4 illustrates a Boolean sum construction of a bilinearly blended Coons patch. It is apparent that a bilinearly blended Coons patch interpolates four boundary curves. When abutting Coons patches, however, bilinearly blended Coons patches are only C^0 across their boundaries. If cross-boundary derivatives are known, bicubically blended Coons patches can be defined using cubic Hermite polynomials (see (Bohm, Farin and Kahmann, 1984) for the details). Likewise, bicubically blended Coons patches interpolate both boundary curves and prescribed boundary derivatives.

Theoretically, transfinite surfaces may produce high-order continuity across adjacent patches. In practice, however, high-order derivatives are often unavailable during the design process. The *ad hoc* and heuristic approach is to estimate the initial derivative values based on generating curves in the vicinity. Another design difficulty comes from the inconsistency

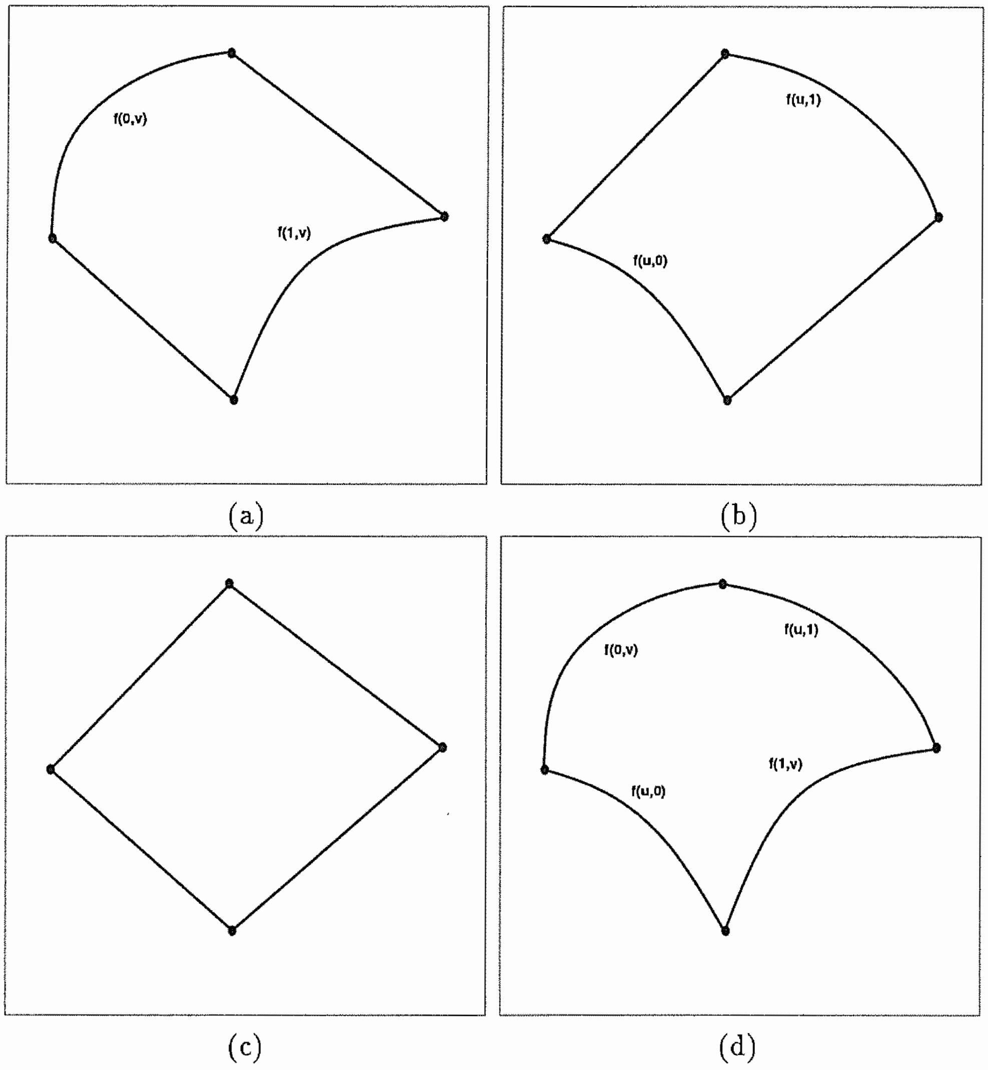


Figure 2.4: The boolean-sum construction of a bilinearly blended Coons patch.

of the cross-derivative (twist) which may influence local surface oscillations. To overcome twist incompatibility in Coons patches, Gregory replaced the incompatible constant twists with variable twists using a convex combination method, which is also known as *Gregory's square.*

Coons' contribution was very significant primarily because it stimulated the development of other surface representations. Generalizing the Coons patch, Gordon constructed a blending surface from a quadrangular network of compatible curves with Lagrange polynomials. The idea of blending was further carried over to transfinite interpolation of triangles (Barnhill, Birhoff and Gordon, 1973).

Two major schemes have been developed for free-form curve/surface design. The Bezier/B-spline technique emphasizes geometric intuition, while the Coons/Gordon method is based on abstract algebraic concepts such as the Boolean sum. The two types of shape representations differ from each other dramatically in terms of their geometric nature. One apparent distinction is that the boundary curves of a B-spline surface must also be B-splines, whereas the Coons patch allows its boundary curves to be of arbitrary form.

Irregular Patches

Despite their simplicity, tensor-product surfaces have drawbacks. The underlying shape (and the parametric domain) of tensor-product surfaces must be "topologically" rectangular. However, when used for representing complex shapes, general n-sided patches are more desirable than rectangular patches. Fig. 2.5 shows a surface composed of triangular patches and its parametric domain.

Historically, de Casteljau considered triangular patches in the late 50's even before he defined tensor-product "Bezier" patches. Nevertheless, it was not until the 70's that

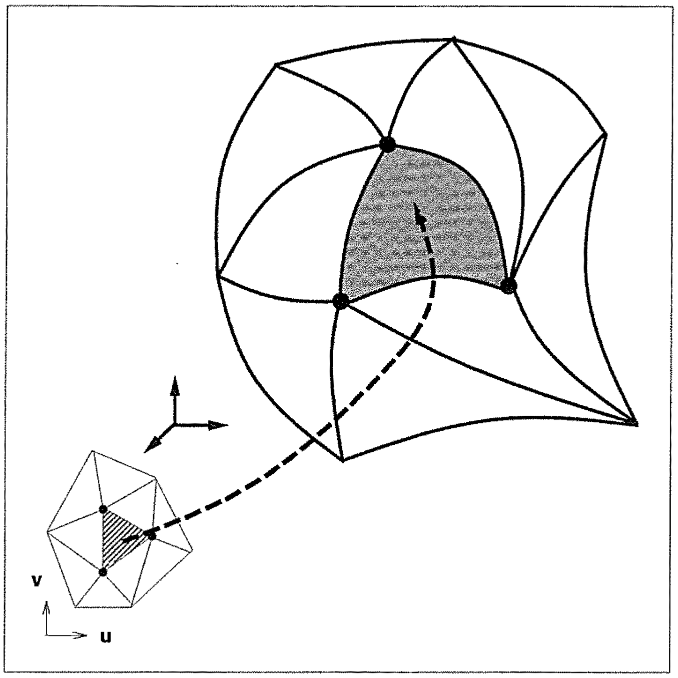


Figure 2.5: An irregular surface and its parametric domain.

triangular schemes started to be widely applied to CAGD. Farin constructed Bezier triangles based on the Bernstein form (see Fig. 2.6 for an example of cubic Bezier triangles). The key component of triangular Bezier patches (Farin, 1986) is the barycentric representation of the triangulated domain (see Fig. 2.7). Barycentric coordinates are affine invariant. Like the univariate Bezier curve and tensor-product Bezier surface, the Bezier triangle has many nice properties such as partition of unity, positivity, and recursive evaluation for basis functions. Detailed discussion about the formulation, derivative computation, degree elevation, subdivision, and continuity condition can be found in (Bohm, Farin and Kahmann, 1984; Farin, 1986). Triangular Bezier patches have been used in various design applications. They can be used as Hermite interpolants which interpolate position and derivative information at vertices and across boundaries. Subsequently, other triangular interpolants have been derived (Barnhill, 1985). The main techniques of triangular patches can be characterized as transfinite, convex combination, and control point approaches.

Significant effort has also been devoted to the derivation of arbitrary n-sided patches.

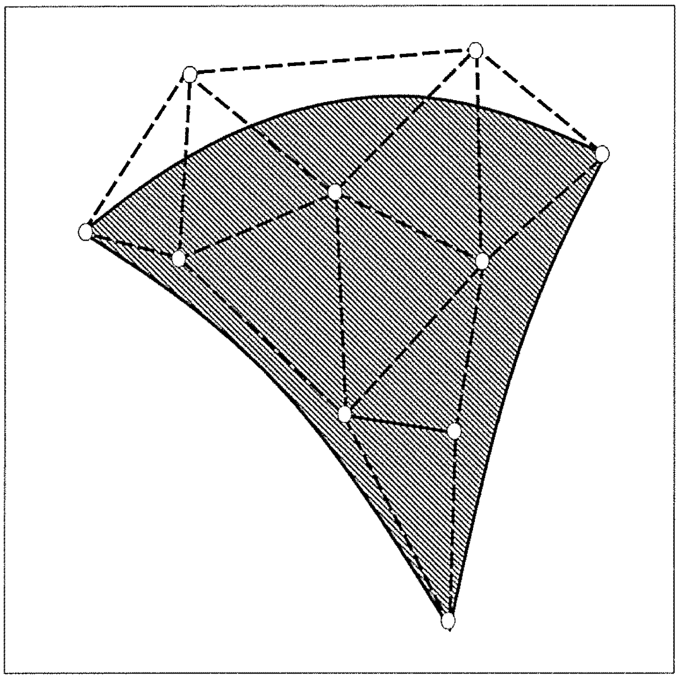


Figure 2.6: The construction of a cubic triangular Bezier patch.

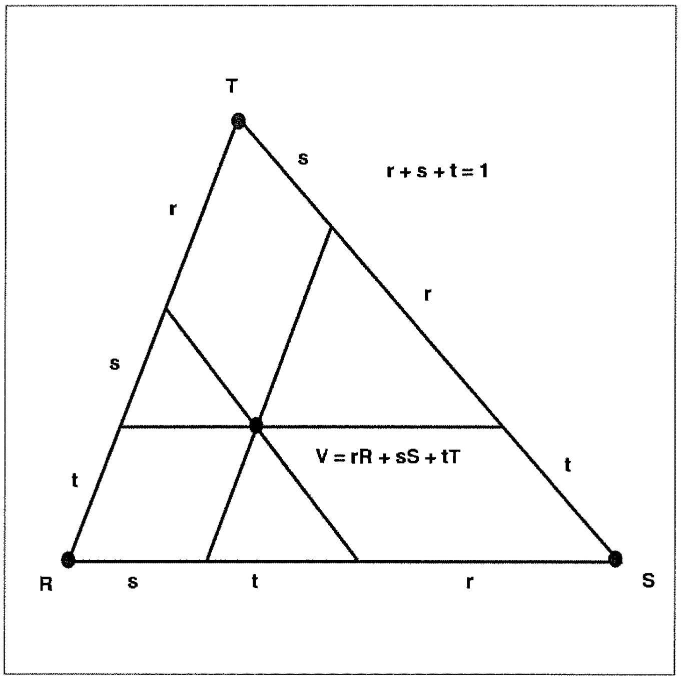


Figure 2.7: Barycentric coordinates of a triangle.

Generalizing triangular and tensor-product Bezier surfaces, Loop and DeRose defined S-patches over an arbitrary n-sided convex polygonal domain (Loop and DeRose, 1989). The key technique is functional composition which embeds a polygonal parametric domain into a higher dimensional simplex. The S-patch is obtained by restricting the Bezier simplex to the embedded polygonal domain. Despite its generalization over regular Bezier surfaces, the S-patch suffers from major drawbacks such as a complicated domain mapping formulation, a time-consuming evaluation algorithm, and the lack of a geometric interpretation for its control polygon.

2.2 NURBS Geometry

In this section, we review the formulation of Non-Uniform Rational B-Spline (NURBS) curves and surfaces. We then describe their analytic and geometric properties. Intuitively, a spatial NURBS (curve or surface) is defined as the projection of a 4D B-spline into a 3D homogeneous space. Fig. 2.8 illustrates a two-dimensional NURBS curve obtained through the projection of a three-dimensional B-spline curve.

2.2.1 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:

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} \mathbf{p}_i w_i B_{i,k}(u)}{\sum_{i=0}^{n} w_i B_{i,k}(u)}, \tag{2.10}$$

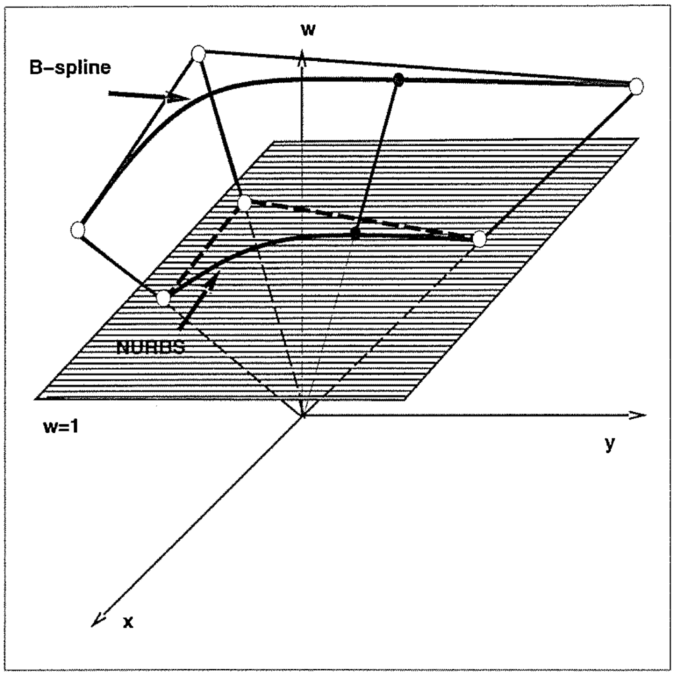


Figure 2.8: A planar NURBS curve obtained through projection.

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 non-decreasing order, $t_0 \leq t_1 \leq \ldots \leq t_{n+k}$. 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 $\mathbf{p}_n$.

2.2.2 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

$$\mathbf{s}(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{i,j} w_{i,j} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} B_{i,k}(u) B_{j,l}(v)}. \tag{2.11}$$

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 non-decreasing 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.

To evaluate NURBS, the de Boor algorithm can be applied to the numerator and denominator, respectively. A more stable and robust evaluation algorithm which does not make explicit use of 4D projective geometry is described in (Farin, 1989). The non-uniform knot vector of NURBS offers much better parameterization and greater flexibility in shape design than the previously presented formulations, especially for the fitting of unequally spaced points.

2.2.3 Properties

NURBS generalize the non-rational parametric form. In analogy to non-rational B-splines, the rational basis functions of NURBS sum to unity. NURBS inherit many properties of non-rational B-splines such as the strong convex hull property, the variation diminishing property, local support, and invariance under standard geometric transformations (see (Farin, 1990) for more details). Furthermore, they have many additional properties:

- NURBS offer a unified mathematical framework for common analytic shapes and parametric polynomial forms. For instance, NURBS can be used to precisely express conic segments as well as full conics. NURBS can also represent extruded surfaces, natural quadrics (plane, cylinder, cone and sphere), ruled surfaces, and surfaces of revolution.

- NURBS are infinitely smooth in the interior of a knot span, provided the denominator is not zero, and at a knot they are at least C^{k-1-r} continuous with knot multiplicity r. This enables designers to employ NURBS satisfying different smoothness requirements.

- NURBS include weights as extra degrees of freedom which can influence local shape. Weights have a clear geometric interpretation. If a particular weight is zero, the corresponding rational basis function is also zero. Thus, the corresponding control point does not affect the NURBS shape. The spline is attracted toward a control point if the corresponding weight is increased, and repelled from the control point if the weight is decreased (see Fig. 2.9). By manipulating control points and weights, NURBS provide the greatest flexibility for designing a large variety of shapes.

- Associated with NURBS are a set of powerful geometric toolkits such as knot insertion, refinement, knot removal, and degree elevation. Their evaluation algorithm is fast and computationally stable.

- NURBS are invariant under scaling, rotation, translation, and shear as well as parallel and perspective projections. A linear or rational linear transformation of parameters will not change the shape and order of NURBS. They are the genuine generalization of non-rational B-splines as well as rational and non-rational Bezier forms.

There are various ways to represent a circle using a NURBS curve (Piegl and Tiller, 1989) (Fig. 2.10 shows one NURBS representation). NURBS can be used to precisely construct conics and rational quadric patches of various types (Piegl, 1985). Spherical patches such as a hemisphere, an octant of the sphere, and a whole sphere can be represented using NURBS (Piegl, 1986).

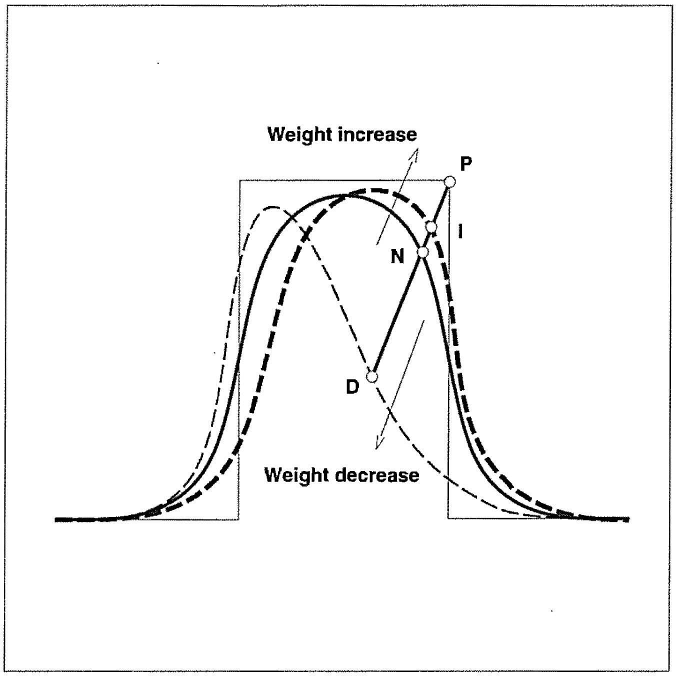


Figure 2.9: Weight influences on the shape of NURBS curve.

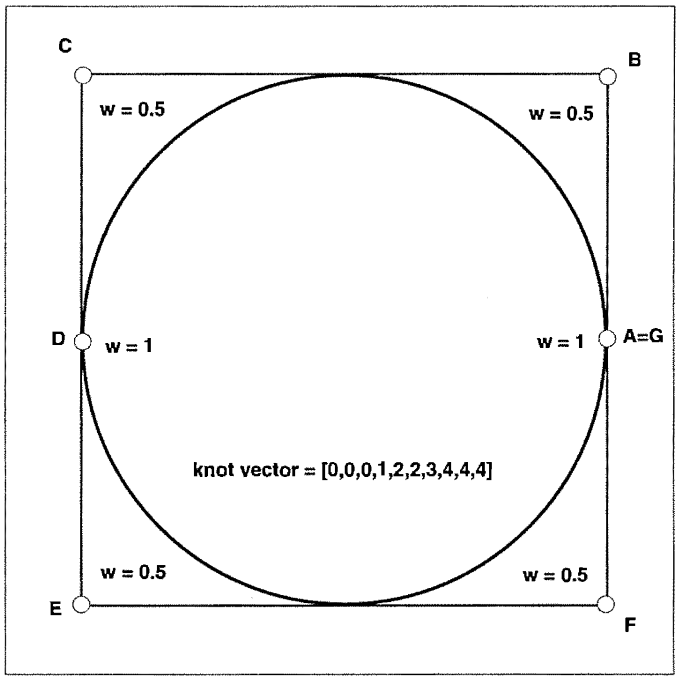


Figure 2.10: A circle is represented using quadratic NURBS curve with 7 control points.

Based on the concept of infinity from projective geometry, Piegl incorporated infinite control points into the NURBS formulation (Piegl, 1987b; Piegl, 1987a). This can forge a link between the B-spline and Hermite representations. Infinite control points simplify the data set in representing complicated shapes. They can be used to define circles, surfaces of revolution, and general torus patches. Infinite control points can also be used as a design tool which facilitates designers having no expertise of projective geometry.

Piegl systematically discussed NURBS modification through knot insertion and control point/weight based manipulation (Piegl, 1989a; Piegl, 1989b). Other NURBS geometric design techniques include the interpolation/approximation of a set of data points and cross-sectional design. If weights are provided *a priori*, interpolation and approximation with NURBS is reduced to the solution of a set of linear equations. If weights are not given, however, one often resorts to guessing weights and other heuristics. To fit a large amount of data, especially when the data set is subject to measurement error, least-square approximation is preferable to strict interpolation.

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.

To date, NURBS are the most general parametric representation because they are capable of representing both analytic shapes and free-form parametric forms. They combine a low-degree rational representation of maximal smoothness with a geometrically intuitive variation. The rapid proliferation of NURBS is due partly to their excellent properties and partly to their incorporation into such national and international standards as IGES and

PHIGS+. NURBS have been incorporated into a large number of commercial modeling systems.

2.3 Multivariate Simplex Splines

In this section, we review the history of multivariate simplex splines and present their formulation.

2.3.1 Background

The theoretical foundation of triangular B-splines lies in the multivariate simplex spline of approximation theory. Motivated by an idea of Curry and Schoenberg for a geometric interpretation of univariate B-splines, de Boor first presented a brief description of multivariate simplex splines (de Boor, 1976). Since then, their theory has been explored extensively (Micchelli, 1979; Dahmen and Micchelli, 1982; Dahmen and Micchelli, 1983; Hollig, 1982). The well-known recurrence relation of multivariate simplex splines was first proposed in (Micchelli, 1979). Subsequently, Grandine gave a stable evaluation algorithm (Grandine, 1988). Dahmen and Micchelli presented a thorough review of multivariate B-splines (Dahmen and Micchelli, 1983). From the point of view of blossoming, Dahmen, Micchelli and Seidel proposed triangular B-splines which are essentially normalized simplex splines (Dahmen, Micchelli and Seidel, 1992).

The practical application of multivariate simplex splines is relatively underdeveloped compared to their theory, because of complicated domain partitioning and time-consuming algorithms for evaluation and derivative computation, especially for high dimensional and high order cases. However, it is possible to derive efficient algorithms for a low dimensional domain such as a plane and/or a low order polynomial such as a quadratic or a cubic. Traas discussed the applicability of bivariate quadratic simplex splines as finite elements and derived differentiation and inner product formulas (Traas, 1990). Auerbach et al. use the bivariate simplex B-spline to fit geological surfaces through scattered data by adjusting the triangulation of the parametric domain in accordance with the data distribution (Auerbach et al., 1991). Recently, the first experimental CAGD software based on the triangular B-spline was developed, demonstrating the practical feasibility of multivariate B-spline algorithms (Fong and Seidel, 1993).

2.3.2 Formulation

The basis functions of multivariate simplex splines can be defined either analytically or recursively. An s-variate simplex spline $M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})$ can be defined as a function of $\mathbf{x} \in R^s$ over the convex hull of a point set $\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}$, depending on the $m+1$ knots $\mathbf{x}_i \in R^s$, $i = 0,\ldots,m$, $(m \geq s)$. It is a piecewise polynomial with degree $d = m - s$ satisfying

$$\int_{R^s} f(\mathbf{x}) M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})\, d\mathbf{x} = (m-s)! \int_{S^m} f(t_0\mathbf{x}_0 + \ldots + t_m\mathbf{x}_m)\, dt_1 \ldots dt_m, \quad (2.12)$$

where f is an arbitrary integrable function defined over the region which covers the convex hull spanned by the knot sequences $\mathbf{x}_i$. The region S^m is the standard m-simplex:

$$S^m = \{(t_1,\ldots,t_m)| \sum_{i=0}^{m} t_i = 1, 1 \geq t_i \geq 0\}.$$

$M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})$ has a very simple geometric interpretation. It is the projection of a higher dimensional simplex on a lower dimensional space. Let Δ be a m-simplex extended by $m+1$ vertices $[\mathbf{v}_0,\ldots,\mathbf{v}_m], \mathbf{v}_i \in R^m$ such that the projection of each $\mathbf{v}_i$ on subspace R^s

is $\mathbf{x}_i$, and for arbitrary $\mathbf{x}$ define a point set

$$A_{\mathbf{x}} = \{\mathbf{v} | \mathbf{v} \in \Delta, \mathbf{v}|_{R^s} = \mathbf{x}\}.$$

Thus, we can explicitly formulate the basis functions of s-variate simplex splines as

$$M(\mathbf{x}|\{\mathbf{x}_0, \ldots, \mathbf{x}_m\}) = \frac{(m-s)!}{m!} \cdot \frac{\mathrm{vol}_{m-s}(A_{\mathbf{x}})}{\mathrm{vol}_m(\Delta)}, \tag{2.13}$$

where $\mathrm{vol}_k(\mathrm{s})$ denotes the k-dimensional volume of a set s.

The basis function of multivariate simplex splines may also be formulated recursively, which facilitates evaluation and derivative computation: When $m = s$,

$$M(\mathbf{x}|\{\mathbf{x}_0, \ldots, \mathbf{x}_m\}) = \begin{cases} \frac{1}{m!\,\mathrm{vol}_s([\mathbf{x}_0, \ldots, \mathbf{x}_m])}, & x \in [\mathbf{x}_0, \ldots, \mathbf{x}_m] \\ 0 & \text{otherwise} \end{cases}$$

and when $m > s$,

$$M(\mathbf{x}|\{\mathbf{x}_0, \ldots, \mathbf{x}_m\}) = \sum_{i=0}^{m} \lambda_i M(\mathbf{x}|\{\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_m\}), \tag{2.14}$$

where

$$\sum_{i=0}^{m} \lambda_i = 1; \qquad \sum_{i=0}^{m} \lambda_i \mathbf{x}_i = \mathbf{x}.$$

Note that when $m = s$ the basis function is discontinuous along the boundary of the convex hull $[\mathbf{x}_0, \ldots, \mathbf{x}_m]$. Thus, the function value is not unique, and extra effort is needed to deal with the boundary evaluation (see (Fong and Seidel, 1993) for the concept of a semi-open convex hull in the context of bivariate B-splines). Second, when $m > s$ the barycentric coefficients are not unique. An efficient method frequently used in applications is to make at least $m - s$ of the λ_i vanish, while the remaining ones are taken as positive barycentric coordinates to obtain stable and fast evaluation. Similarly, the directional derivative $D_{\mathbf{w}}$

of multivariate simplex splines may be recursively formulated as

$$D_{\mathbf{w}}M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}) = \mathbf{w}^{\mathsf{T}}\nabla M = (m-s)\sum_{i=0}^{m}\mu_i M(x|\{\mathbf{x}_0,\ldots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\ldots,\mathbf{x}_m\}), \tag{2.15}$$

where μ_i are coefficients satisfying

$$\sum_{i=0}^{m}\mu_i = 0; \qquad \sum_{i=0}^{m}\mu_i\mathbf{x}_i = \mathbf{w}.$$

Again, these scalar coefficients are not unique. An efficient algorithm to evaluate derivatives is obtained by setting $\mu_i = D_{\mathbf{w}}\lambda_i$, $i = 0,\ldots,m$, where λ_i is defined in (2.14).

2.4 Triangular B-Splines

In this section we review the formulation of triangular B-splines (see (Dahmen, Micchelli and Seidel, 1992; Seidel, 1993; Fong and Seidel, 1993; Greiner and Seidel, 1994) for the details) and summarize their analytic and geometric properties.

2.4.1 Geometry

The triangular B-spline is essentially a normalized bivariate simplex spline. 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 then assign a knot sequence (also called a cloud of knots) $[\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n]$ (which are inside the shaded circles in

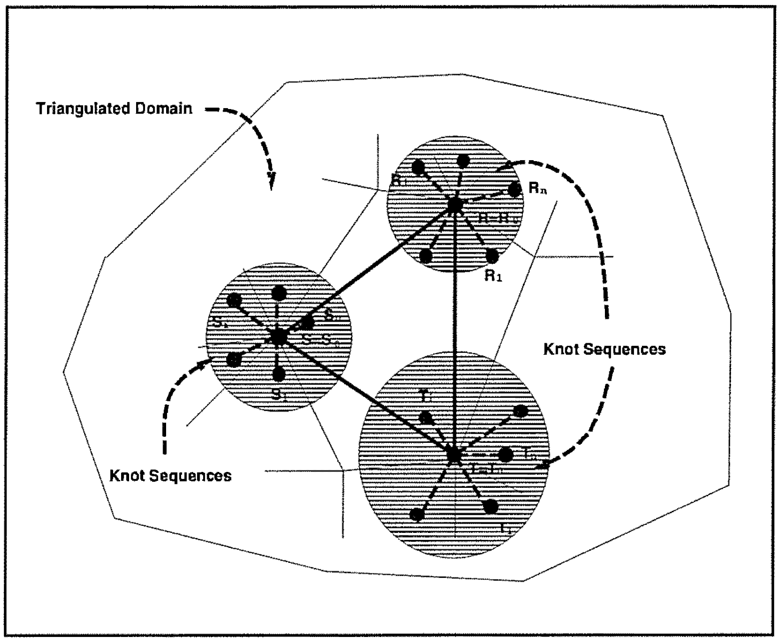


Figure 2.11: Knot vectors associated with each triangle in the domain triangulation.

Fig. 2.11). 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 (see (Dahmen, Micchelli and Seidel, 1992) for the details), where $\mathbf{u} = (u, v)$ defines the triangulated parametric domain of the surface. We then define a bivariate B-spline basis function as

$$N_{\mathbf{i},\beta}(\mathbf{u}) = d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2}) M(\mathbf{u}|V_{\mathbf{i},\beta}), \tag{2.16}$$

where $d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$ is twice the area of $\Delta(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$, which can be explicitly expressed as

$$d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2}) = \left| \det \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{r}_{\beta_0} & \mathbf{s}_{\beta_1} & \mathbf{t}_{\beta_2} \end{pmatrix} \right|.$$

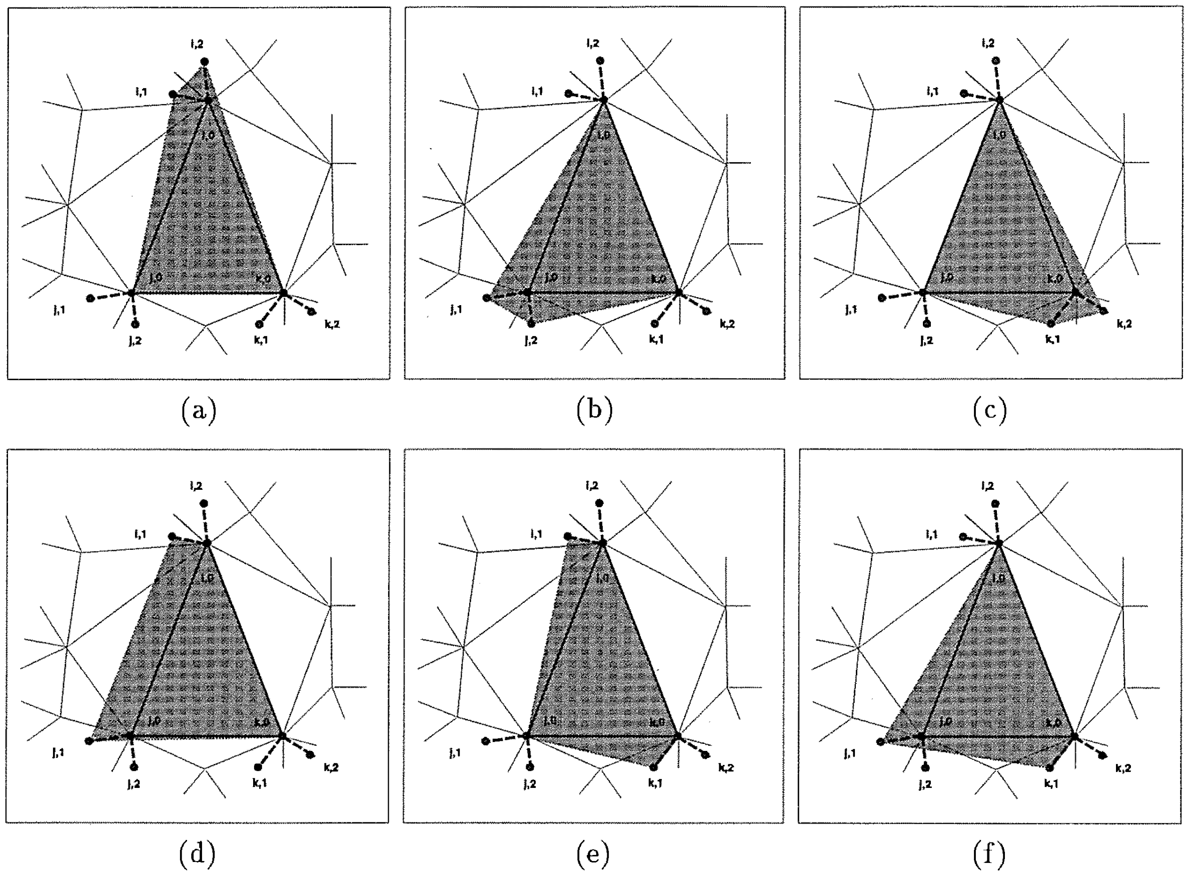


Figure 2.12: Non-zero parametric domains of quadratic basis functions over $\Delta(i,j,k)$. (a) $N_{ijk,200}$. (b) $N_{ijk,020}$. (c) $N_{ijk,002}$. (d) $N_{ijk,110}$. (e) $N_{ijk,101}$. (f) $N_{ijk,011}$.

Like the ordinary tensor-product B-spline, the triangular B-spline surface of degree n defined over an arbitrary triangulated domain is the combination of a set of basis functions with control points $\mathbf{p}_{\mathbf{i},\beta}$:

$$\mathbf{s}(\mathbf{u}) = \sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u}). \tag{2.17}$$

Fig. 2.12 highlights non-zero parametric domains of 6 basis functions defined over a triangle $\Delta(i,j,k)$ of a quadratic surface.

2.4.2 Properties

Triangular B-splines are ideal for geometric design applications because of their many nice properties such as lower polynomial degree and optimum global smoothness with high flexibility. First, their locally defined basis functions are nonnegative piecewise polynomials which sum to unity. Second, polynomial surfaces with degree n can be C^{n-1} continuous if their knots are in general positions. The designer can achieve various smoothness requirements through knot variation. For instance, in quadratic triangular B-splines, the six linearly independent bivariate basis functions are defined over convex hulls spanned by five knots. Making four of the knots collinear renders the corresponding basis function discontinuous along the line. Making three knots collinear leads to a discontinuity of the first derivative. Finally, triangular B-splines have the convex hull property and are affine invariant under standard geometric transformations. Since any piecewise polynomial with degree n over a triangulation can be represented as a linear combination of triangular B-splines (Greiner and Seidel, 1994), they provide a unified representation scheme for polynomial models with arbitrary topology.

Chapter 3

From Geometric to Physics-Based Design

In this chapter we survey the progression of computer-aided design developed during the past several decades from the purely geometric techniques to the physics-based paradigm. We begin with a review of geometric design and discuss some of its limitations. We then review variational design which addresses some of these limitations. Finally we review prior work in physics-based design which may be viewed as a generalization of variational design. The advantages of the physics-based modeling approach will serve as the motivation for D-NURBS.

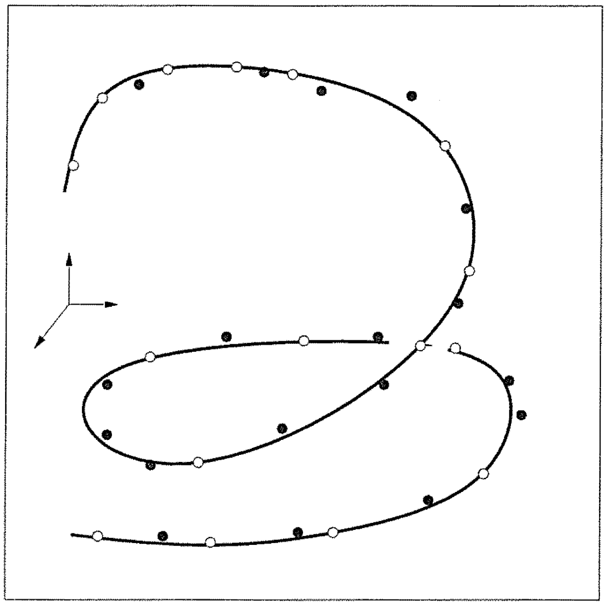


Figure 3.1: Curve interpolation/approximation.

3.1 Geometric Design Paradigms

Many design techniques have been developed for CAGD to achieve various design and manufacturing requirements. First, designers can specify geometric entities by either interpolating or approximating a set of regular data points, scattered data points, or boundary curves. Fig. 3.1 illustrates a curve that interpolates (and approximates) a set of points. Second, designers can indirectly manipulate the DOFs of the underlying geometric formulation to achieve interactive shape design. Third, through the process of cross-sectional design, they can design surfaces by specifying generator curves.

3.1.1 Interpolation

Scientific and engineering applications such as medical imaging, automobile and aircraft design, and geological terrain modeling make use of interpolation techniques. Data interpolation with polynomial splines can be formulated and solved through a set of linear equations. To obtain a unique solution, the number of unknowns must equal the number of independent constraints. For instance, a B-spline curve with $n+1$ control points (see (2.5)) can be used to interpolate independent $n+1$ data points $\mathbf{d}_i$. Assuming the corresponding parametric values $u_0, \ldots, u_n$ are provided, unknown control points can be obtained by solving

$$\begin{bmatrix} B_{0,k}(u_0) & \cdots & B_{n,k}(u_0) \\ \vdots & \ddots & \vdots \\ B_{0,k}(u_n) & \cdots & B_{n,k}(u_n) \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_n \end{bmatrix} = \begin{bmatrix} \mathbf{d}_0 \\ \vdots \\ \mathbf{d}_n \end{bmatrix} \tag{3.1}$$

Let $\mathbf{C}$ be the coefficient matrix in (3.1), $\mathbf{p} = [\mathbf{p}_0, \ldots, \mathbf{p}_n]^\mathsf{T}$, $\mathbf{d} = [\mathbf{d}_0, \ldots, \mathbf{d}_n]^\mathsf{T}$, then (3.1) can be written as

$$\mathbf{Cp} = \mathbf{d}$$

Note that, although the B-spline supports local control, B-spline interpolation is a global scheme. The modification of one data point affects the whole curve. Furthermore, if the parameterization is unknown, (3.1) is no longer a linear system. The chord-length schemes have been commonly used for the curve parameterization. If only positional data are available, derivative information must be estimated based on the given data to produce a smooth curve. Many automatic methods have been derived for this purpose (see (Bohm, Farin and Kahmann, 1984) for the details). To achieve data interpolation with tensor-product surfaces, data points must usually conform to a rectangular grid structure.

Another data interpolation approach is the lofting technique which allows a smooth surface to pass through a set of cross-sectional curves. Boolean sum surfaces such as Gordon, Coons, and Ball surfaces are all constructed through the interpolation of a lattice of curves. Among various cross-sectional design techniques, skinning creates a surface interpolating a set of isoparametric curves (see Fig. 3.2). To interpolate non-isoparametric curves, reparameterization is necessary. Ferguson and Grandine proposed an approach which supports the interpolation of a set of non-constant parameter curves (Ferguson and Grandine, 1990).

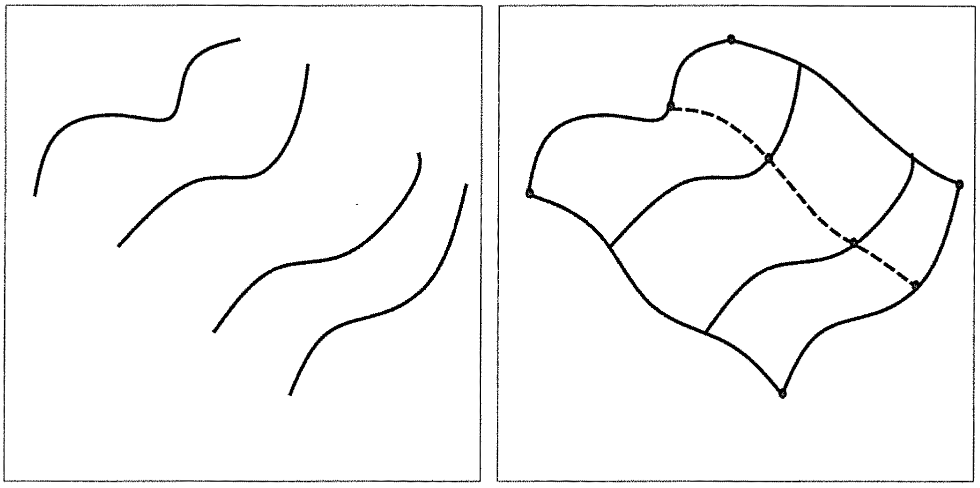


Figure 3.2: The skinning operation (a) isoparametric curves (b) the skinning surface.

Much work has been done on spline interpolation. Lounsbery, Mann and DeRose surveyed various interpolation methods over a triangulated polyhedron (Lounsbery, Mann and DeRose, 1992). Forrest discussed iterative interpolation and approximation of Bezier polynomials (Forrest, 1990). Dyn, Levin, and Gregory used a subdivision surface with tension control to interpolate data points (Dyn, Levin and Gregory, 1990). Geometric splines have become increasingly popular. With geometric continuity, extra degrees of freedom (usually expressed as shape parameters) provide designers additional shape handles. Shape parameters have a clear geometric interpretation. Automatic selection of these parameters is highly desirable from designers' point of view. A procedural method (Shirman and Sequin, 1992) exploits piecewise cubic Bezier curves to construct a G^1 interpolant with shape parameters. The procedure automatically determines tangent direction and two derivative magnitudes using an intuitive geometric ruled-based approach. Also, the procedural methods can provide pleasing curves for a set of irregular interpolation points.

Scattered data interpolation (see Fig. 3.3 for an illustration) has also been widely applied especially in scientific computing. Shepard originally formulated the scattered data interpolation as the problem of finding $f(u, v)$ with $f(u_i, v_i) = f_i$, where (u_i, v_i) are irregularly

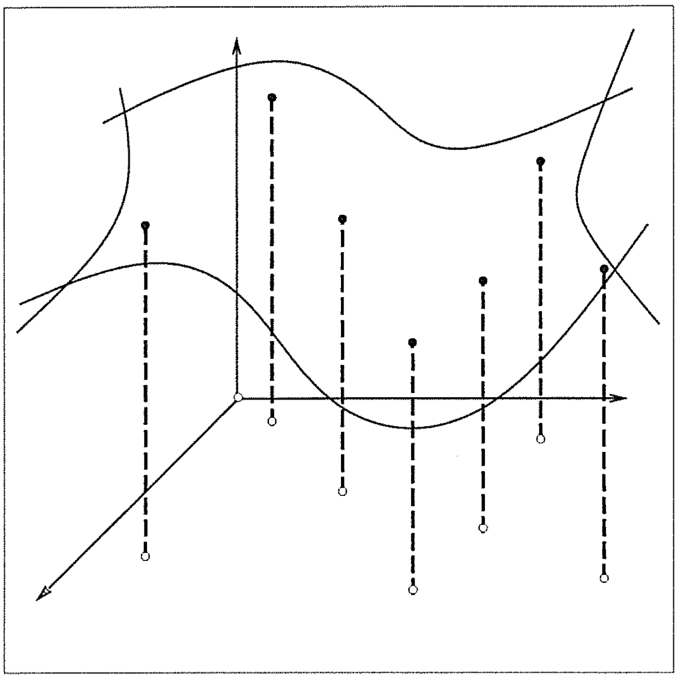


Figure 3.3: Scattered data interpolation.

distributed, $i = 0, \ldots, n$. Typical techniques are based on Shepard's formulas and Hardy's multiquadrics (Barnhill, 1983) which are all distance-weighted and application-dependent interpolants. Shepard's method constructs an interpolant using

$$f(u,v) = \frac{\sum_{i=0}^{i=n} f_i(u,v)/d_i(u,v)^2}{\sum_{i=0}^{i=n} 1/d_i(u,v)^2} \tag{3.2}$$

where $d_i(u,v)$ is the distance between (u,v) and (u_i,v_i), $f_i(u,v) = f_i + (u-u_i)a_i + (v-v_i)b_i$, a_i and b_i are estimated to approximate the tangent plane at (u_i,v_i). In spite of its simplicity, Shepard's method is a global scheme, it does not reproduce any local shape properties implied by the data because it has local extrema at data sites.

Recently, scattered data interpolation has been extended into the higher dimensional data (e.g. 4D data, surface-on-surface data). Nielson *et al.* investigated the scattered data interpolation over volumetric domains and arbitrarily shaped surface (surface-on-surface) domains (Nielson et al., 1991; Nielson, 1993). He used Hardy's multiquadric global interpolant which is defined as

$$f(x,y) = \sum_{i=0}^{n} \alpha_i \sqrt{(x-x_i)^2 + (y-y_i)^2 + R^2} \tag{3.3}$$

subject to the interpolation constraints: $f(x_i, y_i) = f_i, i = 0, \ldots, n$. The coefficients α_i are computed through a $(n+1) \times (n+1)$ linear system of equations based on interpolatory constraints. Note that the extremely large number of unorganized data can make this system over-determined. It will lead Hardy's method to a least-square approximation. Nielson and Ramaraj also introduced an minimum norm network approach using a set of cubic polynomials to interpolate scattered data points over a spherical domain (Nielson and Ramaraj, 1987). The interpolant is determined by functional minimization. The algorithm consists of (i) domain triangulation, (ii) construction of the boundary curve network which satisfies the optimization functional subject to the interpolation constraints, and (iii) patch generation from the curve network to cover the entire domain using triangular interpolants. Step (ii) and (iii) are executed via minimization.

3.1.2 Approximation

Shape approximation is necessary for several reasons.

Data Exchange: Different modeling systems often enforce certain limitations such as maximum allowable degree. To exchange geometric data among several design systems, users must approximate higher order geometric entities which can not be precisely represented in the desired system because of the limitations.

Data Reduction: It can be formally defined as: given a set of control points with basis functions of degree n (e.g. Bezier curve), find another set of control points with basis functions of degree $m < n$, such that the new representation is the best approximation of the original one. Knot removal is a commonly used technique for data reduction. It removes knots from a spline without perturbing the spline more than a given tolerance. Scattered data (especially noisy data) fitting is another typical example for data reduction.

Decomposition: Geometric modeling and data visualization often require a large amount of data. As more and more data are processed by computers, transmission and storage becomes a bottleneck for the geometric process. In order to implement a more economical representation and efficient analysis, multi-resolution techniques are often used.

Hierarchical Modeling: The same geometric object is represented at different levels. It starts with a rough global shape defined on a relatively coarse scale. The local details are only modeled on the refining scale. Hierarchical representation is useful for high speed rendering because it avoids the sampling problem.

Special Shapes: When a curve (or surface) has no simple or compact mathematical representation (*e.g.* the intersecting curve of surfaces, or the offset of curves or surfaces), shape approximation is necessary.

If m data points are provided in (3.1), where $m > n + 1$, the coefficient matrix in (3.1) becomes an $m \times (n + 1)$ matrix. Thus, this linear system becomes overdetermined. The interpolation is transformed into approximation. It can be solved as follows

$$\mathbf{C}^{\mathsf{T}}\mathbf{C}\mathbf{p} = \mathbf{C}^{\mathsf{T}}\mathbf{d}, \tag{3.4}$$

where $\mathbf{C}^{\mathsf{T}}\mathbf{C}$ is the new squared coefficient matrix, $\mathbf{p}$ is the unknown control point vector, and $\mathbf{d}$ is the given data point vector. It can be easily verified that the solution of (3.4) also minimizes the error functional

$$E = \sum_{i=0}^{m} \|\mathbf{c}(u_i) - \mathbf{d}_i\|^2$$

In general, the whole approximation procedure involves parameterization, least-squares fitting and parameter optimization. Chord-length parameterization is often used for curve approximation. Hoschek applied non-linear parameter optimization to the degree reduction of Bezier splines (Hoschek, 1987). Sarkar and Menq presented an algorithm to implement smooth least-square approximation using cubic B-splines, where the parameterization is formulated as a nonlinear minimization problem (Sarkar and Menq, 1991b; Sarkar and Menq, 1991a). In most applications, data points are not regularly distributed. Hoppe *et al.* presented an algorithm which can reconstruct a polygonal approximation from unorganized data points by estimating the tangent plane for each data point (Hoppe et al., 1992). The geometry and topology can be inferred automatically from the data. However, the input data must be dense because it is impossible to recover features where there is insufficient sampling. Very often, a set of data points can not be sampled exactly, Cheng and Barsky presented an algorithm that uses a cubic spline curve to interpolate specified data points at some knots and pass through specified regions at other knots while minimizes the energy (Cheng and Barsky, 1991). Chou and Piegl used piecewise cubic rational Bezier splines to fit a set of data points and their tangent directions in the least-square sense (Chou and Piegl, 1992).

Although rational curves and surfaces started to be widely used in CAGD only during the past ten years, rational functions have been studied for many decades in approximation theory. In general, rational approximation provides better accuracy. Nevertheless, it is difficult to reduce rational approximation into a linear system. Recently, Pratt, Goult and Ye proposed an efficient linearized approximation of rational functions (Pratt, Goult and

Ye, 1993). If $\mathbf{g}(u) = \mathbf{p}(u)/q(u)$ is a rational polynomial, where $\mathbf{p}(u) = \sum_{i=0}^{m} \mathbf{p}_i A_i(u)$, $q(u) = \sum_{j=0}^{n} q_i B_i(u)$, the minimization of the error functional

$$E(\mathbf{g}) = \int_0^1 \left\| \mathbf{f}(u) - \frac{\mathbf{p}(u)}{q(u)} \right\|^2 du$$

reduces to a set of nonlinear equations. Although it can be solved by iterative methods in practice, the result largely depends on a good initialization. To avoid the nonlinearity, alternatively, one can minimize the linear functional

$$\hat{E}(\mathbf{g}) = \int_0^1 \|\mathbf{f}(u)q(u) - \mathbf{p}(u)\|^2 du$$

This linear minimization process can be further divided into two steps. First, solve for

$$\frac{\partial \hat{E}}{\partial \mathbf{p}_i} = 0,$$

where $\mathbf{p}_i$ is evaluated in terms of $q(u)$. Second, solve for

$$\frac{\partial \hat{E}}{\partial q_j} = 0,$$

where the unknown q_j of $q(u)$ are determined uniquely. Although it has been shown that this linearized functional produces good results for many applications, it is not the least-squares (best) approximation of the rational function.

3.1.3 Interactive Modification

Due to measurement errors and the subjective judgement of designers, there are no objective *best* shapes. Thus, it is more reasonable to apply interactive techniques. Conventionally, interactive design is accomplished through control polygon manipulation. If geometric continuity is used, extra shape parameters such as tensions can also be exploited to implement interactive control. The interactive procedure must be easy to use. In addition, the underlying mathematical formulation should be transparent to users who may have little mathematical background.

There exists a large variety of spline formulations. Each formulation encourages the implementation of certain styles of interaction. Even though the mathematical power of two formulations may be equivalent, the ease of use from the designer's viewpoint may vary dramatically. For instance, the cubic B-spline without multiple knots is inherently C^2 continuous. The cubic Bezier curve, however, possesses C^2 continuity only when its control points are subject to certain geometric constraints. Extra design requirements to ensure the piecewise Bezier curve be C^2 must be enforced. This may be quite expensive because the constraints have to be maintained throughout the whole design process. Bartels et al. presented a general statistical analysis approach to measure and analyze user performance with interactive curve manipulation of a single control point (Bartels et al., 1993). B-splines, Bezier splines, Catmull-Rom splines, and two other C^2 interpolating splines are investigated. It has been demonstrated that there is a significant performance difference among these five formulations. The B-spline formulation is the best in terms of both match quality and time cost of achieving the objective for the curve case.

Cross-sectional design allows control polygons of isoparametric curves to be used as tools for interactive surface refinements. High-level operators such as bends, twists and free-form deformations (FFD) are also applicable. But traditional free-form interaction has long been control point based. Control point manipulation is an indirect, often unnecessarily tedious solution for interactive modeling. Although the B-splines have been very popular because of their interaction convenience, the B-spline formulation does not support direct shape interaction. Direct and precise free-form shape manipulation is often desired. Fowler presented a new technique to kinematically manipulate position and/or normal at arbitrary selected points on the tensor-product B-spline surface (Fowler, 1992). His approach is equivalent to solving an underdetermined linear system of equations. To derive a unique solution, extra geometric constraints are employed to minimize the combined movement of the control vertices.

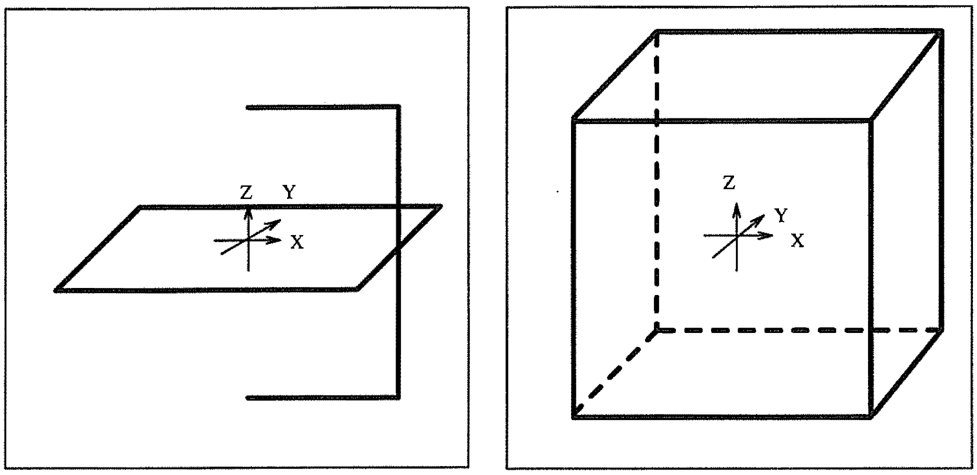
Deformation is a highly intuitive and interactive operation, it can be used to generate large families of geometric shapes. Barr proposed twisting, bending, and tapering transformation of geometric objects which can be used to create complex objects from simpler ones (Barr, 1981). Globally and locally defined deformations may be used as hierarchical operations, expanding the CAD/CAM repertoire. Forsey and Bartels presented Hierarchical B-splines which support local refinement and manipulation on B-spline surfaces (Forsey and Bartels, 1988). In general, refinement is implemented through knot insertion. Knot insertion on the surface will introduce more control points outside the local refining region. Local hierarchical refinement on the B-spline surface can be achieved through the use of an overlay surface and offset referencing of control vertices. Galyean and Hughes presented a new interactive modeling technique for solid sculpting (Galyean and Hughes, 1991). Controlled by a 3D input device, a sculpting tool can modify the voxel array values which represent the volumetric material. A set of sculpting tools such as cutting and adding is provided. The algorithm allows the modification of the material through purely geometric means.

3.1.4 Cross-Sectional Design

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 (Faux and Pratt, 1979). Cross-sectional design operations such as skinning, sweeping, and swinging have been widely used for interactive shape modification. The fundamental requirements are: (i) the surface must be visually smooth if all section curves are visually smooth, (ii) no unnecessary undulation, and (iii) affine invariance.

Originating in lofting theory, a skinning process generates a surface passing through a series of cross-sectional profiles. Surface design can thus be reduced to the design of (a) a series of compatible cross-sectional profiles, and (b) a path trajectory. Extra effort has to be done to make all section curves compatible. One typical technique to achieve this goal is degree elevation. Because of the lack of 3D interaction techniques and tools, sectional curves are often created as planar profiles. But trial-and-error procedures used for designing fair curves can be laborious. To avoid spatial surface interaction, projection curves are used for interrogation and necessary modification. Other compromises must be made towards the conflicting parameterization demands. Sweeping is a special case of skinning where a set of constant section curves are provided. The construction of the surface also depends on the parameterization of the path (spine) curve.

Barr employed a spherical cross-product to construct superquadrics from profiles (Barr, 1981). Woodward (Woodward, 1987) introduced the swinging operator by extending the spherical cross-product with a scaling factor, and applied it to generate surfaces with B-spline profile curves (see also (Cobb, 1984)). Piegl carried the swinging idea over to NURBS curves (Piegl, 1991). He proposed NURBS swung surfaces, a special type of NURBS surfaces formed by swinging one planar NURBS profile curve in the x-z plane along a second NURBS trajectory curve in the x-y plane. For example, Fig. 3.4 illustrates the design of a cubical NURBS swung surface from two NURBS profile curves. Swinging generalizes rotational sweeping. Designers can reposition control points, change weights, modify the knot vector, and move the data points of sectional curves in order to modify the surface shape. Several geometric shape design systems, including the recent one in (Snyder and Kajiya, 1992),



(a) (b)

Figure 3.4: 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.

include some form of swinging (or sweeping) among their repertoire of techniques.

3.2 Limitations of Geometric Design

NURBS have become a *de facto* standard in commercial modeling systems. Fig. 3.5 illustrates that NURBS can be used to integrate various geometric forms. Despite the extraordinary flexibility of NURBS, traditional design methodology can not exploit their full potential. This is because NURBS have been viewed as purely geometric primitives, which require designers to interactively adjust many DOFs—control points and associated weights—to obtain desired shapes.

Conventional geometric design techniques previously described are kinematically driven. Despite modern interactive devices, this process can be clumsy and laborious when it comes to designing complex, real-world objects.

Rational splines such as NURBS provide better accuracy than ordinary polynomial

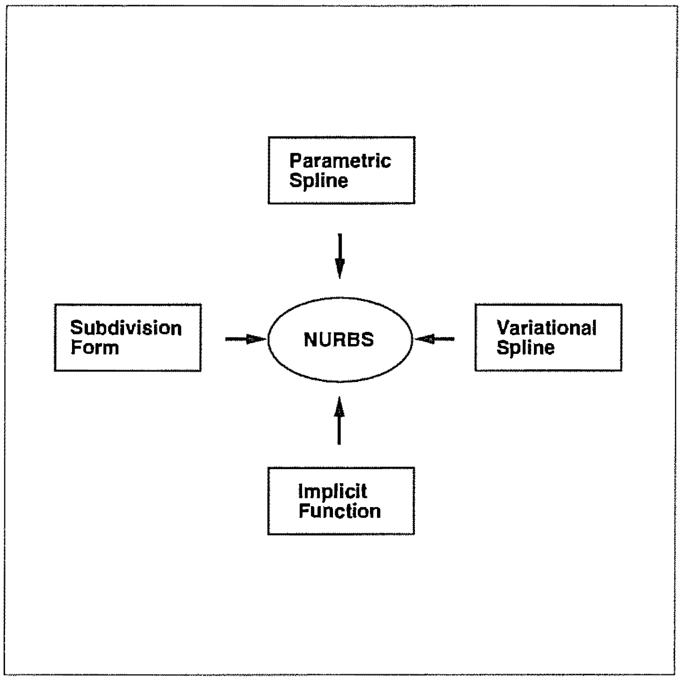


Figure 3.5: NURBS as an integrated form for various geometric entities.

functions when used for approximation. Although NURBS can be used to represent many analytical shapes such as conics precisely, no simple and flexible methods are offered to designers for the automatic selection of weights in an intuitive and meaningful way. Conventional methods through user specification of weights are both heuristic and *ad hoc*.

Given a set of empirical 3D data points provided by a scanner, the task of inferring both geometry and topology from this scattered data is extremely difficult. This is because no implicit neighborhood information is available and the data are often subject to measurement error.

Flexible and real-time interaction between designers and the modeling systems is crucial not only for future CAD systems but also for other diverse applications such as architecture design, surgical planning, and robotics. Directly manipulating geometry in an interactive environment requires the integration of virtual reality techniques with well established CAD design and manufacture methodology. In contrast, dynamic 3D interaction of free-form geometric shapes in the CAD system has been rarely explored. As a result, the full potential of dynamic interaction is yet to be realized.

The above difficulties are due to the intrinsic features of traditional geometric modeling. Pure geometric representation is abstract. It does not have behavior. The design and manufacture process, however, requires a mechanism which can accomplish the interactive modification of geometric information both efficiently and precisely. It can be demonstrated that traditional design methodology can not offer us such a dynamic and interactive framework for time-varying requirements. Strongly motivated to bridge the large gap between the characteristics of geometry and the design and manufacture process requirements, we investigate how to associate physics and physical behavior with abstract and static geometry.

3.3 Variational Design

Designers not only require the shape to satisfy functional requirements such as interpolation and continuity but also need visually-pleasing and fair shapes implied in the given data points. For instance, if the data are locally monotonic or convex, the interpolation function should also be a shape-preserving interpolant. Shape-preservation requires shape fidelity reflected by the data beyond interpolation/approximation. Fig. 3.6 illustrates two typical examples of shape-preserving requirements.

In contrast, ordinary interpolating techniques usually generate shape oscillations not implied in data points. To achieve satisfactory design, both functional requirements and aesthetic criteria are needed. One of the most important problems in CAGD is to construct visually-pleasing (fair) splines that can either interpolate or approximate a given set

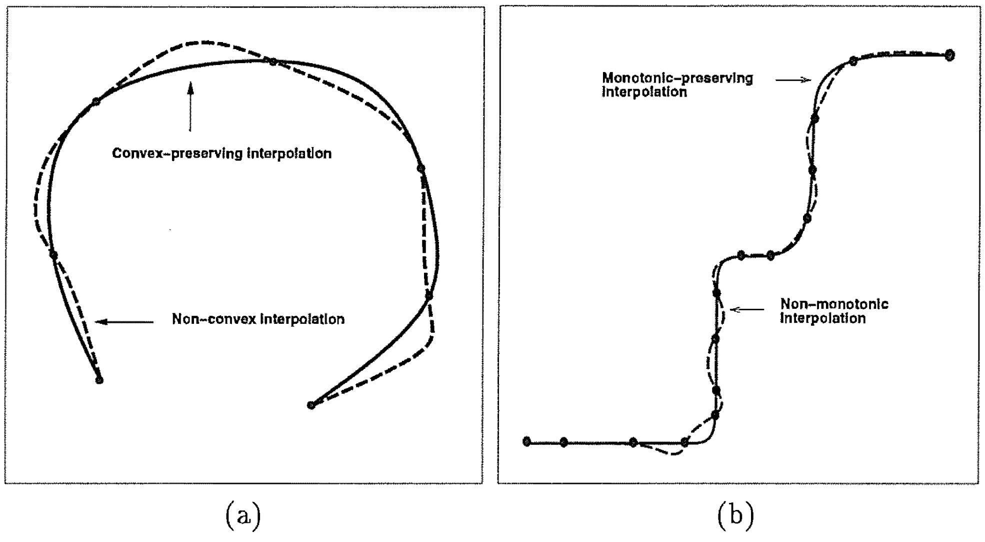

Convex-preserving interpolation
Non-convex interpolation
(a)
Monotonic-preserving interpolation
Non-monotonic interpolation
(b)

Figure 3.6: Shape-preserving interpolants (a) convex-preserving (b) monotonic-preserving.

of points. Note that, aesthetic criteria are very subjective because they are often style-dependent. It can be difficult to express such requirements formally. Nevertheless, quantitative judgement has to be formulated even before the design and manufacture process starts. The variational technique is more appropriate because it can implement qualitative and subjective criteria as quantitative ones.

Variational optimization is also used to determine the interpolant unknowns where only partial information is provided. Once interpolation constraints are satisfied, there are generally surplus degrees of freedom. The values of these free parameters dramatically influences the shape. Local methods apply heuristics to manually set free variables. This will not always achieve satisfactory results because it needs a lot of user intervention. Likewise, manual operation is not efficient for a large number of degrees of freedom. In contrast, global optimization methods allow the primitives to autonomously deform to minimize an energy functional subject to constraints. This approach requires less user input than conventional free-form modeling techniques. Moreover, the fairness criterion acting as an objective function discourages undesired oscillation.

Parkinson interpolated a set of data points with a set of biarcs (Parkinson, 1992).

Unknown gradients at data points are determined by the efficient linearized approximation of an energy minimization. Wever presented a global cubic C^2 interpolant using energy minimization (Wever, 1991). A non-negative exponential spline is derived as a shape-preserving interpolant for positive data (Wever, 1988), where tension parameters are used in the nonlinear constrained optimization of bending energy. Moreton and Sequin presented a simple mechanism which allows the creation of complex smoothly shaped surfaces of any topological types with piecewise biquintic Bezier patches (Moreton and Sequin, 1992). Nonlinear optimization techniques are used to minimize a fairness functional based on the variation of curvature in order to determine remaining free parameters unspecified through geometric interpolatory constraints. The use of the variation of curvature functional allows commonly used shapes such as spheres, cylinders and tori to be reconstructed subject to a set of compatible constraints. However, the computational cost of the modeling software is extremely expensive. It prevents this technique from being used for interactive surface design.

In (Sapidis and Farin, 1990), the fairness criterion is defined as the curvature plot which is a piecewise monotonic function with the fewest possible monotone pieces. To generate a fair curve, an interactive fairing process is often used in which users interactively modify control points repeatedly in terms of curvature plots until a curve of acceptable fairness is obtained. Manual adjustment is inefficient and completely incompatible with any automatically refining processes. For instance, when using B-splines to approximate data points, it is possible to exhibit unwanted behavior due to digitizing errors. Therefore, it is inevitable to have an unacceptable curvature plot. Sapidis and Farin developed an algorithm for locally fairing planar B-spline curves by repeatedly removing and reinserting knots of the spline (Sapidis and Farin, 1990). He further derived a simple geometric condition for the quadratic Bezier curve to ensure monotonic curvatures (Sapidis, 1992). This condition also provides a simple criterion which can be used for automatically correcting the curvature plot.

Optimal design requires designers (engineers and stylists) to focus on the shape instead of the abstract geometric representation. For this matter, functional constraints can be expressed in the form of partial differential equations with suitable boundary conditions (Lowe, Bloor and Wilson, 1990; Bloor and Wilson, 1990b). Although a wide range of surfaces can be generated based on various PDE requirements and boundary constraints, the PDE-based approach is not compatible with conventional spline techniques. It provides little intuition for shape adjustments. Also, it is difficult to determine how the variation of boundary curves affects the surface interior. Thus, the expertise of designers is necessary to achieve desirable shape refinement. To demonstrate that the PDE surface is a compatible approach with established techniques of mainstream surface design, B-splines have been used to approximate PDE surfaces (Bloor and Wilson, 1990a).

3.4 Physics-Based Design

The physics-based design paradigm can provide a means to overcome the above drawbacks. Free-form deformable models, initially introduced to computer graphics in (Terzopoulos et al., 1987) and further developed in (Terzopoulos and Fleischer, 1988; Platt and Barr, 1988; Szeliski and Terzopoulos, 1989; Pentland and Williams, 1989; Szeliski and Tonnesen, 1992; Metaxas and Terzopoulos, 1992) are particularly relevant. They are also useful for user interfaces, virtual reality, image processing, and geometric modeling. Physics-based models are governed by the mechanical laws of continuous bodies which can be expressed in the form of dynamic differential equations. The dynamic and realistic behaviors can be obtained by solving an associated motion equation numerically. To date, however, researchers in the fields of computer vision and graphics devote most of their endeavors to the investigation of constraints, articulated rigid or nonrigid body synthesis, and complex scene control. Less effort has been applied to free-form dynamic interaction between designers and individual manufactured objects which is especially useful for geometric design.

Physical simulation can be used as an effective interactive tool for building and manipulating a wide range of models. It supports the dynamic manipulation of complex physical models. Terzopoulos and Fleischer demonstrated simple interactive sculpting using viscoelastic and plastic models (Terzopoulos and Fleischer, 1988).

Celniker and Gossard developed an interesting prototype system (Celniker and Gossard, 1991) for interactive free-form design based on the finite-element optimization of energy functionals proposed in (Terzopoulos and Fleischer, 1988). 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. Celniker and Welch investigated deformable B-splines with linear constraints (Celniker and Welch, 1992).

Bloor and Wilson developed related models using similar energies and numerical optimization (Bloor and Wilson, 1990b), and in (Bloor and Wilson, 1990a) they proposed the use of B-splines for this purpose. Subsequently, Celniker and Welch investigated deformable B-splines with linear constraints (Celniker and Welch, 1992).

Welch and Witkin extended the approach to trimmed hierarchical B-splines (see also (Forsey and Bartels, 1988)) for interactive modeling of free-form surface with constrained variational optimization (Welch and Witkin, 1992). The traditional control point approach is intuitive by allowing a conceptually simple change. However, to enforce a precise modification, many–even all– the control points have to be repositioned in order to achieve the desired effect.

Motivated by the fact that splines provide insufficient detail for modeling certain natural shapes, such as terrains, and fractals provide insufficient shape control, Szeliski and Terzopoulos proposed constrained fractals, a hybrid of deterministic splines and stochastic fractals which combines their complementary features (Szeliski and Terzopoulos, 1989). Through the use of the energy minimization principles the constrained fractal can be applied to synthesize realistic terrain surface from sparse elevation data. Multiresolution stochastic relaxation is used to compute fractals efficiently.

Thingvold and Cohen proposed a deformable model based on a B-spline surface, whose control points are mass points connected by elastic springs and hinges (Thingvold and Cohen, 1990). The control polygon refinement conserves physical quantities such as mass, spring, and hinge. Pentland et al. used a modal analysis method to decompose non-rigid dynamics into a set of independent vibration modes based on eigenvalues (Pentland and Williams, 1989; Pentland and Sclaroff, 1991; Pentland and Horowitz, 1991). Discarding high-frequency modes, the number of unknowns can be largely reduced while preserving the accuracy and generality of the formulation. A special global polynomial deformation can be associated with a set of vibration modes and applied to the animation with a superquadric ellipsoid.

Extracting geometric information from scattered data is important for visualization and object recognition. Miller et al. presented a method generating a simple topologically closed geometric model from a volumetric data set (Miller et al., 1991). The polyhedron model can expand itself like a balloon until it reaches the volumetric boundary of the scanned object through a relaxation process which also minimizes the prescribed cost function. Global subdivision is needed wherever appropriate for complex shape during the optimization process.

Szeliski and Tonnesen presented a new model of elastic surfaces based on interactive particle systems (Szeliski and Tonnesen, 1992). New particles are added into the system automatically which enables the surface to stretch and grow. Particle based surfaces can split and join without manual intervention. In spite of the interactive power for free-form modeling, particle based surfaces have some disadvantages, such as the lack of a precise and compact mathematical representation which presumably is vital in engineering applications.

In summary, physics-based models have dynamic behavior which is governed by physical laws. This allows designers to directly manipulate and interactively sculpt shapes using a variety of force-based tools. The energy-based optimization process can be implemented automatically. In addition, physics-based shape design can free designers from making nonintuitive decisions such as assigning weights to NURBS or determining shape parameters in variational splines. Furthermore, with physics-based direct manipulation, non-expert users are able to concentrate on visual shape variation without needing to comprehend the underlying mathematical formulation.

3.5 D-NURBS for Physics-Based Design

By marrying advanced geometric modeling with computational physics, we propose and develop D-NURBS, a physics-based generalization of geometric NURBS for geometric design, which will be presented in detail in the following chapters. The significant advantage of this physics-based design framework is that it integrates force-based dynamic manipulation and interactive sculpting with all existing NURBS geometric features and toolkits.

Dynamic NURBS are motivated by prior research aimed at applying the deformable modeling approach to shape design. In (Bloor and Wilson, 1990a; Celniker and Welch, 1992; Welch and Witkin, 1992), 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 (Celniker and Welch, 1992) 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 (Metaxas and Terzopoulos, 1992) for converting arbitrary geometric models into dynamic models using Lagrangian mechanics and finite element analysis. Our approach is systematic. We formulate D-NURBS curves and surfaces and reduce them to algorithms in a principled way, without resorting to any of the *ad hoc* assumptions of prior schemes (cf. (Thingvold and Cohen, 1990)). 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 (Zienkiewicz, 1977).

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, this does not preclude interactive performance on current graphics workstations, at least for the size of surface models with hundreds of patches 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

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

integral expressions for the matrix entries.

Chapter 4

Curve and Tensor Product D-NURBS

In this chapter, we formulate physics-based D-NURBS curves and tensor-product surfaces based on their geometric counterparts. The shape parameters (control points and associated weights) of geometric NURBS become the generalized (physical) coordinates of D-NURBS. We introduce time, mass, and a deformation energy into the standard NURBS formulation and employ Lagrangian dynamics to arrive at a system of nonlinear ordinary differential equations that govern the shape and motion of D-NURBS. Note that, the adjustment of the NURBS knot vector also influences their shape (refer to major CAGD textbooks such as (Farin, 1990) for the details). This dissertation does not treat the (non-uniform) knot sequence as free parameters like the control points and weights in D-NURBS. The modification of the knot vector may be achieved through conventional geometric means within the D-NURBS modeling environment. The incorporation of the NURBS knot vector into the generalized coordinates of D-NURBS remains an open problem. It is addressed as one the future research topics in Chapter 10.

4.1 Kinematic NURBS Curves

This and the next section formulate our physics-based D-NURBS curves and surfaces. The shape parameters of geometric NURBS, which were described in Section 2.2, play the role of generalized (physical) coordinates in D-NURBS.

For simplicity, consider first a D-NURBS space curve. The D-NURBS curve extends the geometric NURBS definition by explicitly incorporating time. It is a function of both the parametric variable u and time t:

$$\mathbf{c}(u,t) = \frac{\sum_{i=0}^{n} \mathbf{p}_i(t) w_i(t) B_{i,k}(u)}{\sum_{i=0}^{n} w_i(t) B_{i,k}(u)}. \tag{4.1}$$

The control points $\mathbf{p}_i(t)$ and weights $w_i(t)$, which are now functions of time, comprise the generalized coordinates of the D-NURBS curve. 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

$$\dot{\mathbf{c}}(u,\mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{4.2}$$

where an overstruck dot denotes a time derivative and $\mathbf{J}(u,\mathbf{p})$ is the Jacobian matrix.

4.1.1 Jacobian Matrix

Let us investigate the contents of $\mathbf{J}$. 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 $\partial\mathbf{c}/\partial w_i$, $i = 0,\ldots,n$.

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 $N_i(u,\mathbf{p})$

$$\mathbf{B}_i(u,\mathbf{p}) = \begin{bmatrix} N_i(u,\mathbf{p}) & 0 & 0 \\ 0 & N_i(u,\mathbf{p}) & 0 \\ 0 & 0 & N_i(u,\mathbf{p}) \end{bmatrix}$$

where

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

and let the 3-vector

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

The subscript x, y, and z denote the component of a 3-vector. We collect the $\mathbf{B}_i$ into $\mathbf{B}$

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

$$\mathbf{B}(u,\mathbf{p}) = \begin{bmatrix} \mathbf{B}_0 & \cdots & \mathbf{B}_n \end{bmatrix},$$

$$\mathbf{W}(u,\mathbf{p}) = \begin{bmatrix} \mathbf{w}_0 & \cdots & \mathbf{w}_n \end{bmatrix}.$$

The Jacobian matrix may then be written as

$$\mathbf{J}(u,\mathbf{p}) = \begin{bmatrix} \mathbf{B}_0 & \mathbf{w}_0 & \cdots & \mathbf{B}_n & \mathbf{w}_n \end{bmatrix}. \tag{4.3}$$

Using the foregoing notations, we can express

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

Next, we show that

$$\mathbf{W}\mathbf{p}_w = \mathbf{0}. \tag{4.4}$$

Clearly,

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

and

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

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 (4.4), hence we can express the D-NURBS as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{c}(u, \mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{4.5}$$

Moreover, taking the time derivative of (4.5) yields

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

Given (4.2), it follows that $\dot{\mathbf{J}}\mathbf{p} = \mathbf{0}$. This will enable us to simplify the discretized version of the D-NURBS differential equations and arrive at an efficient numerical implementation.

4.2 Kinematic NURBS Surfaces

A tensor-product D-NURBS surface has a similar structure to the curve. Proceeding analogously from (2.11), we define

$$\mathbf{s}(u, v, t) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j}(t) w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}. \tag{4.6}$$

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 (4.2) and (4.5), we obtain for the D-NURBS surface

$$\mathbf{s}(u, v, \mathbf{p}) = \mathbf{J}\mathbf{p}, \tag{4.7}$$

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}. \tag{4.8}$$

However, the contents of the Jacobian $\mathbf{J}$ differ from those in the curve case.

4.2.1 Jacobian Matrix

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 diagonal entries are the rational basis functions $N_{i,j}(u, v, \mathbf{p})$

$$\mathbf{B}_{i,j}(u, v, \mathbf{p}) = \begin{bmatrix} N_{i,j}(u, v, \mathbf{p}) & 0 & 0 \\ 0 & N_{i,j}(u, v, \mathbf{p}) & 0 \\ 0 & 0 & N_{i,j}(u, v, \mathbf{p}) \end{bmatrix}$$

where

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

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,

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

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

4.3 Lagrangian Mechanics

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 (Gossick, 1967). 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

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{p}_i} - \frac{\partial T}{\partial p_i} + \frac{\partial F}{\partial \dot{p}_i} + \frac{\partial U}{\partial p_i} = f_i. \tag{4.9}$$

Variants of this equation have served as the basis for deformable model formulations (Terzopoulos and Fleischer, 1988). Using (4.9), 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 (Metaxas and Terzopoulos, 1992).

4.4 D-NURBS Dynamics

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

$$T = \frac{1}{2} \iint \mu \dot{\mathbf{s}}^\top \dot{\mathbf{s}}\, du\, dv = \frac{1}{2} \dot{\mathbf{p}}^\top \mathbf{M} \dot{\mathbf{p}}, \tag{4.10}$$

where (using (4.8))

$$\mathbf{M}(\mathbf{p}) = \iint \mu \mathbf{J}^\top \mathbf{J}\, du\, dv \tag{4.11}$$

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

$$F = \frac{1}{2} \iint \gamma \dot{\mathbf{s}}^\top \dot{\mathbf{s}}\, du\, dv = \frac{1}{2} \dot{\mathbf{p}}^\top \mathbf{D} \dot{\mathbf{p}}, \tag{4.12}$$

where

$$\mathbf{D}(\mathbf{p}) = \iint \gamma \mathbf{J}^\top \mathbf{J}\, du\, dv \tag{4.13}$$

is the damping matrix.

For the elastic potential energy of D-NURBS, we can adopt the *thin-plate under tension* energy model (Terzopoulos, 1986), which was also used in (Celniker and Gossard, 1991; Welch and Witkin, 1992) (other energies are possible, including the nonquadratic, curvature-based energies in (Terzopoulos and Fleischer, 1988; Moreton and Sequin, 1992)):

$$\begin{aligned} U = \frac{1}{2} \iint \Bigg(& \alpha_{1,1} \frac{\partial \mathbf{s}^\top}{\partial u} \frac{\partial \mathbf{s}}{\partial u} + \alpha_{2,2} \frac{\partial \mathbf{s}^\top}{\partial v} \frac{\partial \mathbf{s}}{\partial v} + \beta_{1,1} \frac{\partial^2 \mathbf{s}^\top}{\partial u^2} \frac{\partial^2 \mathbf{s}}{\partial u^2} \\ & + \beta_{1,2} \frac{\partial^2 \mathbf{s}^\top}{\partial u \partial v} \frac{\partial^2 \mathbf{s}}{\partial u \partial v} + \beta_{2,2} \frac{\partial^2 \mathbf{s}^\top}{\partial v^2} \frac{\partial^2 \mathbf{s}}{\partial v^2} \Bigg) du\, dv = \frac{1}{2} \mathbf{p}^\top \mathbf{K} \mathbf{p}. \end{aligned} \tag{4.14}$$

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.[1] In view of (4.7), the $N \times N$

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

stiffness matrix is

$$\mathbf{K}(\mathbf{p}) = \iint \left(\alpha_{1,1}\mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2}\mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1}\mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2}\mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2}\mathbf{J}_{vv}^\top \mathbf{J}_{vv}\right) du\, dv, \tag{4.15}$$

where the subscripts on **J** denote parametric partial derivatives.

We can simplify D-NURBS equations of motion. Applying (4.9), the D-NURBS motion equations are

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top - \dot{\mathbf{M}}\dot{\mathbf{p}} - \begin{bmatrix} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial p_i}\mathbf{p} & \cdots \end{bmatrix}^\top . \tag{4.16}$$

The two vectors involving **M** on the right side of (4.16) may be be combined into a single vector:

$$\dot{\mathbf{M}}\dot{\mathbf{p}} - \begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top = \mathbf{I}\dot{\mathbf{p}}. \tag{4.17}$$

Using the product rule of differentiation, we have $\dot{\mathbf{M}} = \mathbf{I} + \mathbf{I}^\top$. For (4.17) to hold, we must have

$$\mathbf{I}^\top \dot{\mathbf{p}} = \begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial p_i}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top . \tag{4.18}$$

It is obvious from (4.10) that the two sides of (4.18) are integrals of the two vectors, respectively. The two vectors in (4.18) are equal when, for $i = 1, \ldots, N$, (note that we assume **J** is the concatenation of N vectors $\mathbf{j}_i$),

$$\dot{\mathbf{j}}_i^\top \mathbf{J}\dot{\mathbf{p}} = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial(\mathbf{J}^\top \mathbf{J})}{\partial p_i}\dot{\mathbf{p}}. \tag{4.19}$$

We now prove (4.19). 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, (4.19) is proved.

Next, we derive another mathematical identity:

$$\begin{bmatrix} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial p_i}\mathbf{p} & \cdots \end{bmatrix}^\top = \mathbf{0}. \tag{4.20}$$

The left side of (4.20) is the integral of the summation of the five terms of (4.15). 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

$$\mathbf{p}^{\top}\frac{\partial}{\partial p_i}\left(\frac{\partial \mathbf{J}^{\top}}{\partial u}\frac{\partial \mathbf{J}}{\partial u}\right)\mathbf{p} = 0. \tag{4.21}$$

Now, (4.21) is the ith component of the first vector on the left side of (4.20). Similarly, the other four vectors inside the integral operator of the left hand side of (4.20) are all zero.

Thus, we demonstrate that by applying (4.9), the D-NURBS equations of motion are given by

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \mathbf{I}\dot{\mathbf{p}}, \tag{4.22}$$

where the generalized force vector, obtained through the principle of virtual work (Gossick, 1967) done by the applied force distribution $\mathbf{f}(u,v,t)$, is

$$\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{J}^{\top}\mathbf{f}(u,v,t)\,du\,dv, \tag{4.23}$$

and where

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

Chapter 5

Swung D-NURBS

In this chapter, we develop a physics-based generalization of the geometric NURBS swung surface. We refer to our new models as *swung D-NURBS*. The NURBS swung surface retains a considerable breadth of geometric coverage. It can represent common geometric primitives such as spheres, tori, cubes, quadrics, surfaces of revolution, etc. Fig. 1.1 illustrates four NURBS swung surfaces with distinct topological structures. Fig. 3.4 illustrates the design of a cubical NURBS swung surface from two NURBS profile curves. In this chapter, we formulate swung D-NURBS in terms of the degrees of freedom of two generator D-NURBS curves. This allows us to derive equations of motion for swung D-NURBS associated with mass, damping, and deformation energy distributions. The NURBS swung surface is efficient compared to a tensor-product NURBS surface for designing symmetric shapes, inasmuch as it can represent a broad class of symmetric shapes with essentially as few degrees of freedom as it takes to specify the two generator curves that define the shape.

5.1 Kinematic NURBS Swung Surfaces

In this section, we formulate the underlying geometry of the dynamic swung surfaces and derive the Jacobian and basis function matrices that lead to succinct expressions analogous to (4.2) and (4.5) for the velocity and position functions of the surface, respectively.

Let the two generator curves $\mathbf{c}_1(u, \mathbf{a})$ and $\mathbf{c}_2(v, \mathbf{b})$ be of the form (4.1). The swung surface is then defined as

$$\mathrm{s}(u, v, t) = \begin{bmatrix} \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,x} & \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,y} & \mathbf{c}_{1,z} \end{bmatrix}^\top \tag{5.1}$$

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.

5.1.1 Jacobian Matrix

Denoting the Jacobian matrices of the two profile curves as $\mathbf{J}_1(u, \mathbf{a})$ and $\mathbf{J}_2(v, \mathbf{b})$, the curve position and velocity functions take the form of (4.2) and (4.5):

$$\mathbf{c}_1(u, \mathbf{a}) = \mathbf{J}_1\mathbf{a}, \qquad \dot{\mathbf{c}}_1(u, \mathbf{a}) = \mathbf{J}_1\dot{\mathbf{a}},$$
$$\mathbf{c}_2(v, \mathbf{b}) = \mathbf{J}_2\mathbf{b}, \qquad \dot{\mathbf{c}}_2(v, \mathbf{b}) = \mathbf{J}_2\dot{\mathbf{b}},$$

where $\mathbf{J}_1$ is a $3 \times 4(m+1)$ matrix, and $\mathbf{J}_2$ is a $3 \times 4(n+1)$ matrix. Both are of the form (4.3).

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

$$\mathbf{J}_1 = \begin{bmatrix} \mathbf{X}_1^\top & \mathbf{Y}_1^\top & \mathbf{Z}_1^\top \end{bmatrix}^\top$$

and

$$\mathbf{J}_2 = \begin{bmatrix} \mathbf{X}_2^\top & \mathbf{Y}_2^\top & \mathbf{Z}_2^\top \end{bmatrix}^\top .$$

The swung surface is therefore written as

$$\mathbf{s}(u, v, \mathbf{p}) = \begin{bmatrix} \alpha(t)(\mathbf{X}_1\mathbf{a})(\mathbf{X}_2\mathbf{b}) \\ \alpha(t)(\mathbf{X}_1\mathbf{a})(\mathbf{Y}_2\mathbf{b}) \\ \mathbf{Z}_1\mathbf{a} \end{bmatrix} . \tag{5.2}$$

The velocity of the swung surface is

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{L}\dot{\mathbf{p}} \tag{5.3}$$

where $\mathbf{L}(u, v, \mathbf{p})$ is the Jacobian matrix with respect to the generalized coordinate vector $\mathbf{p}$.

Hence, $\mathbf{L}$ comprises the vectors $\partial \mathbf{s}/\partial \alpha$, $\partial \mathbf{s}/\partial \mathbf{a}$, and $\partial \mathbf{s}/\partial \mathbf{b}$, which are given as follows:

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

where

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

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

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

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

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

$$\mathbf{L} = \begin{bmatrix} \mathbf{A}\mathbf{c}_2 & \mathbf{B}_\alpha\mathbf{J}_1 & \mathbf{A}_\alpha\mathbf{J}_2 \end{bmatrix} \tag{5.4}$$

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

5.1.2 Basis Function Matrix

Unlike $\mathbf{J}$ in (4.5), $\mathbf{L}$ cannot also serve as the basis function matrix of the swung surface. Let

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

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

It is straightforward to verify that

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

Thus we have

$$\mathbf{s}(u, v, \mathbf{p}) = \mathbf{H}\mathbf{p}, \tag{5.5}$$

where

$$\mathbf{H} = (\mathbf{L} + 2\mathbf{H}_4)/3 \tag{5.6}$$

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

5.2 Equations of Motion

To proceed with the Lagrangian formulation, we express the kinetic energy due to a prescribed mass distribution function $\mu(u, v)$ over the parametric domain of the surface and a Raleigh dissipation energy due to a damping density function $\gamma(u, v)$. To define an elastic potential energy, we adopt the *thin-plate under tension* energy model which has been applied in the previous chapter.

$$U = \frac{1}{2}\iint \left(\alpha_{1,1}\frac{\partial \mathbf{s}^\top}{\partial u}\frac{\partial \mathbf{s}}{\partial u} + \alpha_{2,2}\frac{\partial \mathbf{s}^\top}{\partial v}\frac{\partial \mathbf{s}}{\partial v} + \beta_{1,1}\frac{\partial^2 \mathbf{s}^\top}{\partial u^2}\frac{\partial^2 \mathbf{s}}{\partial u^2} \right.$$
$$\left. + \beta_{1,2}\frac{\partial^2 \mathbf{s}^\top}{\partial u \partial v}\frac{\partial^2 \mathbf{s}}{\partial u \partial v} + \beta_{2,2}\frac{\partial^2 \mathbf{s}^\top}{\partial v^2}\frac{\partial^2 \mathbf{s}}{\partial v^2}\right) du\, dv. \tag{5.7}$$

The $\alpha_{i,j}(u,v)$ and $\beta_{i,j}(u,v)$ are elasticity functions which control tension and rigidity, respectively, in the two parametric coordinate directions. Other energies are applicable, including the nonquadratic, curvature-based energies in (Terzopoulos et al., 1987; Moreton and Sequin, 1992)).

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

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \tag{5.8}$$

where the mass matrix is

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

the damping matrix is

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

and the stiffness matrix is

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

(the subscripts on **L** and **H** denote parametric partial derivatives). **M**, **D** and **K** are $M \times M$ matrices. The generalized force, obtained through the principle of virtual work (Gossick, 1967) done by the applied force distribution $\mathbf{f}(u,v,t)$ is

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

Because of the geometric nonlinearity, generalized inertial forces

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

are also associated with the models. The derivation of the equations of motion (5.8) proceeds in the same manner as for D-NURBS (see Chapter 4 for the details).

We examine the mass and damping matrices. Both matrices involve the integration of $\mathbf{L}^\top\mathbf{L}$ in the parametric domain where $\mathbf{L}$ is given in (5.4). Based on (5.4), $\mathbf{L}^\top\mathbf{L}$ is decomposed into the following block matrices:

$$\mathbf{L}^\top\mathbf{L} = \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \mathbf{F}_{1,3} \\ & \mathbf{F}_{2,2} & \mathbf{F}_{2,3} \\ \textbf{symmetric} & & \mathbf{F}_{3,3} \end{bmatrix} \tag{5.9}$$

where

$\mathbf{F}_{1,1} = \mathbf{c}_2^\top \mathbf{A}^\top \mathbf{A}\mathbf{c}_2 = (\mathbf{X}_1\mathbf{a})^2 \|\mathbf{c}_2\|^2$,

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

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

$\mathbf{F}_{2,2} = \mathbf{J}_1^\top \mathbf{B}_\alpha^\top \mathbf{B}_\alpha \mathbf{J}_1 = \alpha^2 \|\mathbf{c}_2\|^2 \mathbf{X}_1^\top \mathbf{X}_1 + \mathbf{Z}_1^\top \mathbf{Z}_1$,

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

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

Now we investigate the structure of stiffness matrix. Clearly

$$\mathbf{L}_u = \begin{bmatrix} \mathbf{A}_u\mathbf{c}_2 & \mathbf{B}_\alpha(\mathbf{J}_1)_u & (\mathbf{A}_\alpha)_u\mathbf{J}_2 \end{bmatrix}, \ \mathbf{L}_v = \begin{bmatrix} \mathbf{A}(\mathbf{c}_2)_v & (\mathbf{B}_\alpha)_v\mathbf{J}_1 & \mathbf{A}_\alpha(\mathbf{J}_2)_v \end{bmatrix}.$$

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

$$(\mathbf{H}_4)_u = \begin{bmatrix} \mathbf{0} & \mathbf{C}(\mathbf{J}_1)_u & \mathbf{0} \end{bmatrix}, \ (\mathbf{H}_4)_v = \mathbf{0}.$$

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

$$\begin{aligned} \mathbf{K}_1 &= \frac{1}{3}\iint(\alpha_{1,1}\mathbf{L}_u^\top\mathbf{L}_u + \alpha_{2,2}\mathbf{L}_v^\top\mathbf{L}_v + \beta_{1,1}\mathbf{L}_{uu}^\top\mathbf{L}_{uu} + \\ &\qquad \beta_{1,2}\mathbf{L}_{uv}^\top\mathbf{L}_{uv} + \beta_{2,2}\mathbf{L}_{vv}^\top\mathbf{L}_{vv})\,du\,dv \end{aligned} \tag{5.10}$$

and

$$\begin{aligned} \mathbf{K}_2 &= \frac{2}{3}\iint(\alpha_{1,1}\mathbf{L}_u^\top(\mathbf{H}_4)_u + \alpha_{2,2}\mathbf{L}_v^\top(\mathbf{H}_4)_v + \beta_{1,1}\mathbf{L}_{uu}^\top(\mathbf{H}_4)_{uu} + \\ &\qquad \beta_{1,2}\mathbf{L}_{uv}^\top(\mathbf{H}_4)_{uv} + \beta_{2,2}\mathbf{L}_{vv}^\top(\mathbf{H}_4)_{vv})\,du\,dv \end{aligned} \tag{5.11}$$

In view of (5.6), it is easy to verify

$$\mathbf{K}\mathbf{p} = (\mathbf{K}_1 + \mathbf{K}_2)\mathbf{p} \tag{5.12}$$

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

$$\mathbf{L}_{uv}^\top\mathbf{L}_{uv} = \begin{bmatrix} \mathbf{U}_{1,1} & \mathbf{U}_{1,2} & \mathbf{U}_{1,3} \\ & \mathbf{U}_{2,2} & \mathbf{U}_{2,3} \\ \textbf{symmetric} & & \mathbf{U}_{3,3} \end{bmatrix}, \tag{5.13}$$

where

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

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

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

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

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

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

Next, we discuss $\mathbf{K}_2$. Because $(\mathbf{H}_4)_v = \mathbf{0}$, (5.11) can be simplified as

$$\mathbf{K}_2 = \frac{2}{3} \iint (\alpha_{1,1} \mathbf{L}_u^\top (\mathbf{H}_4)_u + \beta_{1,1} \mathbf{L}_{uu}^\top (\mathbf{H}_4)_{uu})\, du\, dv \tag{5.14}$$

We consider the first derivative entry

$$\mathbf{L}_u^\top (\mathbf{H}_4)_u = \begin{bmatrix} \mathbf{0} & \mathbf{U}'_{1,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}'_{2,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}'_{3,2} & \mathbf{0} \end{bmatrix} \tag{5.15}$$

Using the foregoing notations, it is easy to verify that

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

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

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

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

We have formulated the equations of motion of swung D-NURBS surfaces. One of the challenges in implementing these models is the nonlinear dynamic formulation stemming from the underlying swung NURBS geometry. The NURBS swung surface is inherently nonlinear with respect to its degrees of freedom, even if both NURBS generator curves are reduced to simple B-splines by fixing their weights to unity. As a consequence of the nonlinearity, the mass, damping, and stiffness matrices in the dynamic formulation are all functions of time.

Chapter 6

Triangular D-NURBS

In the previous two chapters, we have formulated tensor product D-NURBS and swung D-NURBS. The main drawback of tensor product NURBS, however, is that the underlying shape (and the parametric domain) is "topologically" rectangular. Consequently, the designer is forced to model multisided irregular shapes using degenerate patches with diminished inter-patch continuity. To compensate, explicit non-linear smoothness constraints must be enforced within the underlying rational representation, thus increasing the complexity of the design task in general.

Triangular B-splines are useful for modeling a broad range of complex objects defined over arbitrary, non-rectangular domains. Using triangular B-splines, designers can benefit from arbitrary parametric domains, non-degeneracy for multi-sided surfaces, and other important features. For instance, designers can construct C^1 continuous surfaces with quadratic triangular B-splines, whereas biquadratic tensor-product B-splines are needed to achieve the same continuity. They can express topologically complex objects without degeneracy. They model smoothness as well as discontinuities by varying the distribution of knots. In particular, they subsume Bernstein-Bezier triangles as a special case with n-fold knots. Moreover, any piecewise polynomial can be represented as a linear combination of triangular B-splines (Seidel, 1992). Thus, triangular B-splines can provide a common representation among various modeling systems for product data exchange and representation conversion.

In this chapter, we propose triangular NURBS, the rational generalization of triangular B-splines, with weights as extra degrees of freedom to increase the power of the modeling scheme. Furthermore, we formulate triangular D-NURBS, a physics-based generalization of triangular NURBS, in order to ameliorate the indirect geometric manipulation. We introduce time, mass, deformation energy into triangular D-NURBS and employ Lagrangian dynamics to derive their motion equations. Like tensor product D-NURBS and swung D-NURBS, triangular D-NURBS are a free-form, rational model that provides a systematic and unified approach for a variety of modeling tasks.

6.1 Geometry

In this section, we formulate triangular NURBS, the rational generalization of triangular B-splines, with weights as extra degrees of freedom. Then, we summarize their analytic and geometric properties.

Generalizing (2.17) by associating a weight $w_{\mathrm{i},\beta}$ with each control point, we define triangular NURBS as the combination of a set of piecewise rational functions:

$$\mathrm{s}(\mathbf{u}) = \frac{\sum_{\mathrm{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathrm{i},\beta} w_{\mathrm{i},\beta} N_{\mathrm{i},\beta}(\mathbf{u})}{\sum_{\mathrm{i}} \sum_{|\beta|=n} w_{\mathrm{i},\beta} N_{\mathrm{i},\beta}(\mathbf{u})}. \tag{6.1}$$

Like non-rational B-splines, the rational nonnegative basis functions of triangular NURBS sum to unity. They inherit many of the properties of non-rational B-splines, such as the convex hull property, local support, affine invariance, and form a common representation for any piecewise polynomial (Dahmen, Micchelli and Seidel, 1992; Seidel, 1993; Fong and Seidel, 1993; Greiner and Seidel, 1994). Moreover, they have some additional properties:

- Triangular NURBS are infinitely smooth in the interior of non-overlapping sub-triangles formed by the knot nets, provided the denominator is nonzero. At the boundary of sub-triangles, they are C^{n-1} continuous if the knots are in general position. The designer can obtain different smoothness conditions by varying the knot arrangement.
- Triangular 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 affect 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.

As in conventional NURBS, fixing the weights to unity reduces the model to triangular B-splines. Using triangular NURBS, the designer can overcome the limitations of tensor product NURBS. Shape design based on triangular NURBS includes the specification of knots and a control polygon, or interpolation/approximation of data points to generate the initial shape. The initial shape is then refined into the final desired shape through interactive adjustment of control points, weights, and knots. The availability of weights as additional degrees of freedom expands the geometric coverage of triangular NURBS. Unlike regular NURBS, however, the special analytic shapes that can be represented precisely by triangular NURBS remains an open question. Because of the irregularity of triangulation vertices and knot sequences, the shape refinement process is *ad hoc* and it can become extremely tedious. Hence, the considerable geometric flexibility of triangular NURBS can make conventional geometric design difficult.

To improve matters, we propose and develop a physics-based triangular NURBS model in the next two sections. The control points and weights of the geometric model of section 2.4 become generalized (physical) coordinates in the dynamic model. We derive the Jacobian and basis function matrices that lead to compact expressions for the velocity and position functions of the surface.

6.2 Kinematic Triangular NURBS

The dynamic triangular NURBS extend the geometric triangular NURBS in (6.1) by explicitly incorporating time and physical behavior. The surface is a function of both the parametric variable $\mathbf{u}$ and time t:

$$\mathbf{s}(\mathbf{u},t) = \frac{\sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta}(t) w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{i}} \sum_{|\beta|=n} w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}. \tag{6.2}$$

To simplify the notation, 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 (6.2) 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 dynamic triangular NURBS is

$$\dot{\mathbf{s}}(\mathbf{u},\mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{6.3}$$

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

$$\mathbf{J} = \left[\ldots, \begin{bmatrix} R_{\mathbf{i},\beta} & 0 & 0 \\ 0 & R_{\mathbf{i},\beta} & 0 \\ 0 & 0 & R_{\mathbf{i},\beta} \end{bmatrix}, \mathbf{w}_{\mathbf{i},\beta}, \ldots \right] \tag{6.4}$$

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:

$$\mathbf{s}(\mathbf{u},\mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{6.5}$$

The proof of (6.5) is the same as that for tensor-product D-NURBS (see Chapter 4).

6.3 Equations of Motion

In the sequel, we will discuss triangular D-NURBS (we consider dynamic triangular B-splines as a special case of triangular D-NURBS with frozen unity weights).

The equations of motion of our dynamic triangular NURBS are derived from Lagrangian dynamics (Gossick, 1967). We express the kinetic energy due to a prescribed mass distribution function $\mu(u, v)$ over the parametric domain of the surface and a dissipation energy due to a damping density function $\gamma(u, v)$. To define an elastic potential energy, we adopt the *thin-plate under tension* energy model (Terzopoulos, 1986; Celniker and Gossard, 1991; Welch and Witkin, 1992)

$$U = \frac{1}{2} \iint \left(\alpha_{1,1} \mathbf{s}_u^2 + \alpha_{2,2} \mathbf{s}_v^2 + \beta_{1,1} \mathbf{s}_{uu}^2 + \beta_{1,2} \mathbf{s}_{uv}^2 + \beta_{2,2} \mathbf{s}_{vv}^2 \right) du\, dv.$$

The subscripts on $\mathbf{s}$ denote parametric partial derivatives. The $\alpha_{i,j}(u, v)$ and $\beta_{i,j}(u, v)$ are elasticity functions which control tension and rigidity, respectively. Other energies are applicable, including the nonquadratic, curvature-based energies in (Terzopoulos and Fleischer, 1988; Moreton and Sequin, 1992). Applying the Lagrangian formulation, we obtain the second-order nonlinear equations of motion

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \tag{6.6}$$

where the mass matrix is

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

the damping matrix is

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

and the stiffness matrix is

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

All are $M \times M$ matrices. The generalized forces on generalized coordinates due to the applied force distribution $\mathbf{f}(u,v,t)$ is

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

Because of the geometric nonlinearity, generalized inertial forces

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

are also associated with the models. The derivation of (6.6) proceeds as for tensor-product D-NURBS (see Chapter 4).

We now briefly discuss the formulation of dynamic triangular B-splines. First, we incorporate time into the geometric triangular B-splines in (2.17):

$$\mathbf{s}(\mathbf{u},t) = \sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u}). \tag{6.7}$$

Then, we define the vector of generalized coordinates (control points only) $\mathbf{p}_{\mathbf{i},\beta}$ as

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

We can express (6.7) as $\mathbf{s}(\mathbf{u},\mathbf{p})$. Thus, the velocity of the dynamic triangular B-splines is

$$\dot{\mathbf{s}}(\mathbf{u},\mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{6.8}$$

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

is a $3 \times M$ matrix, which is expressed as

$$\mathbf{J} = \begin{bmatrix} \ldots, \begin{bmatrix} R_{\mathbf{i},\beta} & 0 & 0 \\ 0 & R_{\mathbf{i},\beta} & 0 \\ 0 & 0 & R_{\mathbf{i},\beta} \end{bmatrix}, \ldots \end{bmatrix}, \tag{6.9}$$

where

$$R_{\mathbf{i},\beta}(\mathbf{u}) = \frac{\partial \mathbf{s}_x}{\partial \mathbf{p}_{\mathbf{i},\beta,x}} = \frac{\partial \mathbf{s}_y}{\partial \mathbf{p}_{\mathbf{i},\beta,y}} = \frac{\partial \mathbf{s}_z}{\partial \mathbf{p}_{\mathbf{i},\beta,z}} = N_{\mathbf{i},\beta}(\mathbf{u}).$$

Obviously, we can express the surface as the product of the Jacobian matrix and the generalized coordinate vector

$$\mathbf{s}(\mathbf{u},\mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{6.10}$$

Chapter 7

Geometric Constraints

We have derived the Lagrangian equations of motion for a variety of D-NURBS. In this chapter, we incorporate geometric constraints into the D-NURBS framework to make D-NURBS more useful for geometric design. Linear constraints allow us to reduce the generalized coordinates vector and matrices to a minimal unconstrained set of generalized coordinates. It is also possible to impose nonlinear constraints on D-NURBS through Lagrange multiplier techniques. Finally, we show that swung D-NURBS discussed in Chapter 5 are tensor-product D-NURBS (see Chapter 4) that have been subjected to a dimensionality-reducing nonlinear constraint.

7.1 Linear Constraints

Linear geometric constraints such as point, curve, and surface normal constraints are often useful (Celniker and Welch, 1992). To incorporate linear geometric constraints into D-

NURBS, we reduce the matrices and vectors in (4.22) to a minimal unconstrained set of generalized coordinates. Linear constraints are generally expressible as follows:

$$\mathbf{C}(\mathbf{p}) = \mathbf{A}\mathbf{p} + \mathbf{b} = \mathbf{0}, \tag{7.1}$$

where $\mathbf{A}$ is a matrix of coefficients. If (7.1) is an underdetermined linear system, we can eliminate variables to express the generalized coordinate vector $\mathbf{p}$ as

$$\mathbf{p} = \mathbf{G}\mathbf{q} + \mathbf{q}_0, \tag{7.2}$$

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 (4.7) and (4.8) with (7.2) 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

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

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

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

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

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

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

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

Applying (4.9), the D-NURBS motion equations with linear constraints are

$$\mathbf{M}_q\ddot{\mathbf{q}}+\mathbf{D}_q\dot{\mathbf{q}}+\mathbf{K}_q\mathbf{q}-\begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top+\begin{bmatrix} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial q_j}\mathbf{p} & \cdots \end{bmatrix}^\top = \mathbf{f}_q-\mathbf{G}^\top(\dot{\mathbf{M}}\dot{\mathbf{p}}+\mathbf{K}\mathbf{q}_0). \tag{7.3}$$

To simplify (7.3) we first show that it reduces to the following

$$\mathbf{G}^\top \dot{\mathbf{M}}\dot{\mathbf{p}} - \begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top = \mathbf{G}^\top \mathbf{I}\dot{\mathbf{p}}. \tag{7.4}$$

As in Chapter 4, $\dot{\mathbf{M}} = \mathbf{I} + \mathbf{I}^\top$. Hence, (7.4) is also expressed as

$$\mathbf{G}^\top \mathbf{I}^\top \dot{\mathbf{p}} = \begin{bmatrix} \cdots & \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial \mathbf{M}}{\partial q_j}\dot{\mathbf{p}} & \cdots \end{bmatrix}^\top. \tag{7.5}$$

Similar to (4.18), the two sides of (7.5) are integrals of two vectors, respectively. Hence, (7.5) holds if corresponding components of the two vectors are equal; i.e., for $j = 1, \ldots, M$,

$$\dot{\mathbf{i}}_j^\top \mathbf{J}\dot{\mathbf{p}} = \frac{1}{2}\dot{\mathbf{p}}^\top \frac{\partial(\mathbf{J}^\top \mathbf{J})}{\partial q_j}\dot{\mathbf{p}}. \tag{7.6}$$

We now prove (7.6). 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, (7.6) follows.

The proof of

$$\begin{bmatrix} \cdots & \frac{1}{2}\mathbf{p}^\top \frac{\partial \mathbf{K}}{\partial q_j}\mathbf{p} & \cdots \end{bmatrix}^\top = \mathbf{0} \tag{7.7}$$

parallels that in Chapter 4, with q_j replacing p_j and $\mathbf{l}_j$ replacing $\mathbf{j}_i$.

We have proven several identities that yield the following equations of motion for D-NURBS with linear constraints:

$$\mathbf{M}_q\ddot{\mathbf{q}} + \mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{g}_q - \mathbf{I}_q\dot{\mathbf{q}}, \tag{7.8}$$

where the generalized forces are

$$\mathbf{f}_q = \iint \mathbf{L}^\top \mathbf{f}(u, v, t)\, du\, dv, \tag{7.9}$$

and where

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

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

Although (7.8) looks more complicated than (4.22), its implementation is surprisingly straightforward in view of the sparseness of $\mathbf{G}$ and the reduced size of $\mathbf{q}$.

7.2 Nonlinear Constraints

It is possible to impose nonlinear geometric (equality) constraints

$$\mathbf{C}(\mathbf{p}) = \mathbf{0}, \tag{7.10}$$

on D-NURBS through Lagrange multiplier techniques (Strang, 1986). 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 (Celniker and Welch, 1992; Welch and Witkin, 1992). The augmented Lagrangian method (Minoux, 1986) combines the Lagrange multipliers with the simpler penalty method (Platt and Barr, 1988).

One of the best known techniques for applying constraints to dynamic models is the Baumgarte stabilization method (Baumgarte, 1972) which solves constrained equations of motion through linear feedback control (see also (Metaxas and Terzopoulos, 1992; Platt, 1992)). We augment (4.22) as follows:

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \mathbf{I}\dot{\mathbf{p}} - \mathbf{C}_{\mathbf{p}}^\top \boldsymbol{\lambda}, \tag{7.11}$$

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 (7.10) 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 (7.10). Taking the second time derivative of (7.10) and rearranging terms yields

$$\mathbf{C}_{\mathbf{p}}\ddot{\mathbf{p}} = -\ddot{\mathbf{C}} - (\mathbf{C}_{\mathbf{p}}\dot{\mathbf{p}})_{\mathbf{p}}\dot{\mathbf{p}} - 2\dot{\mathbf{C}}_{\mathbf{p}}\dot{\mathbf{p}} = \boldsymbol{\sigma}. \tag{7.12}$$

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

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{p}}^{\top} \\ \mathbf{C}_{\mathbf{p}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{p}} - \mathbf{K}\mathbf{p} - \mathbf{I}\dot{\mathbf{p}} + \mathbf{f}_p \\ \boldsymbol{\sigma} - 2a\mathbf{C}_{\mathbf{p}}\dot{\mathbf{p}} - b^2\mathbf{C} \end{bmatrix}. \tag{7.13}$$

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).

7.3 Constraining the Weights

D-NURBS geometry have an interesting idiosyncrasy due to the weights. While the control point components of $\mathbf{p}$ may take arbitrary finite values in $\Re$, this is not the case for the weights $\mathbf{p}_w$. 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. 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 lower weight values tend to flatten the surface in the vicinity of the control points, lowering the deformation energy; thus the weights may tend to decrease. One solution is to use a more complex deformation energy that does not favor flat surfaces as in (Moreton and Sequin, 1992). Alternatively, we can counteract the tendency and also give the designer the option of constraining the weights near certain desired target values w_i^0 by including in the surface energy the penalty term

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

in which c controls the tightness of the constraint.

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 $\mathbf{p}_w^0$.

7.4 Swung D-NURBS as Constrained Tensor Product D-NURBS

It is known that a geometric NURBS swung surface is a NURBS surface (Piegl, 1991). In this section, we show that dynamic NURBS swung surfaces are, analogously, tensor-product D-NURBS surfaces that have been subjected to a dimensionality-reducing nonlinear constraint.

A D-NURBS surface generalizes the geometric NURBS surface:

$$\mathbf{s}(u,v,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{q}_{i,j}(t)w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}. \tag{7.14}$$

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

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

Similar to (4.2) and (4.5), we have

$$\dot{\mathbf{s}}(u,v,\mathbf{q}) = \mathbf{J}\dot{\mathbf{q}}, \qquad \mathbf{s}(u,v,\mathbf{q}) = \mathbf{J}\mathbf{q}. \tag{7.15}$$

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

$$\mathbf{M}_q\ddot{\mathbf{q}} + \mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{g}_q, \tag{7.16}$$

where the mass matrix $\mathbf{M}_q$, the damping matrix $\mathbf{D}_q$, and the stiffness matrix $\mathbf{K}_q$ are all $N\times N$ matrices, $\mathbf{f}_q$ is the generalized force vector acting on $\mathbf{q}$, and $\mathbf{g}_q$ is the generalized inertial force. See (Terzopoulos and Qin, 1994) for the details of the D-NURBS formulation.

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

$$\mathbf{q}_{i,j} \;=\; \begin{bmatrix} \alpha\mathbf{a}_{i,x}\mathbf{b}_{j,x} & \alpha\mathbf{a}_{i,x}\mathbf{b}_{j,y} & \mathbf{a}_{i,z} \end{bmatrix}^{\top}$$

$$w_{i,j} \;=\; w_i^a w_j^b . \tag{7.17}$$

where α, $\mathbf{a}_i$, w_i^a, $\mathbf{b}_j$, and w_j^b, for $i = 0, \ldots, m$ and $j = 0, \ldots, n$, are defined as in Section 5.1. Differentiating (7.17), we obtain

$$\begin{aligned}
\dot{\mathbf{q}}_{i,j} &= \begin{bmatrix} \dot{\alpha}\mathbf{a}_{i,x}\mathbf{b}_{j,x} + \alpha\dot{\mathbf{a}}_{i,x}\mathbf{b}_{j,x} + \alpha\mathbf{a}_{i,x}\dot{\mathbf{b}}_{j,x} \\ \dot{\alpha}\mathbf{a}_{i,x}\mathbf{b}_{j,y} + \alpha\dot{\mathbf{a}}_{i,x}\mathbf{b}_{j,y} + \alpha\mathbf{a}_{i,x}\dot{\mathbf{b}}_{j,y} \\ \dot{\mathbf{a}}_{i,z} \end{bmatrix} \\
\dot{w}_{i,j} &= \dot{w}_i^a w_j^b + w_i^a \dot{w}_j^b
\end{aligned} \tag{7.18}$$

Using the notations in Section 5.1, we can rewrite (7.17) and (7.18) in the matrix form

$$\dot{\mathbf{q}} = \mathbf{G}\dot{\mathbf{p}}, \qquad \mathbf{q} = \mathbf{B}\mathbf{p}, \tag{7.19}$$

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

Substituting (7.19) into (7.16), we arrive at the equations of motion for the dynamic NURBS swung surface (5.8), where the $M \times M$ mass, damping, and stiffness matrices are given by

$$\mathbf{M} = \mathbf{G}^{\mathsf{T}}\mathbf{M}_q\mathbf{G}, \;\; \mathbf{D} = \mathbf{G}^{\mathsf{T}}\mathbf{D}_q\mathbf{G}, \;\; \mathbf{K} = \mathbf{G}^{\mathsf{T}}\mathbf{K}_q\mathbf{B}.$$

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

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

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

Chapter 8

Finite Element Implementation

The evolution of the D-NURBS generalized coordinates is determined by second-order nonlinear differential equations with time-varying mass, damping, and stiffness matrices. We cannot obtain an analytical solution in general. However, an efficient numerical implementation of D-NURBS can be obtained through the use of finite-element analysis and standard numerical techniques. Each D-NURBS patch (span for D-NURBS curves) is treated as a finite element. We compute the individual D-NURBS element matrices using Gaussian quadrature. The discrete equations of motion are simulated through time integration using forward and backward Euler methods, and the D-NURBS state is evolved by using parallelized conjugate gradient algorithms.

8.1 Finite Element Method

The finite element method is used to approximate the continuum physics with a finite number of smaller parts called elements. The behavior of the continuum object is then determined by a finite number of parameters. With the advent of digital computers, discrete problems can be readily solved even if the number of elements is very large. Finite element methods can be characterized in terms of

- dimensionality, *e.g.* curve, surface, and solid,
- form, *e.g.* 2D triangular and quadrilateral elements, 3D tetrahedron, hexahedron, and pentahedron elements,
- type of basis functions, and
- the order of continuity across adjacent elements

A typical element contains a number of nodal points which often consist of the positional and derivative information. The continuum model can be described as a linear combination of nodal points and shape basis functions. Polynomials are frequently used as shape functions. If only coordinates are given, Lagrange polynomials are applicable for shape functions. Objects expressed as a set of Lagrange elements are often C^0 across the adjacent elements. If derivative information is provided, the Hermite interpolation is used. the Hermite type supports C^1 because cross-boundary derivatives are explicitly incorporated into the nodal variables. To achieve high-order inter-element continuity, however, the order of polynomial may be much higher. For example, the quintic Hermite triangular element with 18 nodal variables is only C^1. In mechanical engineering and other engineering applications, precise material properties are vital, and hence interpolation schemes such as Lagrange and/or Hermite polynomials are very well suited to this requirement.

Commonly used finite elements are often formulated with piecewise polynomial basis functions that are non-zero only over one element domain. To achieve the necessary continuity requirements across adjacent elements, nodal points are either constrained or used to explicitly represent derivative information. In contrast, when B-spline basis functions are employed in a finite element formulation, no extra continuity constraints are needed because B-spline basis functions intrinsically inherit high-order continuity.

The finite element (Rayleigh-Ritz) method can also be defined in the form of an approximation of functional minimization. An approximated optimum is constructed in a finite dimensional function space, which can be expressed as a weighted linear combination of basis functions. Individual trial functions can be defined over one element domain, and they can be used to satisfy continuity constraints when stitched together, or basis functions may span several consecutive element domains with intrinsic continuity.

The finite element method has been used to represent continuous deformable models. In (Celniker and Gossard, 1991), triangular elements with 12 degrees of freedom for C^1 continuity are used. Human modeling and animation have proven to be difficult because the human body is a collection of complex rigid and non-rigid components. Quintic triangular finite elements (McInerney and Terzopoulos, 1993) have been used to represent a C^1 continuous surface simulating the deformable behavior of human heart.

A continuum object can be approximated using either the finite difference or the finite element method. The common feature of the two methods is that the approximation is based on displacements at a finite number of nodal points. The disadvantage of the finite difference method is that no shape and continuity information is provided explicitly between nodal points. Note that, derivatives can be approximated at nodal points. In contrast, the finite element method provides a continuous approximation and an analytic characterization over the whole element domain can be derived. In addition, the finite element method generally requires fewer number of nodals points than the finite difference technique to achieve the same approximation accuracy. To model complex shapes, however, both finite difference and finite element methods need a large amount of computation.

8.2 D-NURBS Finite Element

Standard finite element codes assemble individual element matrices into the global matrices that appear in the discrete equations of motion (Zienkiewicz, 1977; Kardestuncer, 1987). 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.

8.2.1 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. 8.1 illustrates a typical finite element of a cubic triangular B-spline surface, along with its local degrees of freedom. Note that, the degrees of freedom of this finite element consist of all control points whose basis functions are non-zero over the current triangle in the parametric domain. Because of

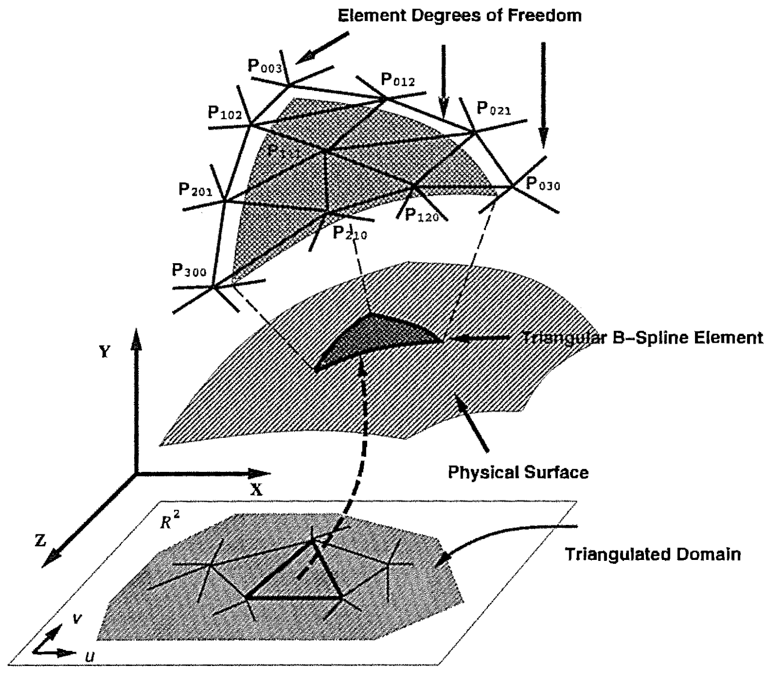


Figure 8.1: One finite element and its degrees of freedom of a triangular B-Spline surface.

the irregular knot distribution of triangular B-splines, we do not display all the degrees of freedom for this finite element; only 10 indexed control points are shown in Fig. 8.1.

We define an element data structure which contains the geometric specification of the D-NURBS element along with its physical properties, as is illustrated in Fig. 8.2. 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 **p**. Note that neighboring elements will share some generalized coordinates. The shared variables will have multiple pointers impinging on them (see Fig. 8.2).

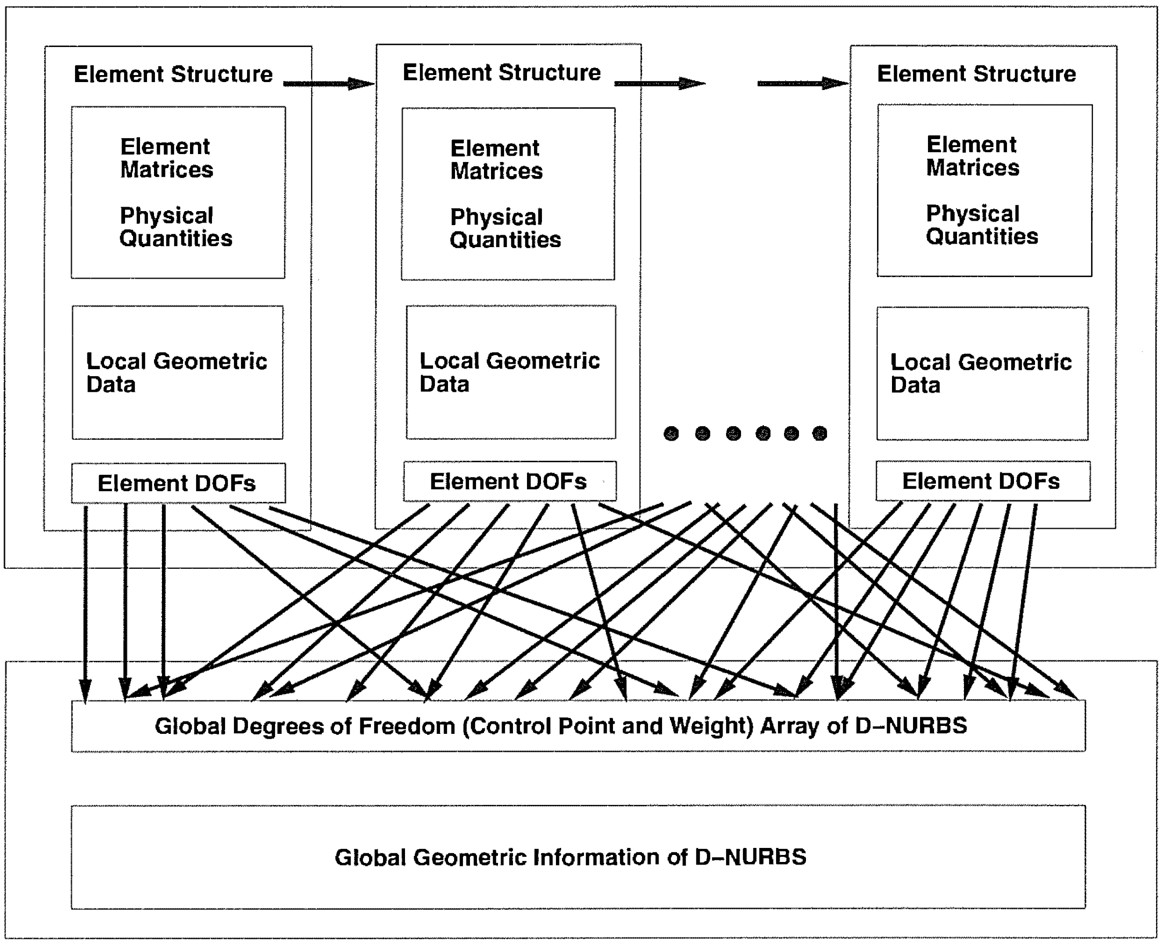


Figure 8.2: Element data structure of D-NURBS.

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.

8.2.2 Calculation of Element Matrices

We evaluate the integral expressions for the matrices (4.11), (4.13), and (4.15) numerically using Gaussian quadrature (Press et al., 1986). 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

$$k_{ij} = \int_{u_0}^{u_1} \int_{v_0}^{v_1} f_{ij}(u,v)\, du\, dv, \tag{8.1}$$

where, according to (4.15),

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

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 ((Press et al., 1986))

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

We apply the de Boor algorithm (de Boor, 1972) to evaluate $f_{ij}(u_g, v_h)$.[1]

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.

For triangular D-NURBS elements, we apply a different formula to compute the integral expressions for the stiffness matrices due to the irregularity of parametric domain. Assuming the parametric domain of the element is $\Delta(\mathbf{r}, \mathbf{s}, \mathbf{t})$, the expression for entry k_{ij} of the stiffness matrix takes the integral form

$$k_{ij} = \iint_{\Delta(\mathbf{r},\mathbf{s},\mathbf{t})} f_{ij}(u, v)\, du\, dv.$$

Given integers N_g, we can find Gauss weights a_g, and abscissas u_g and v_g in the two parametric directions such that k_{ij} can be approximated by

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

We apply the recursive algorithm of multivariate simplex B-Splines (Micchelli, 1979) to evaluate $f_{ij}(u_g, v_g)$. In our implementation we choose N_g to be 7 for quadratic and cubic

[1]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: $k_{ij} \approx \sum_{g=1}^{N_g} a_g f_{ij}(u_g)$.

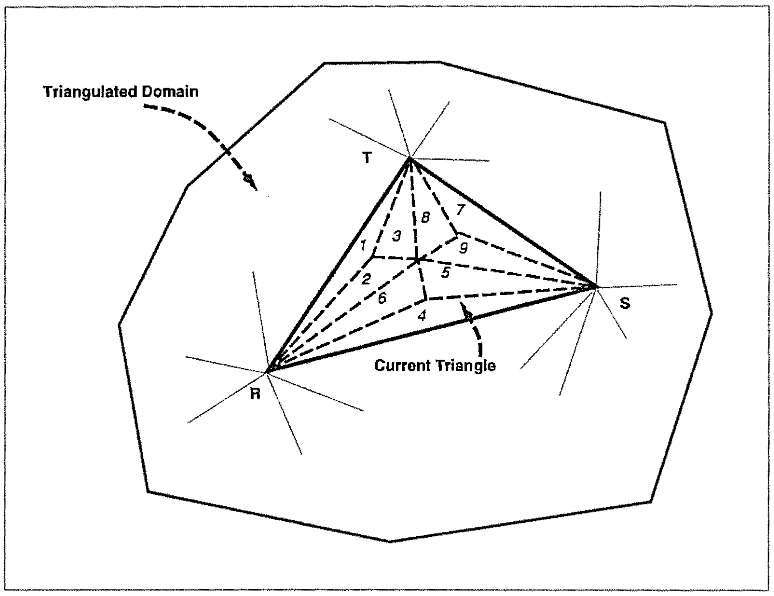


Figure 8.3: Nine subtriangles for numerical quadrature.

triangular B-splines. Note that because of the irregular knot distribution, many f_{ij}'s are zero over the sub-region of $\Delta(\mathbf{r},\mathbf{s},\mathbf{t})$. We can further subdivide the $\Delta(\mathbf{r},\mathbf{s},\mathbf{t})$ in order to decrease the numerical quadrature error. We subdivide each triangular domain into 9 subtriangles (Fig. 8.3) and observe that matrices computed in this way lead to stable, convergent solutions. Alternatively, a more precise though more expensive approach is to convert a triangular B-spline into piecewise Bezier surfaces defined on a *finer* triangulation due to the extra knot lines (see Fig. 8.4 for the quadratic case). In general, about $10-100$ finer triangles are obtained for one triangle in the parametric domain for cubics and quadratics.

8.3 Discrete Dynamics Equations

In order to integrate the D-NURBS ordinary differential equations of motion 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

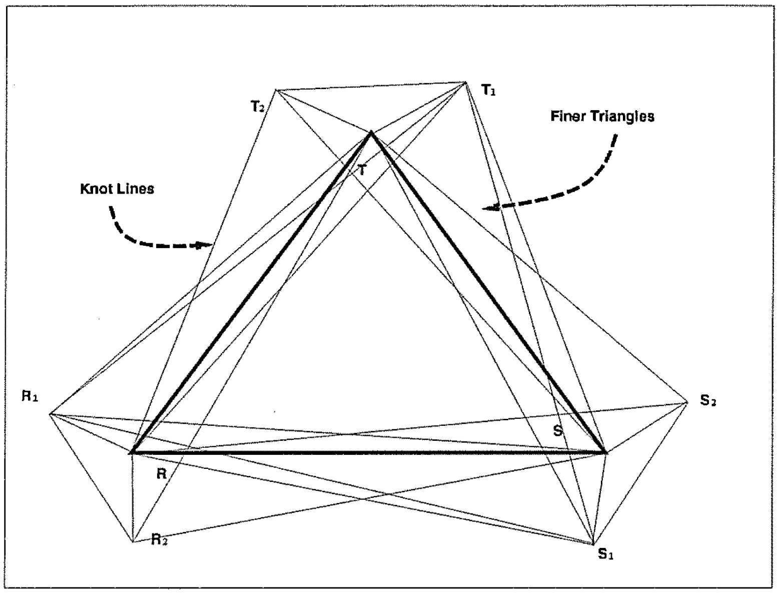


Figure 8.4: Finer triangulation due to intersection of knot lines.

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.

We integrate the differential equations 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, (4.22) 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

$$\left(4\mathbf{M} + 2\Delta t\mathbf{D} + 4\Delta t^2\mathbf{K}\right)\mathbf{p}^{(t+\Delta t)} = 4\Delta t^2\mathbf{f}_p + 8\mathbf{M}\mathbf{p}^{(t)} - (3\mathbf{M} - 2\Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)} + \iint \mu\mathbf{J}^\top\mathbf{s}\,du\,dv, \tag{8.2}$$

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

$$\mathbf{M}^{(t+\Delta t)} = \mathbf{M}^{(t)} + \Delta t\dot{\mathbf{M}}^{(t)} = 2\mathbf{M}^{(t)} - \mathbf{M}^{(t-\Delta t)} \tag{8.3}$$

and likewise for the other matrices and vectors in (8.2). The simpler, constant extrapolations $\mathbf{M}^{(t+\Delta t)} = \mathbf{M}^{(t)}$, etc., ((Kardestuncer, 1987) Section 8.6) also work satisfactorily.

In the interest of efficiency, we do not factorize the matrix expression on the left hand side of (8.2) in order to solve for $\mathbf{p}^{(t+\Delta t)}$. Instead, we employ the conjugate gradient method to obtain an iterative solution (Press et al., 1986; Strang, 1986). 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.

For the D-NURBS curve, we simply substitute $\mathbf{c}$ with $\mathbf{s}$ in (8.2) 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 (8.2), the discrete version of (7.8) is

$$(4\mathbf{M}_q + 2\Delta t\mathbf{D}_q + 4\Delta t^2\mathbf{K}_q)\mathbf{q}^{(t+\Delta t)} =$$
$$4\Delta t^2(\mathbf{f}_q + \mathbf{g}_q) + 8\mathbf{M}_q\mathbf{q}^{(t)} - (3\mathbf{M}_q - 2\Delta t\mathbf{D}_q)\mathbf{q}^{(t-\Delta t)} - \mathbf{G}^\top\mathbf{M}\mathbf{q}_0 + \iint \mu\mathbf{L}^\top\mathbf{s}\,du\,dv. \quad (8.4)$$

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 $\mathbf{Gq}$ and the multiplication of $\mathbf{G}$ with several vectors in (8.4).

For nonlinear constraints, at each time step we can apply the conjugate gradient algorithm to solve (7.13) 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 $\dot{\mathbf{p}}^{(t+\Delta t)} = \dot{\mathbf{p}}^{(t)} + \Delta t\ \ddot{\mathbf{p}}^{(t)}$; $\mathbf{p}^{(t+\Delta t)} = \mathbf{p}^{(t)} + \Delta t\ \dot{\mathbf{p}}^{(t+\Delta t)}$).

8.4 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 (8.2) and

(8.4), 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 (4.22) and (7.8) 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 (4.22) is appropriate if the models must respond with realistic dynamics. Typical applications include animation of deformable objects, object sculpting in virtual environment with force feedback, and testing material properties for manufacturing. 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 (4.22) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish and no oscillation occurs. This economizes on storage and makes the algorithm more efficient. With zero mass density, (4.22) reduces to

$$\mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p, \tag{8.5}$$

while (7.8) reduces to

$$\mathbf{D}_q\dot{\mathbf{q}} + \mathbf{K}_q\mathbf{q} = \mathbf{f}_q + \mathbf{h}_q. \tag{8.6}$$

Discretizing the derivatives of $\mathbf{p}$ and $\mathbf{q}$ in (8.5) and (8.6) with backward differences, we obtain the integration formulas

$$(\mathbf{D} + \Delta t\mathbf{K})\,\mathbf{p}^{(t+\Delta t)} = \Delta t\mathbf{f}_p + \mathbf{D}\mathbf{p}^{(t)} \tag{8.7}$$

and

$$(\mathbf{D}_q + \Delta t \mathbf{K}_q)\, \mathbf{q}^{(t+\Delta t)} = \Delta t(\mathbf{f}_q + \mathbf{h}_q) + \mathbf{D}_q \mathbf{q}^{(t)} \tag{8.8}$$

respectively.

8.5 Explicit Time Integration

Note that when physical parameter values are chosen such that the equations (4.22) are not stiff, it is much cheaper to employ an explicit time integration method using forward differences. In this case, we can 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 (4.22) as

$$\begin{aligned}\mathbf{M}\mathbf{p}^{(t+\Delta t)} &= \Delta t^2(\mathbf{f}_p - \mathbf{K}\mathbf{p}^{(t)}) + 2\mathbf{M}\mathbf{p}^{(t)} - \Delta t \mathbf{D}\mathbf{p}^{(t)} \\ &\quad + \Delta t \mathbf{D}\mathbf{p}^{(t-\Delta t)} - \iint \mu \mathbf{J}^\top \mathbf{s}^{(t-\Delta t)}\, du\, dv.\end{aligned} \tag{8.9}$$

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, (7.8) is discretized as

$$\begin{aligned}\mathbf{M}_q \mathbf{q}^{(t+\Delta t)} &= \Delta t^2(\mathbf{f}_q + \mathbf{g}_q - \mathbf{K}_q \mathbf{q}^{(t)}) + 2\mathbf{M}_q \mathbf{q}^{(t)} - \Delta t \mathbf{D}_q \mathbf{q}^{(t)} + \\ &\quad \Delta t \mathbf{D}_q \mathbf{q}^{(t-\Delta t)} + \mathbf{G}^\top \mathbf{M} \mathbf{q}_0^{(t-\Delta t)} - \iint \mu \mathbf{L}^\top \mathbf{s}^{(t-\Delta t)}\, du\, dv.\end{aligned} \tag{8.10}$$

For the D-NURBS curve, we substitute $\mathbf{c}$ with $\mathbf{s}$ in (8.9) and (8.10).

The discretized forms of the simplified first order equations of motion (8.5) and (8.6) are

$$\mathbf{D}\mathbf{p}^{(t+\Delta t)} = \Delta t(\mathbf{f}_p - \mathbf{K}\mathbf{p}^{(t)}) + \mathbf{D}\mathbf{p}^{(t)} \tag{8.11}$$

and

$$\mathbf{D}_q\mathbf{q}^{(t+\Delta t)} = \Delta t(\mathbf{f}_q + \mathbf{h}_g - \mathbf{K}_q\mathbf{q}^{(t)}) + \mathbf{D}_q\mathbf{q}^{(t)}. \tag{8.12}$$

Note that the explicit method requires values for the matrices only at time t, hence (8.3) is not needed.

Chapter 9

Modeling Environment and Applications

This chapter describes our D-NURBS modeling environment and presents a wide range of applications including shape blending, scattered data fitting, surface trimming, cross-sectional shape design, shape metamorphosis, and free-form deformation. Thus we show that D-NURBS provide a systematic and unified approach for a variety of CAD and graphics modeling problems such as constraint-based optimization, variational parametric design, automatic weight selection, shape approximation, and user interaction. They also support direct manipulation and interactive sculpting using force-based manipulation tools through the imposition of forces, the specification of geometric constraints, and the adjustment of physical parameters such as mass, damping, and elasticity.

9.1 Interactive Modeling Environment

We have developed a prototype modeling environment based on the D-NURBS model. The system is written in C and it is packaged as an interactive Iris Explorer module on Silicon Graphics workstations. Our parallelized iterative numerical algorithm takes advantage of a 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. They may be combined with existing Explorer modules for data input and surface visualization.

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, along with 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.[1]

9.2 Physics-Based Design Tools

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

[1]At present, our software assumes uniform mass, damping, and elasticity densities over the parametric domain, except across trimming boundaries (see Section 9.3.1). 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.

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.

9.2.1 Sculpting 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. (Terzopoulos and Fleischer, 1988; Celniker and Welch, 1992; Szeliski and Tonnesen, 1992).

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

$$\mathbf{f}(u, v, t) = \iint k(\mathbf{d}_0 - \mathbf{s}(u, v, t))\delta(u - u_0, v - v_0)\, du\, dv, \tag{9.1}$$

where is the δ is the unit delta function. Equation (9.1) 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.

9.2.2 Constraints

In practical applications, design requirements may be posed as a set of physical parameters or as geometric constraints. Nonlinear constraints can be enforced through Lagrange multiplier techniques (Minoux, 1986; Strang, 1986; Platt, 1992). 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 (Minoux, 1986) combines the Lagrange multipliers with the simpler penalty method (Platt and Barr, 1988). The Baumgarte stabilization method (Baumgarte, 1972) solves constrained equations of motion through linear feedback control (see also (Metaxas and Terzopoulos, 1992)).

Linear geometric constraints can be easily incorporated into D-NURBS by reducing the matrices and vectors to a minimal unconstrained set of generalized coordinates. 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. For swung D-NURBS, the two generator curves must be embedded in $x - z$ and $x - y$ planes, respectively. For triangular D-NURBS, by confining all associated weights to be unity, we obtain dynamic triangular B-splines. We can also arrive at the *continuous net* (Fong and Seidel, 1993) (which is a special case of general triangular B-splines) by constraining respective control points along common boundaries of two adjacent triangles in parametric triangulation (Fig. 9.1).

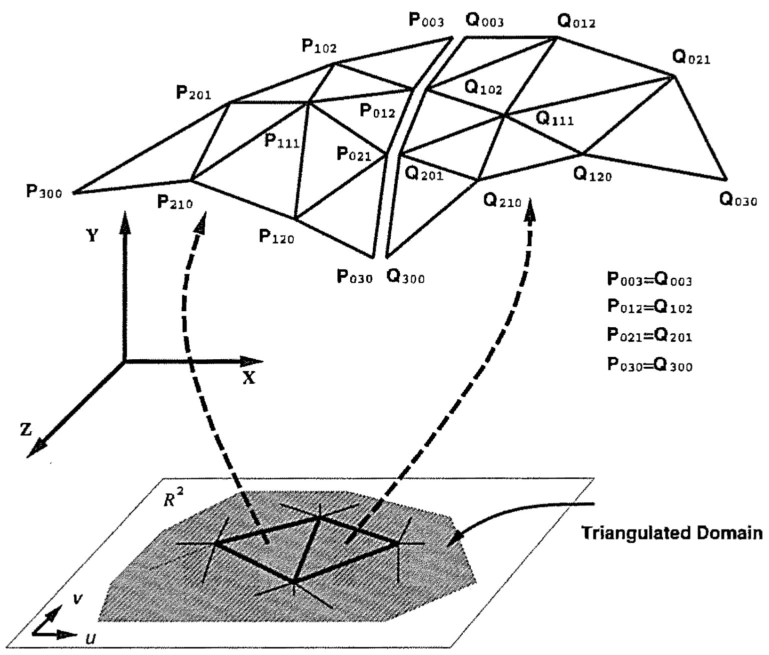


Figure 9.1: A constrained triangular B-Spline configuration: *continuous net.*

9.3 Tensor Product D-NURBS Applications

9.3.1 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 8.2.2) 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 pruning 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, only abscissas that sample locations with zero physical parameters make zero 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 (Press et al., 1986). Fortunately, in practice, the D-NURBS model appears tolerant of the reduced integration accuracy in boundary elements.

Fig. 9.2 illustrates the trimming of D-NURBS surfaces using D-NURBS trimming curves in the parametric domain. Fig. 9.2(a) shows the creation of a triangular surface with three linear curves each with 4 control points. Fig. 9.2(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.

9.3.2 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 smoothes 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. 9.3 demonstrates edge rounding using D-NURBS surfaces. In Fig. 9.3(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. 9.3(a2).

Fig. 9.3(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 constraints along the far boundaries of the faces (Fig. 9.3(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.

9.3.3 Optimal Surface Fitting

D-NURBS are applicable to the optimal fitting of regular or scattered data (Schumaker, 1976). 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 (Terzopoulos, 1986; Szeliski and Terzopoulos, 1989). 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 7.1. The optimal approximation problem can be approached in physical terms, by coupling the D-NURBS to the data through Hookean spring forces (9.1). We interpret $\mathbf{d}_0$ in (9.1) 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.[2]

We present three examples of surface fitting using D-NURBS coupled to data points through spring forces. Fig. 9.4(a) shows 19 data points sampled from a hemisphere and their interpolation with a quadratic D-NURBS surface with 49 control points. Fig. 9.4(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. 9.4(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.

9.3.4 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 (Piegl, 1991)—for generating common surfaces such as extruded surfaces, natural quadrics,

[2]Cross-validation (Wahba, 1990) 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 (Terzopoulos, 1986).

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. 9.5 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. 9.5(a)). The generalized cylinder is interactively sculpted into various shapes by applying spring forces on the green and red cubic D-NURBS curves (Fig. 9.5(b-d)). Second, Fig. 9.6 shows a torus with 49 control points generated by swinging the green curve over the red curve (Fig. 9.6(a)). Both generators are closed cubic D-NURBS curves with 7 control points. In Fig. 9.6(b-d), the torus is deformed interactively by applying a spring force. Third, Fig. 9.7 shows a 35 control point "wine glass" shape obtained by sweeping the green generator curve on the red generator curve in Fig. 9.7(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. 9.7(b-d)).

9.3.5 Shape Metamorphosis

Metamorphosis is the blending of one shape into another. Work on 3D shape blending includes (Chen and Parent, 1989; Kaul and Rossignac, 1991). The blending of 2D curves has widespread application in illustration, animation, etc., and simple (e.g., linear) interpolation techniques usually produce unsatisfactory results (Sederberg et al., 1993). Shinagawa and Kunii propose an method (Shinagawa and Kunii, 1991) which interpolates differential properties of the 2D shape using the elastic surfaces of (Terzopoulos et al., 1987; Terzopoulos and Fleischer, 1988). 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 curves 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. 9.8 shows minimal-energy D-NURBS surfaces with 3×6 control points (3 control points along u) interpolating between two closed elliptical curves. Fig. 9.8(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 (Sederberg et al., 1993). 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. 9.8(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. 9.8(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. 9.9 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.

9.3.6 Free-Form Deformation

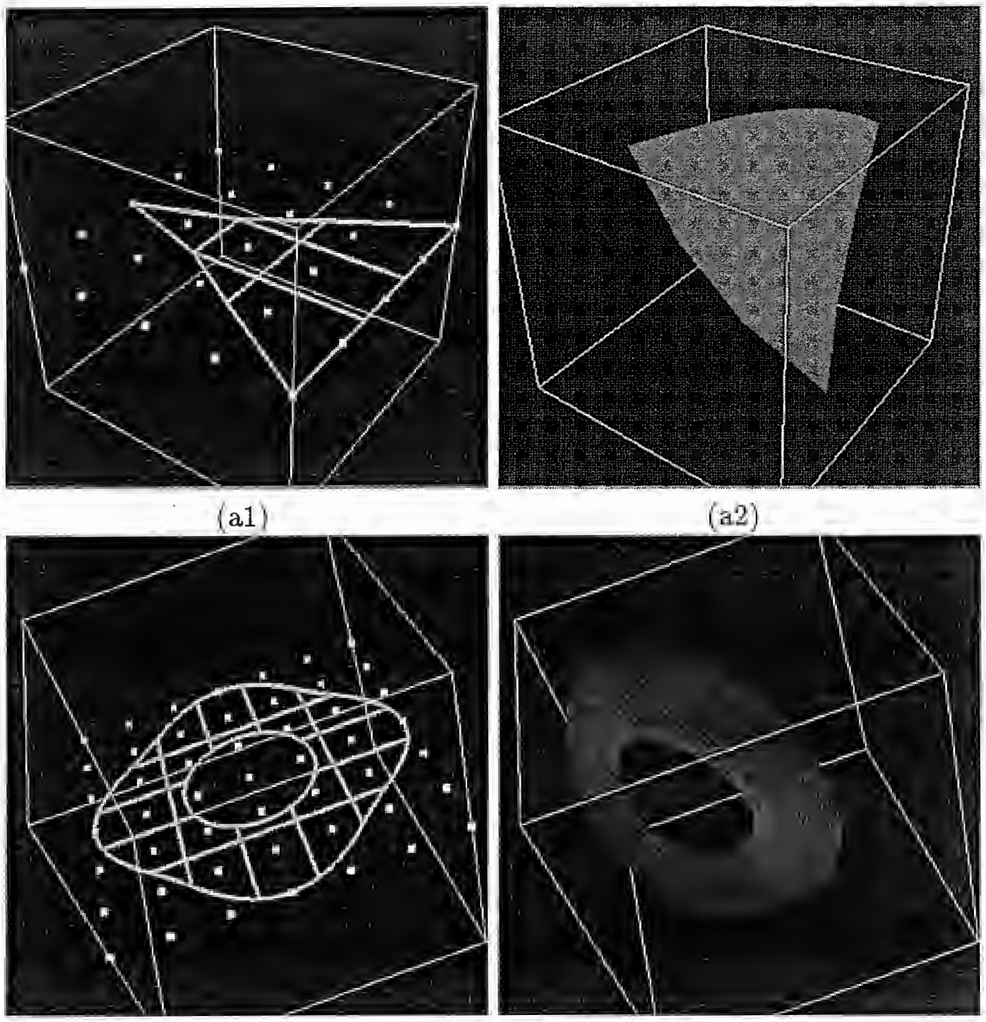
Bezier introduced the idea of globally deforming a shape through a $\Re^n \to \Re^n$ mapping implemented as a free-form (tensor product) spline. The shape is embedded in the parametric domain of the spline and deformed by manipulating the spline's control points. Sederberg and Parry (Sederberg and Parry, 1986) 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 (Chadwick, Haumann and Parent, 1989), 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).

9.4 Swung D-NURBS Applications

9.4.1 Rounding and Blending

The dynamic NURBS swung surface can produce a smooth fillet by minimizing its internal deformation energy subject to position and normal constraints. The dynamic simulation automatically produces the desired final shape as it achieves equilibrium.



(b1) (b2)

Figure 9.2: 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.

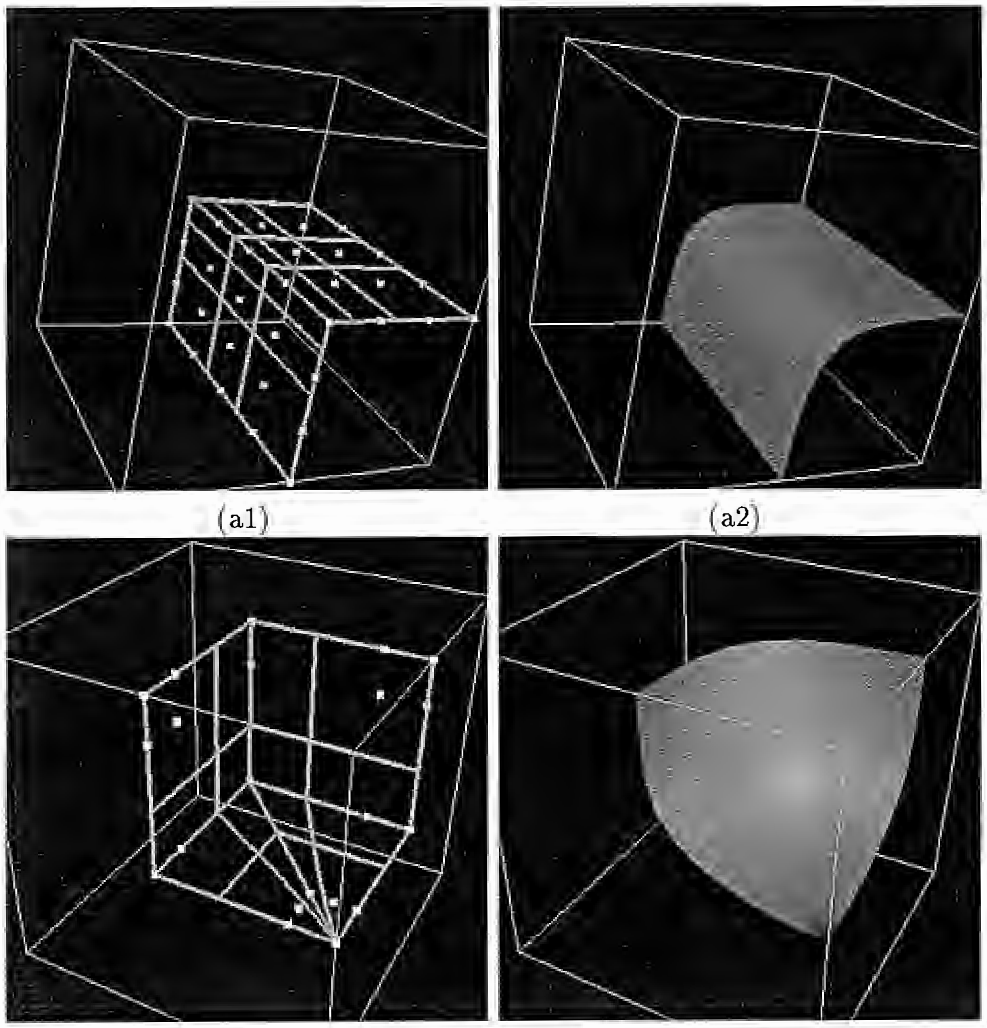

(a1) (a2) (b1) (b2)

Figure 9.3: 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.

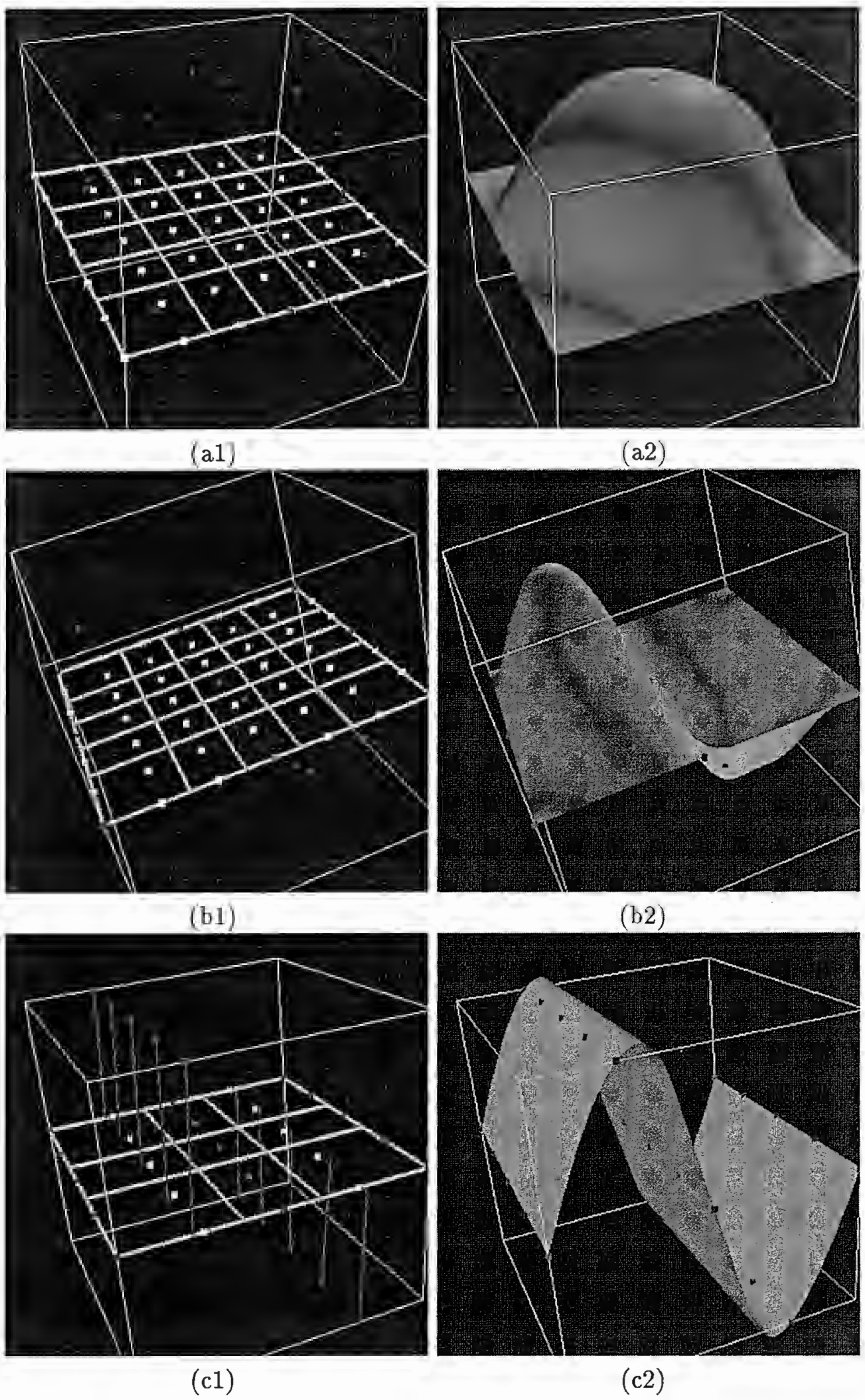


Figure 9.4: Optimal surface fitting: D-NURBS surfaces fit to sampled data from (a) a hemisphere, (b) a convex/concave surface, (c) a sinusoidal surface. (a–c1) D-NURBS patch outline with control points (white) and data points (red) shown. (a–c2) D-NURBS surface at equilibrium fitted to scattered data points. Red line segments in (c2) represent springs with fixed attachment points on surface.

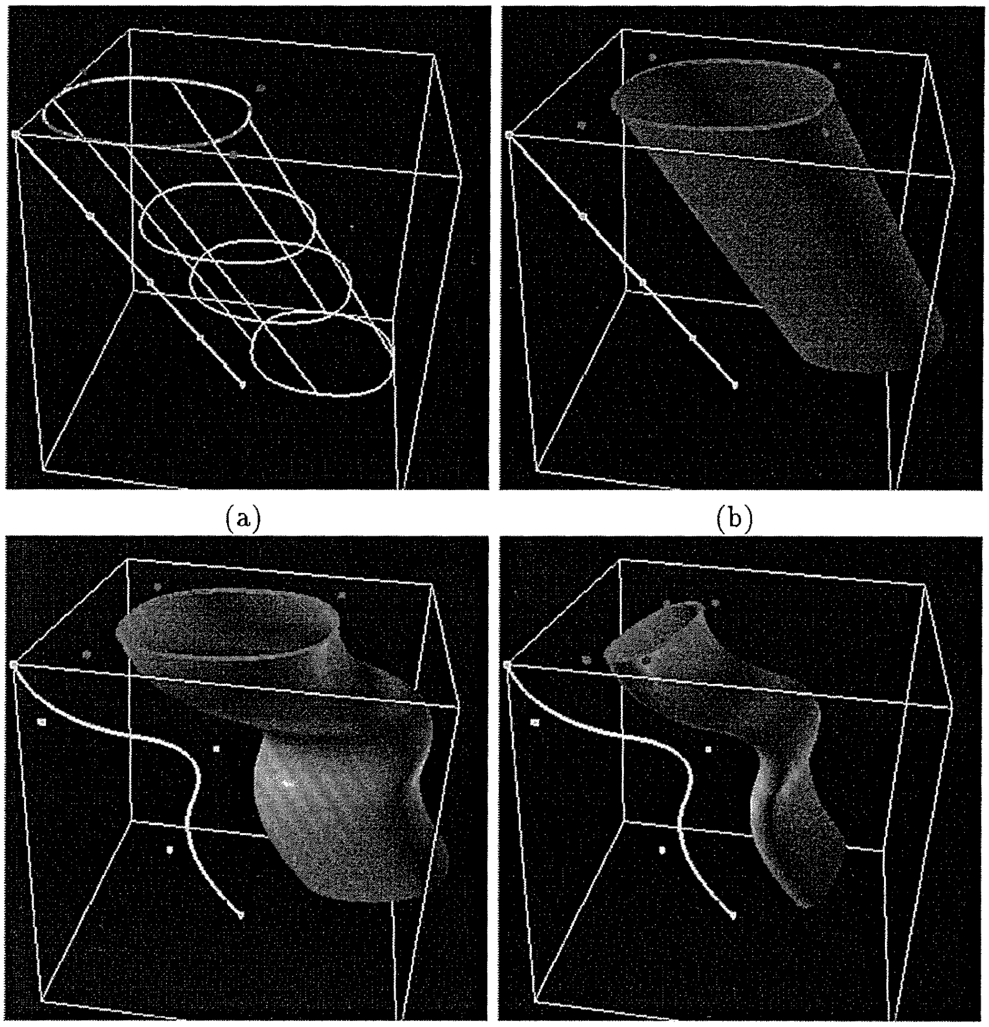

(a) (b) (c) (d)

Figure 9.5: 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.

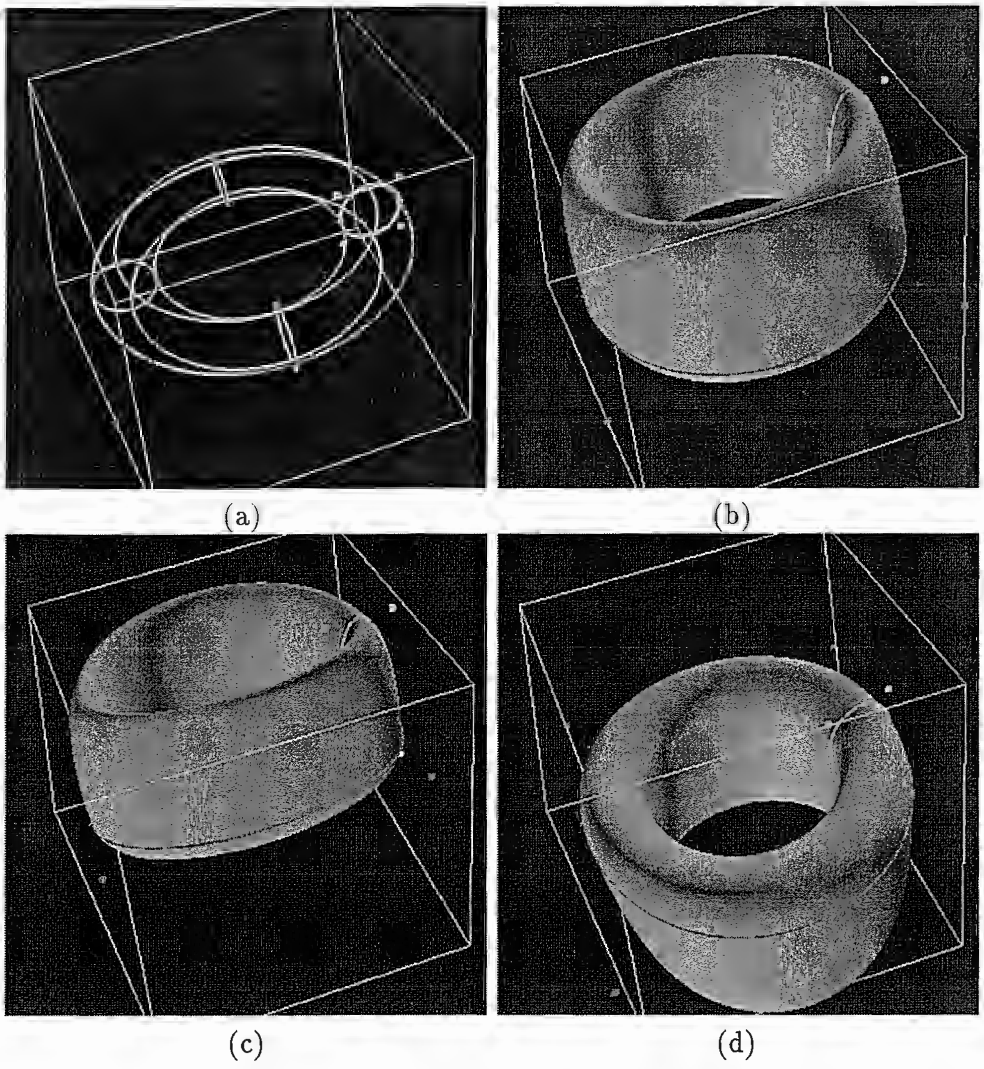
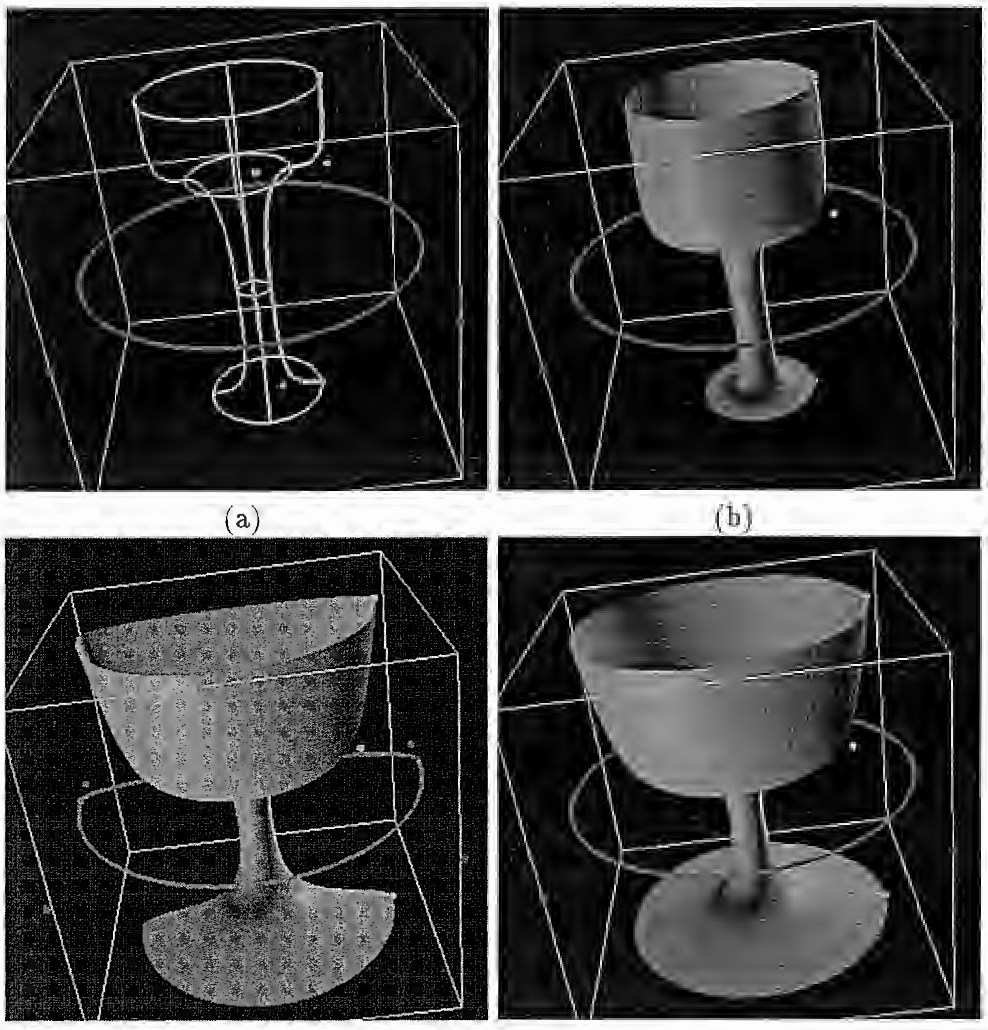

(a) (b) (c) (d)

Figure 9.6: 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 9.7: 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.

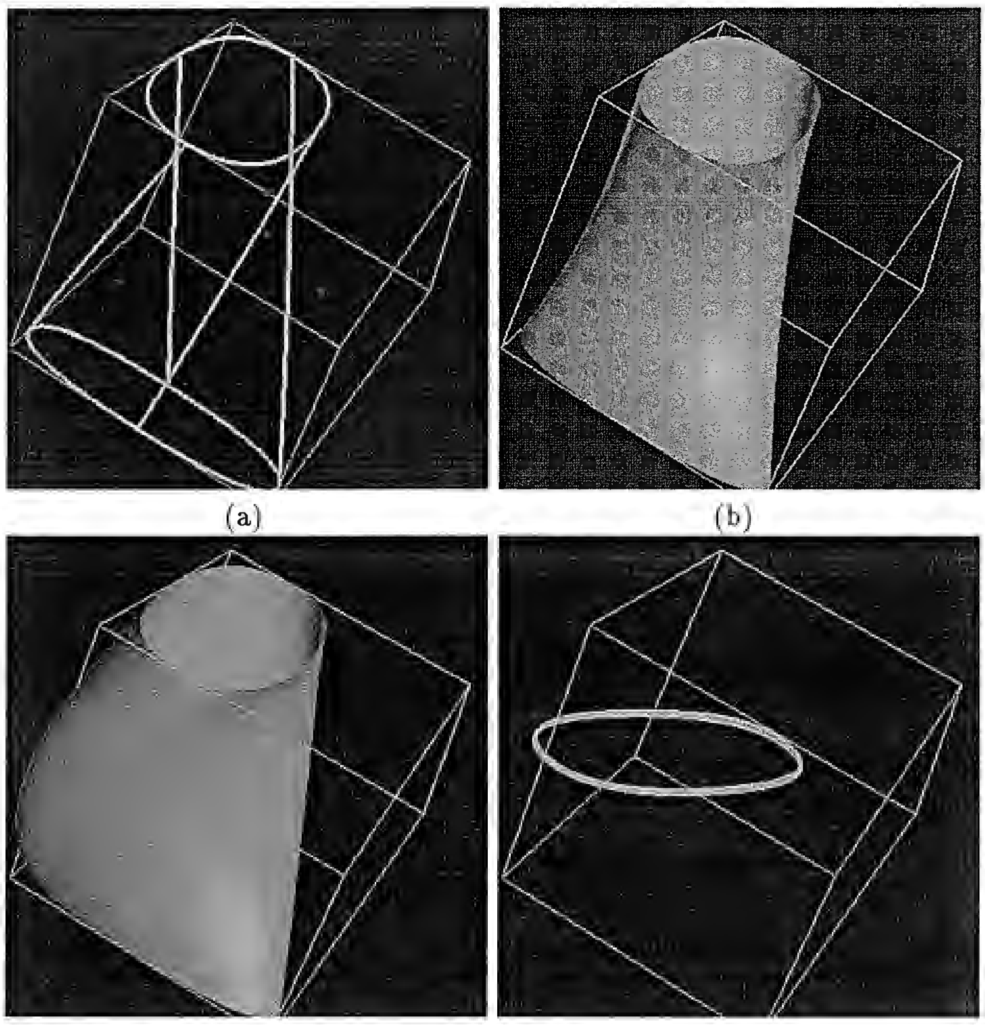

(a) (b) (c) (d)

Figure 9.8: 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).

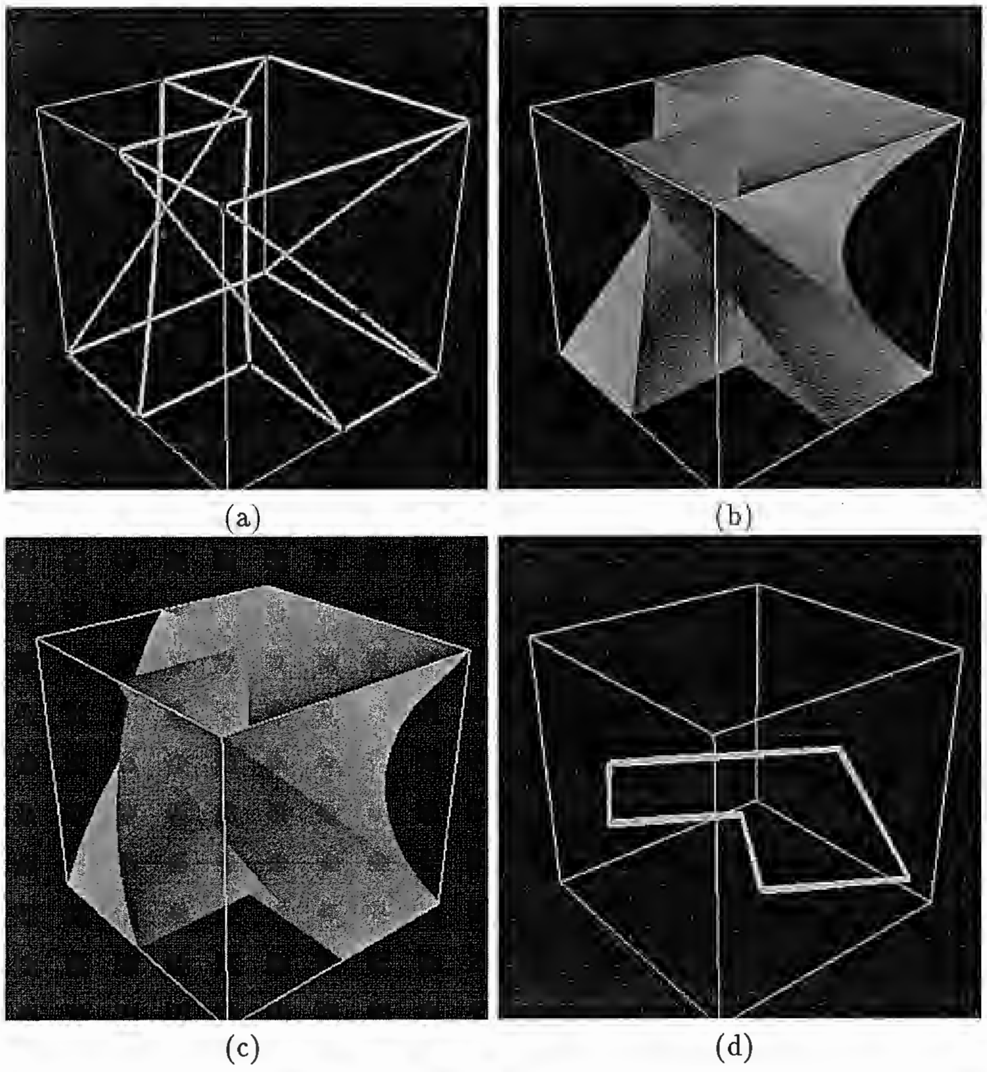


Figure 9.9: Metamorphosis between two planar polygonal curves using D-NURBS interpolating surface. (a) Control points and patch outlines of cylindrical surface terminated by the two planar curves. (b) Linear interpolating surface. (c) Constrained nonlinear interpolating surface combines rigid rotation with nonrigid deformation. (d) An intermediate morphed curve obtained as cross section of surface in (c).

Fig. 9.12 demonstrates the rounding of a polyhedral toroid. The profile
x-z plane is a quadratic NURBS curve with 17 control points. The trajectory
x-y plane is also a quadratic NURBS with 17 control points. Note that, the c
curves can be represented exactly with multiple control points or approximat
a very large weight. If this model were a general NURBS surface, it would hav
points and weights. As a swung surface it has only 34 control points and weig
considered the generalized coordinates of the dynamic model. The wirefram
shapes is shown in Fig. 9.12(a) and Fig. 9.12(b). After initiating the physical simulation, the corners and sharp edges are rounded as the final shape equilibrates into the minimal energy state shown in Fig. 9.12(c-d).

Fig. 9.13 illustrates the rounding of a cubical solid. The profile is a quadratic NURBS with 15 control points, and the trajectory is a linear NURBS with 5 control points. The rounding operation is applied in the vicinity of the middle edge. In Fig. 9.13(a-b) shows the wireframe and shaded objects, respectively. The rounded shape is shown in Fig. 9.13(c-d).

Fig. 9.14 shows a blending example involving a cylindrical pipe. The circular profile is a quadratic curve with 7 control points. The piecewise linear trajectory has 5 control points. The initial right-angle pipe and the final rounded pipe are shown in Fig. 9.14(a-d).

To demonstrate automatic weight variation in the dynamic model in accordance with physical parameters, we have conducted an experiment with the polyhedral toroid shown in Fig. 9.12(a). We fix all 137 degrees of freedom of the toroid, which occupies the cube $-1 \leq x, y, z, \leq 1$, except for one weight associated with the control point $(1.0, 0.0, 1.0)$ which is initially set to 50.0 to form the corner. This weight is permitted to vary subject to varying physical parameters and external force (Fig. 9.10). Initially $\gamma = 100.0$ and the remaining parameters are zero ($\Delta t = 0.248$). Starting the dynamic simulation, we first gradually increase $\alpha_{1,2}$ from zero to 150.0. As shown in Fig. 9.10, the value of the free

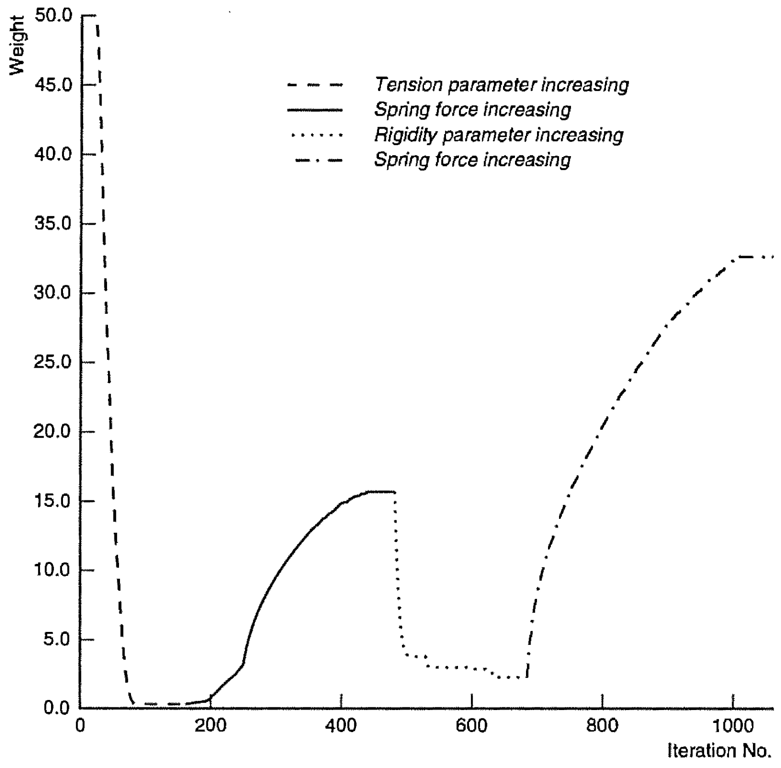


Figure 9.10: Free weight variation subject to varying physical parameters and external force (see text).

weight decreases quickly towards its lower limit 0.3 in under 100 iterations (dashed plot). This reduces the deformation energy by rounding the corner. Next, we attach a spring force from the point $(1.0, 1.0, 1.0)$ to the nearest surface point. As the stiffness constant of this spring is slowly increased from zero to $1,000.0$, the weight increases (solid plot in Fig. 9.10) thus recreating a corner. Next, we vary $\beta_{2,2}$ from zero to 100.0 (dotted plot in Fig. 9.10). The increased rigidity of the surface causes the weight to decrease, again rounding the corner. Finally, we increase the spring stiffness to $10,000.0$ (stippled plot in Fig. 9.10), thus counteracting the increased stiffness and causing the weight to increases to again recreate the corner. This experiment demonstrates that when control points are fixed, a free weight will automatically decrease to decrease the deformation energy by flattening the surface locally, unless external forces counteract this tendency. When control points are free, they will also vary as well to produce an analogous physical effect.

9.4.2 Scattered Data Fitting

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

An important advantage of our models, despite the fact that they are profile surfaces, is that they can be fitted to arbitrarily distributed empirical data that are not aligned

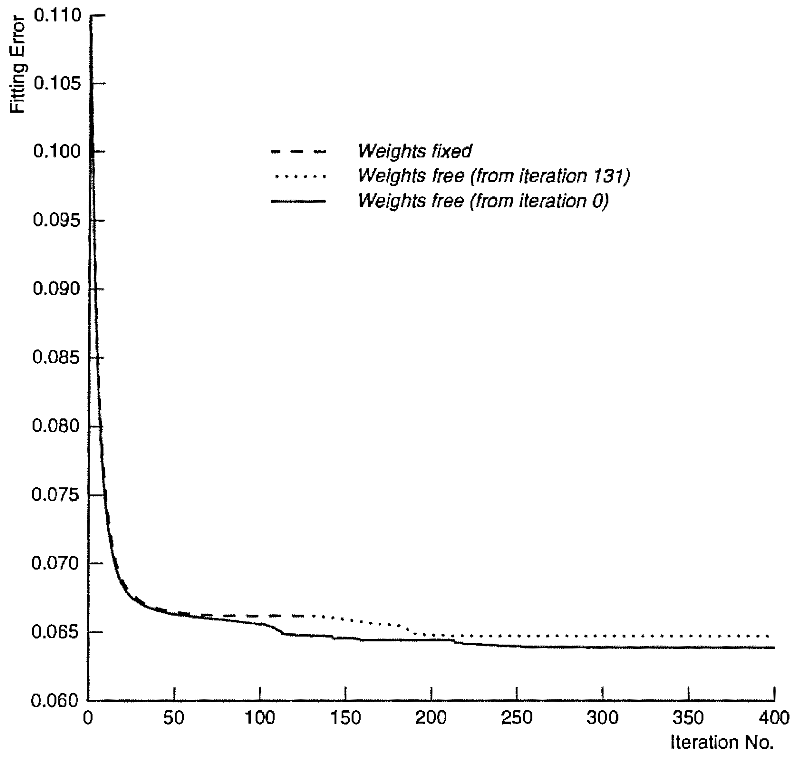


Figure 9.11: Least Squared Fitting Errors For NRCC Pot (see text).

along any particular isoparametric curve pattern. We first use a dynamic swung surface generated by two quadratic profiles with 10 and 7 control points to reconstruct a clay pot which has been densely sampled by a cylindrical laser scanner to produce about 1.2×10^6 data points. The 7 control points and weights of the cross-section profile are constrained so as to permit only surfaces of circular cross section. We randomly selected only 20 data points from which to reconstruct a surface of revolution. Fig. 9.15 shows the sample points, the cylindrical initial condition, and the final fitted shape rendered with the texture map of the object acquired by the scanner. The elastic energy of the surface allows it to interpolate between data points. The physical parameters used in this experiment were mass $\mu = 0.0$, damping $\gamma = 60.0$, bending stiffness parameter $\beta_{1,1} = 14.0$ while all the others are zero, data spring constants $c_i = 900.0$. The surface fitting stabilizes in a few seconds with a time step $\Delta t = 0.3$.

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

Table 9.1: Variation of weights in experiment 1.

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

Table 9.2: Variation of weights in experiment 2.

We report two more experiments with the above surface fitting scenario in order to investigate the least-squares fitting errors under different circumstances. In the first experiment, we initially fix all 10 weights of the profile curve. An optimal fit is achieved after 81 iterations. The decreasing error is plotted by the dashed plot in Fig. 9.11. Then we free the weights at iteration 131 and the model is able to further reduce the fitting error because it now has more degrees of freedom at its disposal. The dotted plot in the figure indicates the improved fit. In the second experiment, the weights are set free from the start of the dynamic simulation. After 295 iterations, the optimal fitting is recorded by the solid plot in Fig. 9.11. The weight values in the two experiments are given in Table 9.1 and 9.2, respectively. The experiments indicate that, at least in this situation, a slightly better surface fit is obtained when the model is permitted to use all of the available degrees of freedom from the start of the fitting process. Obviously, the final surface in the first experiment had attained a local optimum.

Next, we use the same surface model to approximate 10 data points sampled from a vase. The wireframe and textured images of the dynamic swung surface are illustrated in Fig. 9.16. Fig. 9.17(a-d) shows the final reconstructed shapes from four other fitting experiments using synthetic data to recover another pot, a vase, a bottle, and a wine glass. The number of randomly sampled data are 10, 13, 14, and 17, respectively.

9.4.3 Interactive Sculpting

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

The physics-based modeling approach is ideal for interactive sculpting of surfaces. It provides direct manipulation of the dynamic surface to refine the shape of the surface through the application of interactive sculpting tools in the form of forces. Fig. 1.1(a) illustrates the results of four interactive sculpting sessions using spring forces. A sphere was generated using two quadratic curves with 4 and 7 control points and was sculpted into the ovoid shown in Fig. 1.1(a). A torus whose two profile curves are quadratic with 7 and 7 control points, respectively, has been deformed into the shape in Fig. 1.1(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. 1.1(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. 1.1(c).

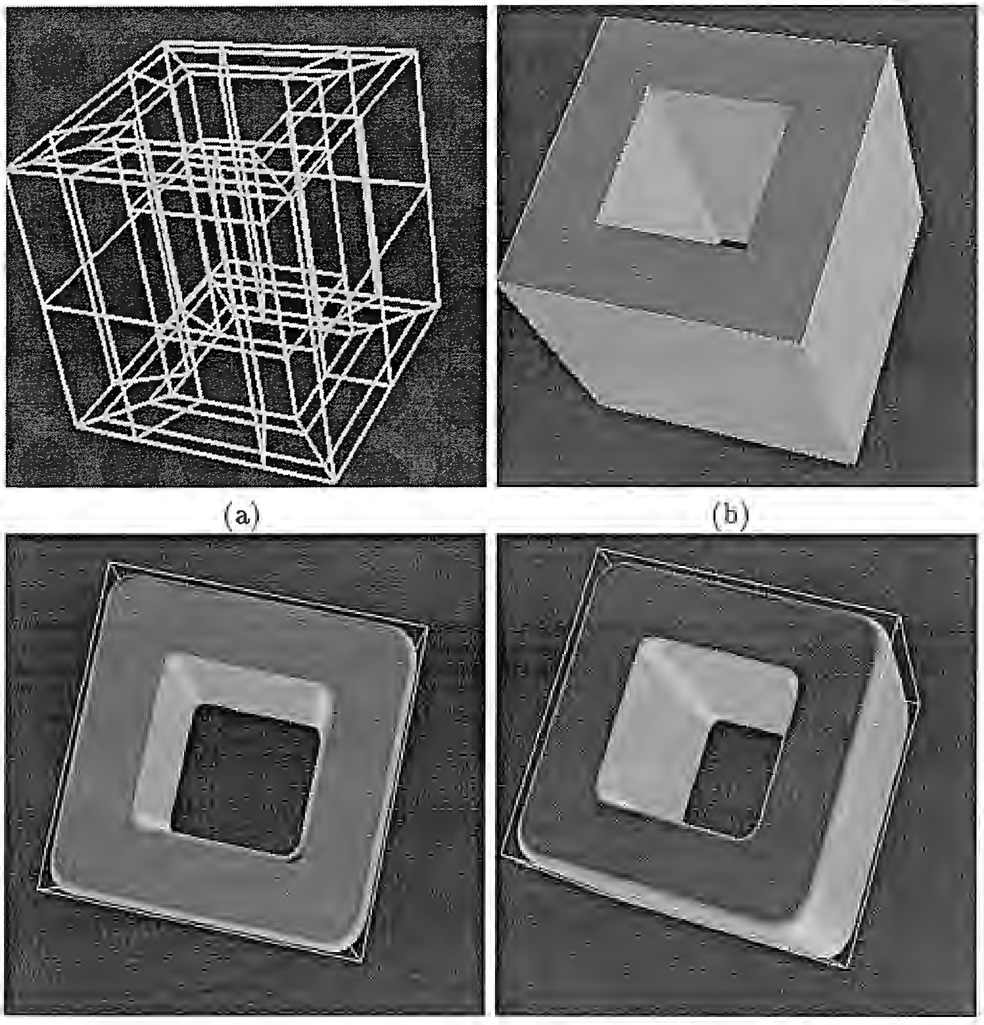

(a) (b) (c) (d)

Figure 9.12: Rounding of polyhedral toroid. (a) Wireframe. (b) Shaded object. (c-d) Final rounded shape.

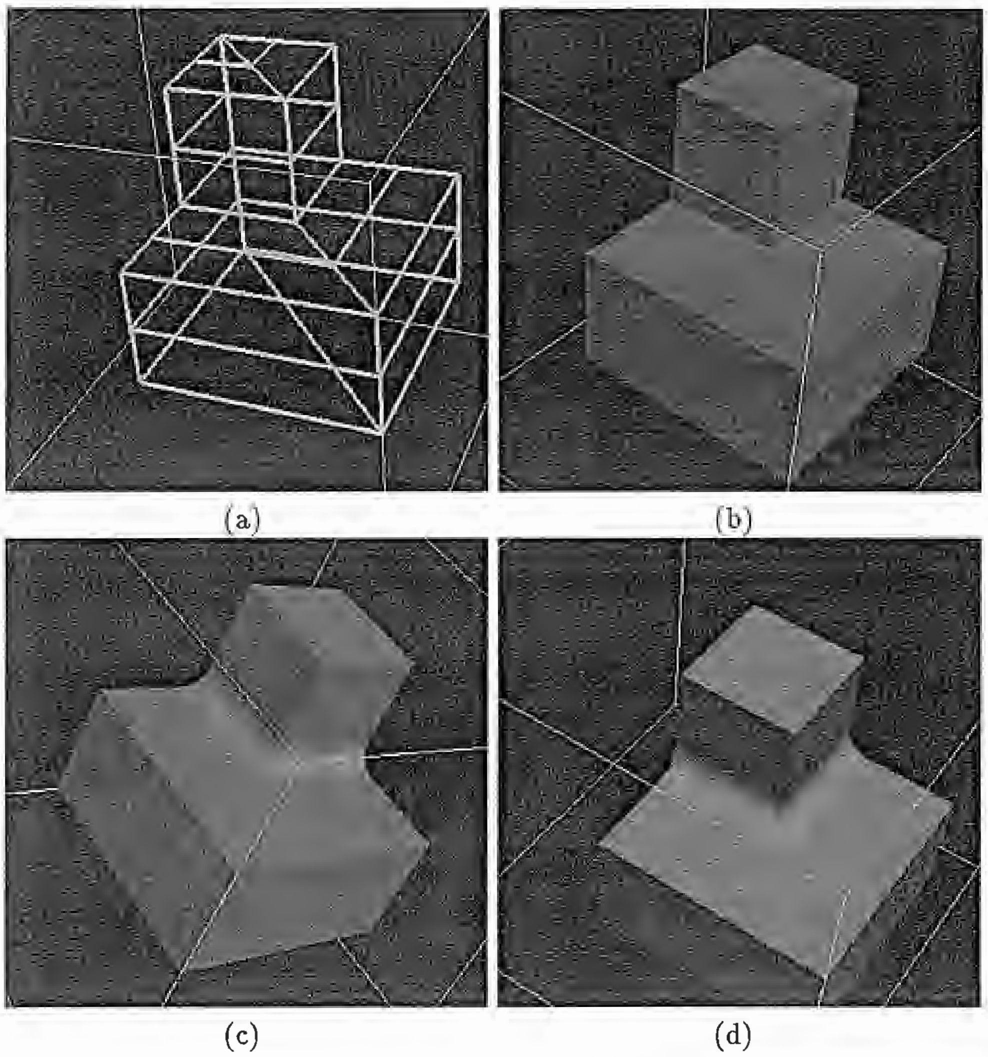


Figure 9.13: Rounding of cubical solid. (a) Wireframe. (b) Shaded object. (c-d) Final rounded shape.

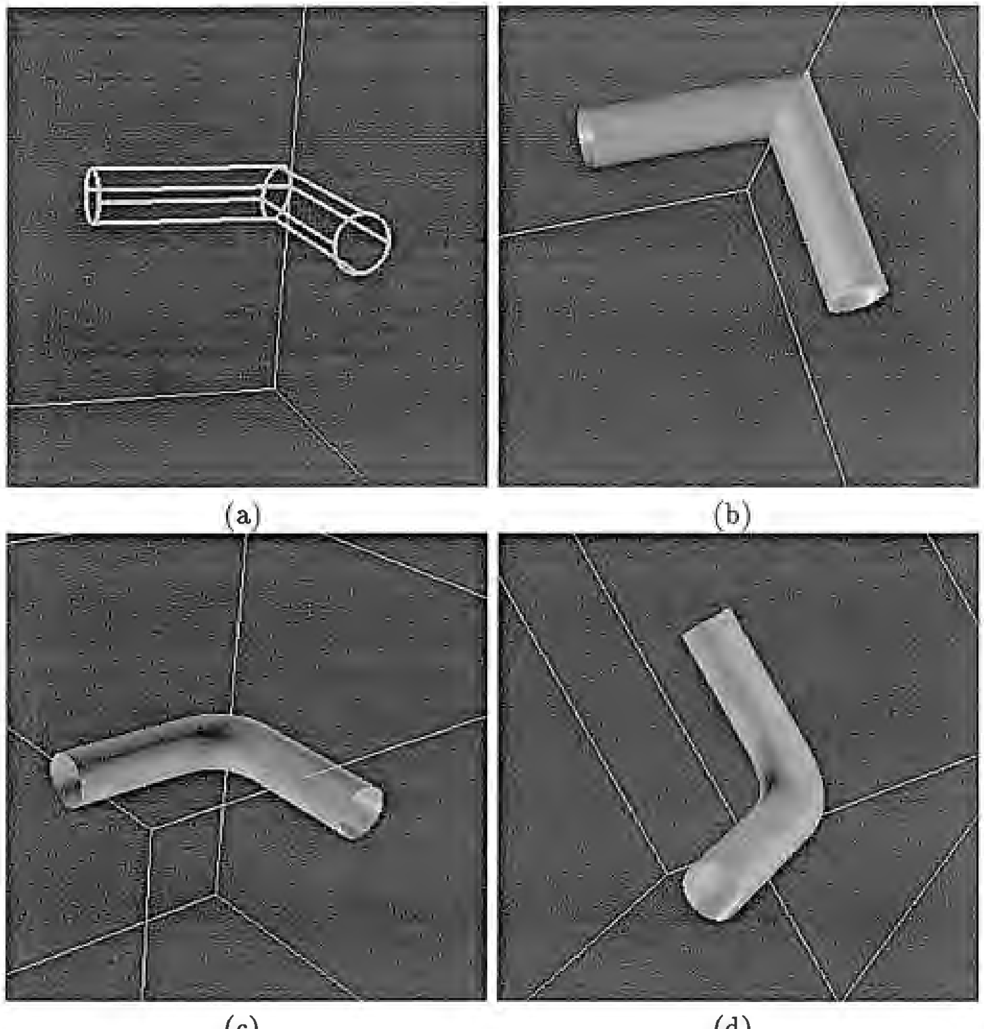

(a) (b) (c) (d)

Figure 9.14: Surface blending of pipe. (a) Wireframe. (b) Shaded object. (c-d) Final smooth blend.

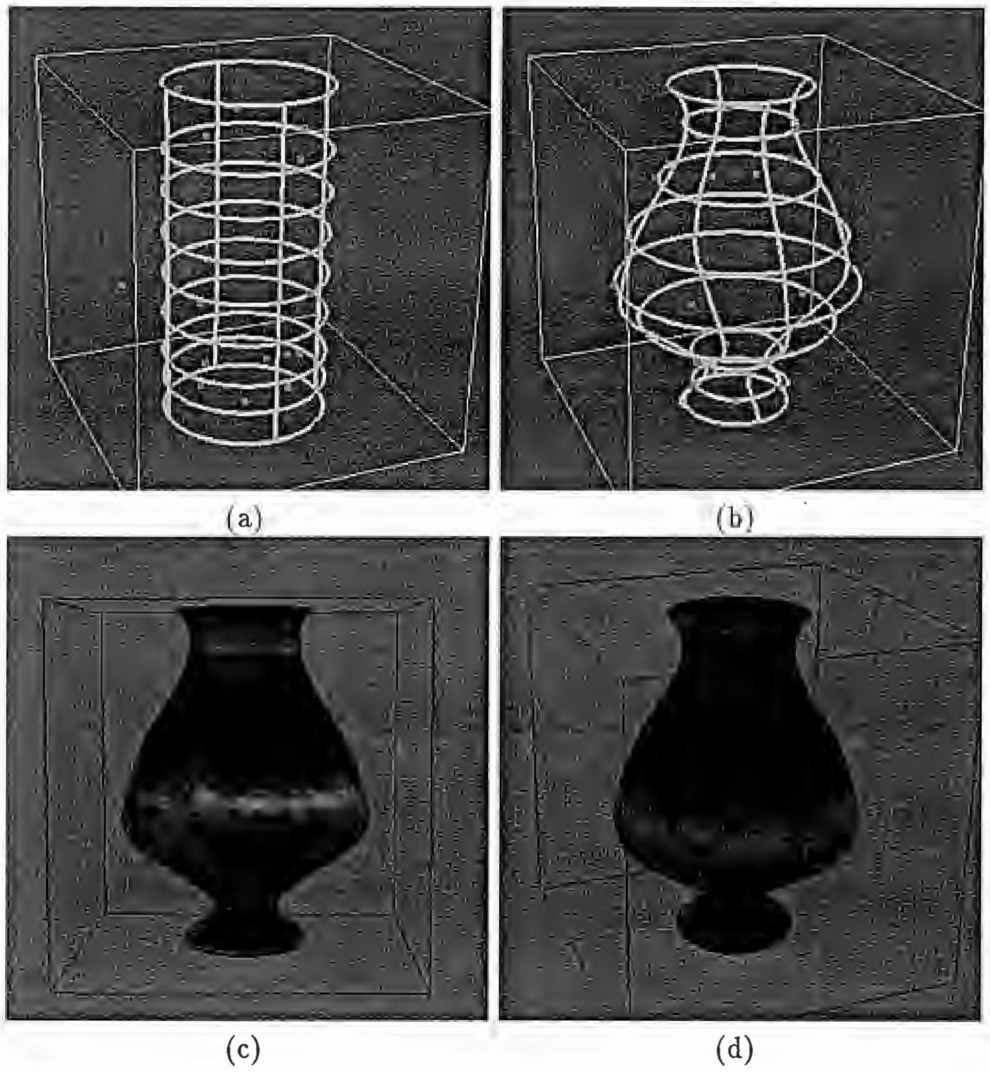


Figure 9.15: Fitting of 3D laser scanner data. (a) Original cylinder wireframe. (b) Reconstructed pot wireframe. (c-d) Textured pot.

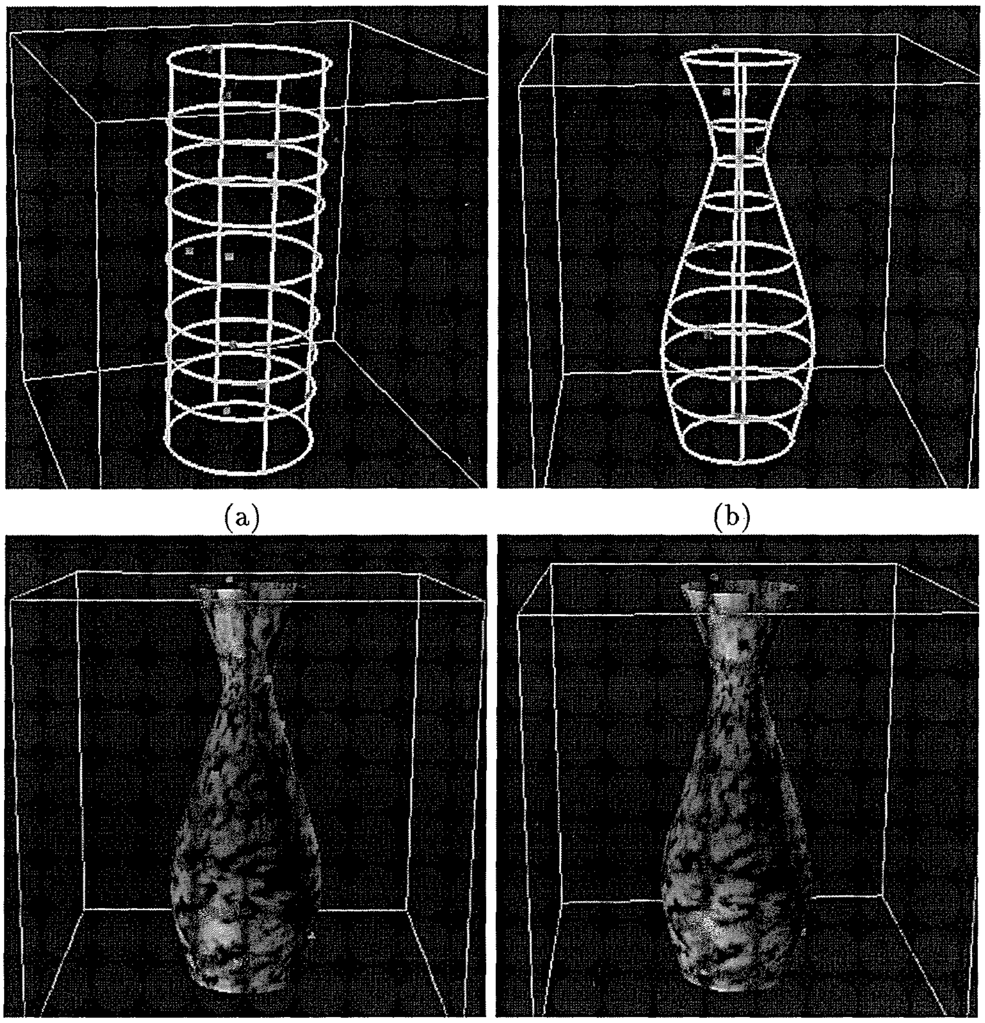

(a) (b) (c) (d)

Figure 9.16: Fitting of synthetic data. (a) cylinder wireframe. (b) Reconstructed vase wireframe. (c-d) Textured vase.

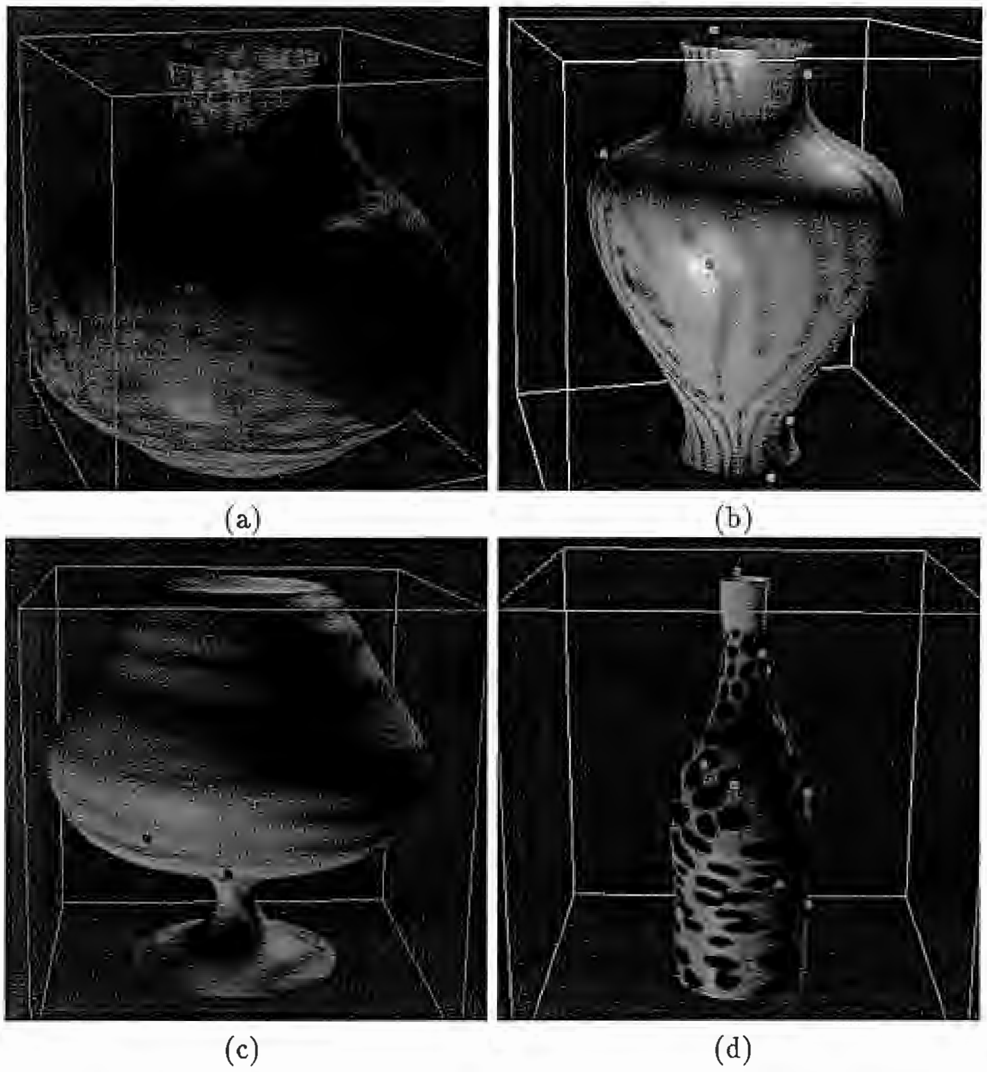


Figure 9.17: Four fitted shapes. (a) Pot. (b) Vase. (c) Glass. (d) Bottle.

Applications	*Physical Parameters*								
	μ	γ	$\alpha_{1,1}$	$\alpha_{2,2}$	$\beta_{1,1}$	$\beta_{1,2}$	$\beta_{2,2}$	Δt	k
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
Hill fitting	0.0	10.0	0.0	0.0	10.0	10.0	10.0	0.04	1000.0
Convex/Concave fitting	0.0	5.0	0.0	0.0	5.0	5.0	5.0	0.04	2000.0
Mountain/Valley fitting	0.0	25.0	0.0	0.0	1.0	1.0	1.0	0.04	2000.0
Quadratic object sculpting	5.0	25.0	10.0	10.0	1000.0	1000.0	1000.0	0.04	2000.0
Cubic patch sculpting	5.0	10.0	100.0	100.0	10.0	10.0	10.0	0.04	1000.0

Table 9.3: Physical parameters used in the examples. Parameter k denotes the stiffness of the spring force.

9.5 Triangular D-NURBS Applications

We have adopted the data structure, file, and rendering formats of existing geometric triangular B-spline software (Fong and Seidel, 1993). 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. Table 9.3 specifies the physical parameters used in the subsequent experiments. Fig. 9.18 illustrates the parametric domain triangulation of the various surfaces used in these experiments.

9.5.1 Rounding

Fig. 9.19 demonstrates the rounding of a sharp edge represented by a quadratic triangular B-spline surface with 36 control points. The sharp edge can be represented exactly with multiple control points. By restricting the control polygon to be a continuous net, we reduced the number of control points to 21. The initial wireframe and shaded shapes are shown in Fig. 9.19(a–b). After initiating the physical simulation, the sharp edges are rounded as the final shape equilibrates into the minimal energy state shown in Fig. 9.19(c).

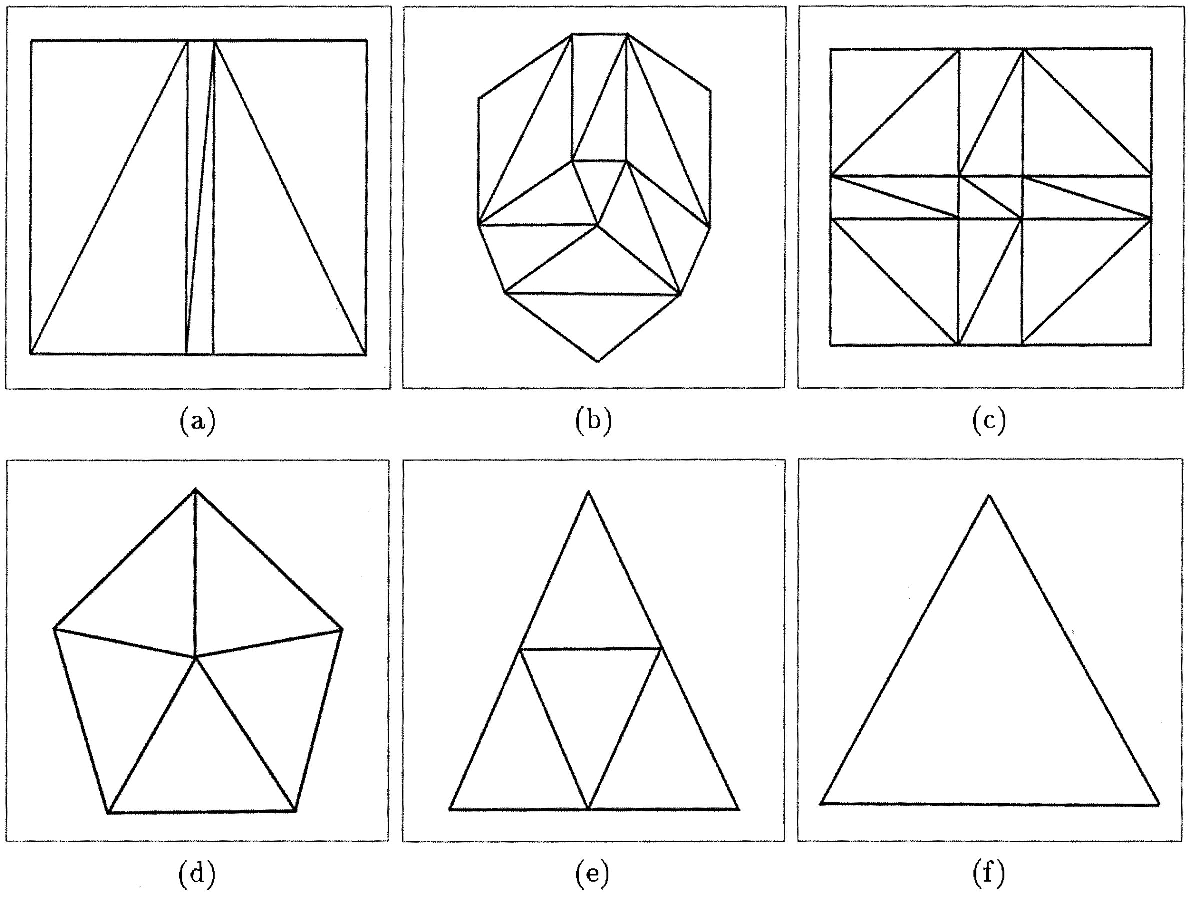
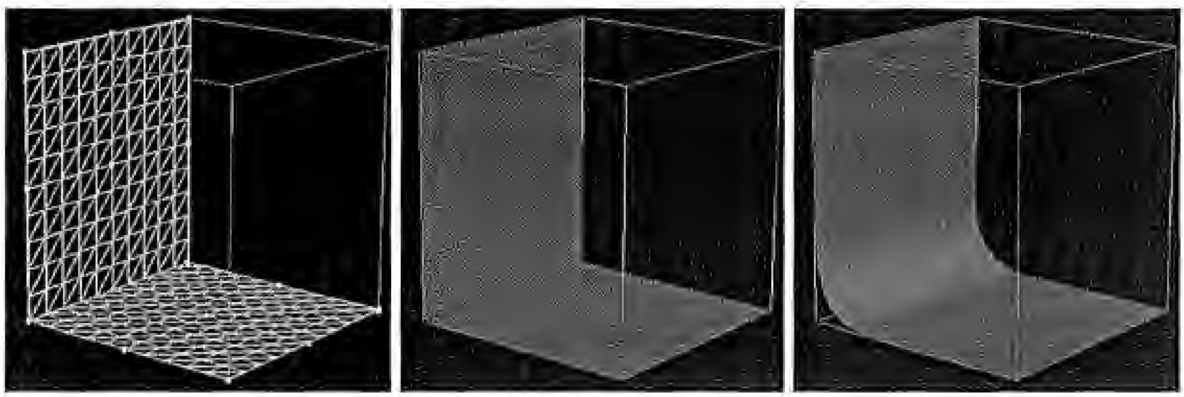
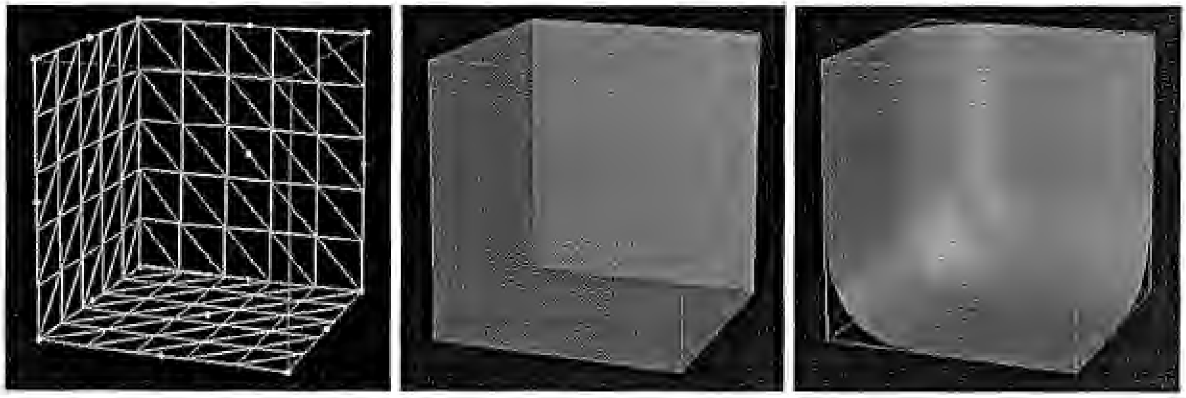


Figure 9.18: Domain triangulation of surfaces used in the examples (see text). (a) An edge. (b) A trihedral corner. (c) A bevel joint. (d) A pentagonal surface. (e) An open quadratic surface. (f) A cubic patch.



(a) (b) (c)

Figure 9.19: Rounding of an edge. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.



(a) (b) (c)

Figure 9.20: Rounding of a trihedral corner. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.

Fig. 9.20 illustrates the rounding of a trihedral corner of a cube. The corner is represented using a quadratic triangular B-spline with 78 control points. The initial wireframe and shaded shapes are demonstrated in Fig. 9.20(a–b). 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 shown in Fig. 9.20(c).

Fig. 9.21 shows a rounding example involving a bevel joint. The bevel joint is a quadratic triangular B-spline with 108 control points. The initial right-angle joint and the final rounded shapes are shown in Fig. 9.21(a–c).

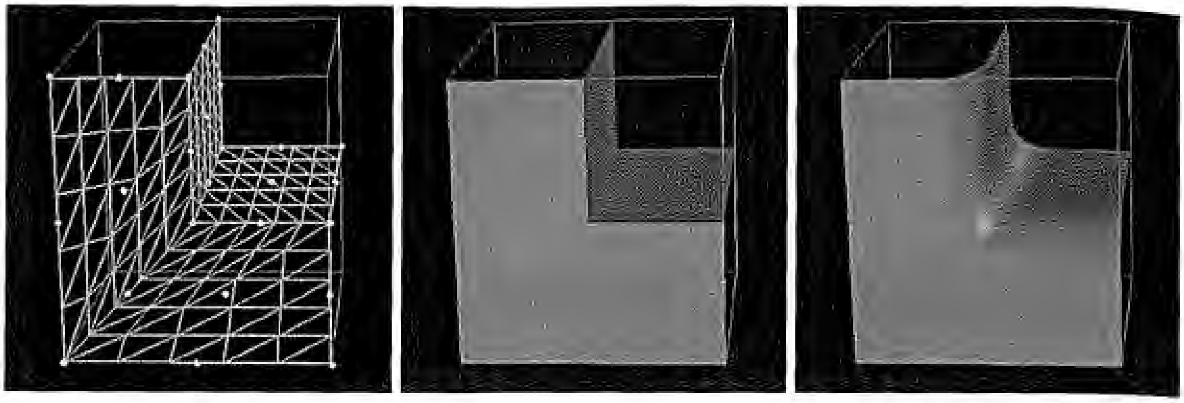


Figure 9.21: Rounding of a bevel joint. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.

9.5.2 Scattered Data Fitting

We present three examples of surface fitting using dynamic triangular B-spline surfaces coupled to data points through spring forces. The initial surface is a quadratic pentagon with 30 control points defined over the pentagonal domain. Three different sets of 6 data points are shown in Fig. 9.22(a1–c1) along with the initial surfaces. The spring forces associated with the data points are applied to the nearest points on the surfaces. Note that the spring attachments shown in Fig. 9.22(a1–c1) show the initial correspondence and are not fixed during the dynamic surface fitting process. Fig. 9.22(a2–c2) show the final fitted surfaces.

9.5.3 Dynamic Sculpting

Fig. 9.23 illustrates four shapes sculpted using spring forces. The initial open surface is generated using a quadratic B-splines with 24 control points. Second, a cubic triangular planar patch with 10 control points shown in Fig. 9.24(a) was dynamically manipulated into the shape shown in Fig. 9.24(b).

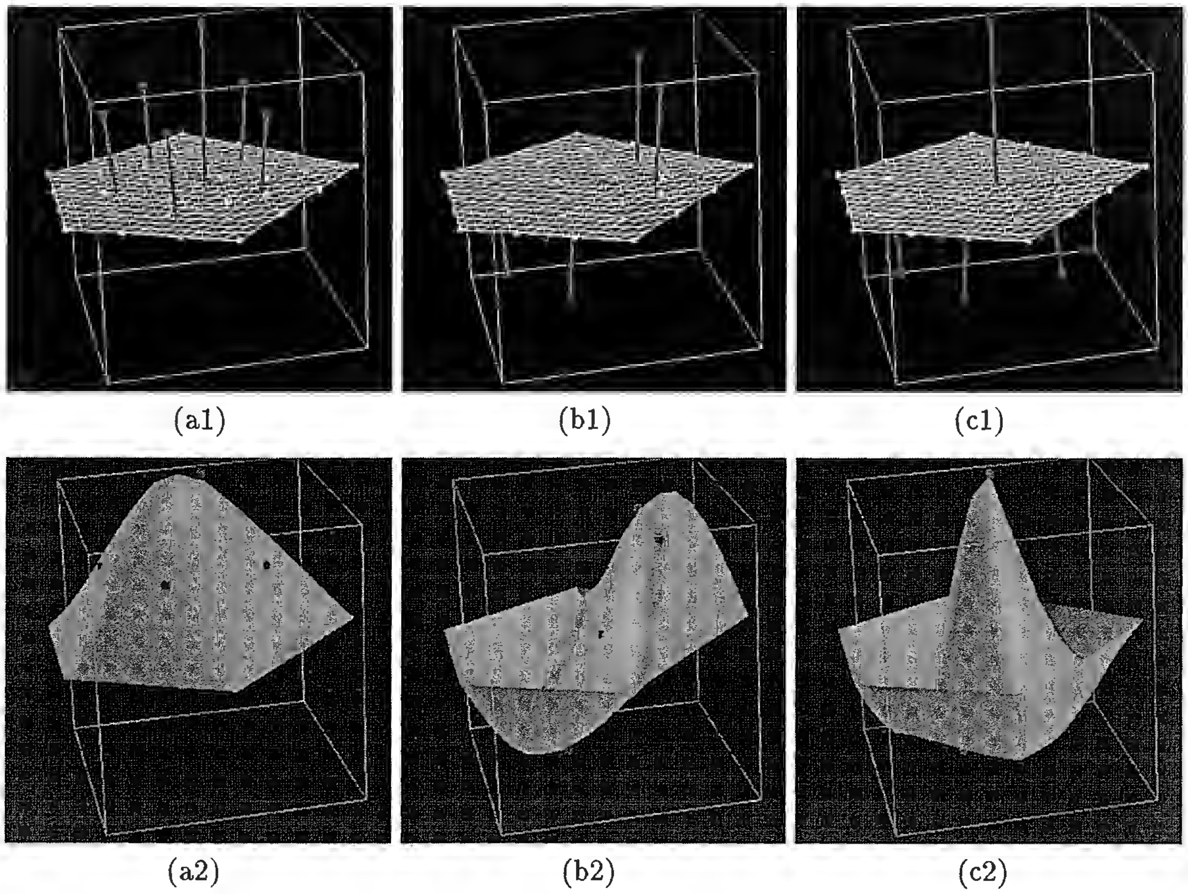


Figure 9.22: Fitting a pentagonal surface to three different sets of scattered data. Data points and initial surfaces (a1,b1,c1). Final fitted surfaces (a2,b2,c2).

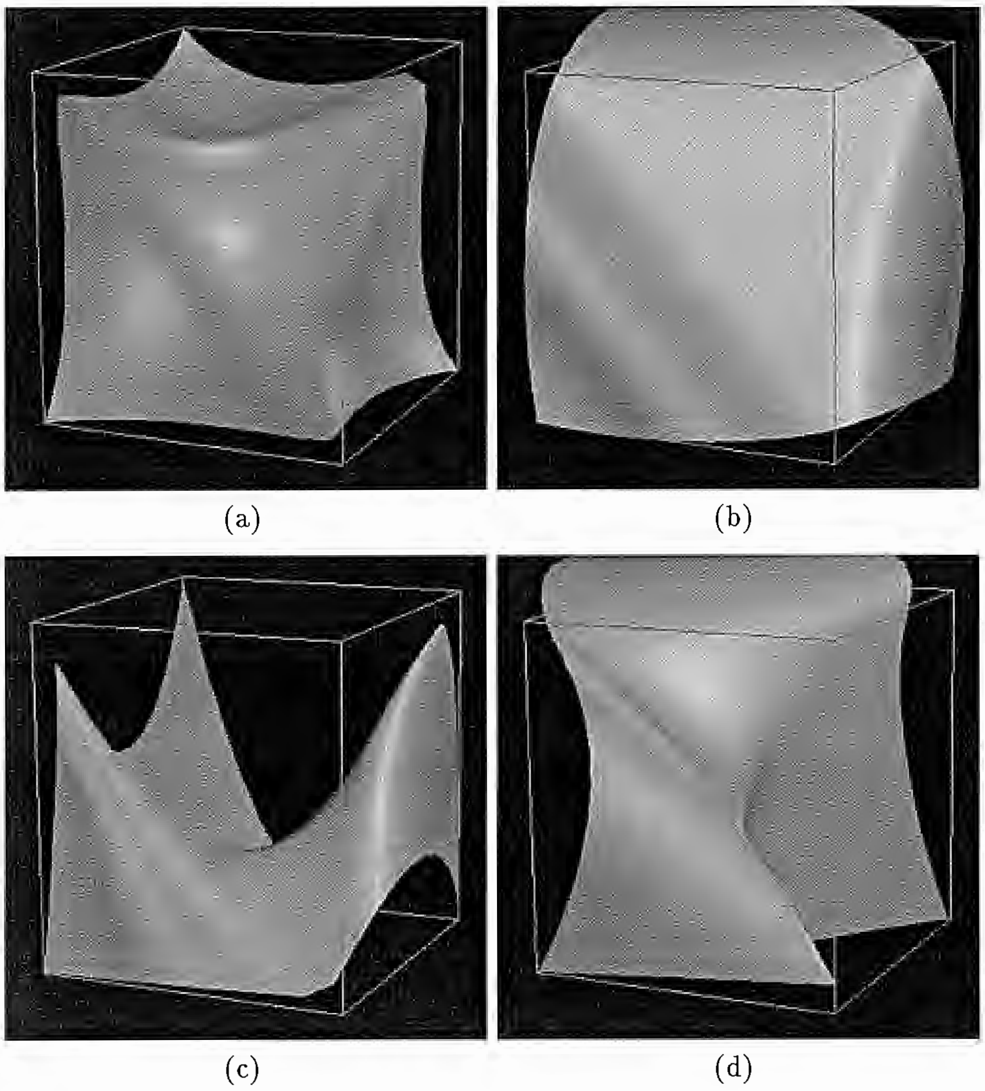


Figure 9.23: Interactive sculpting of an open quadratic surface into four different shapes (a–d).

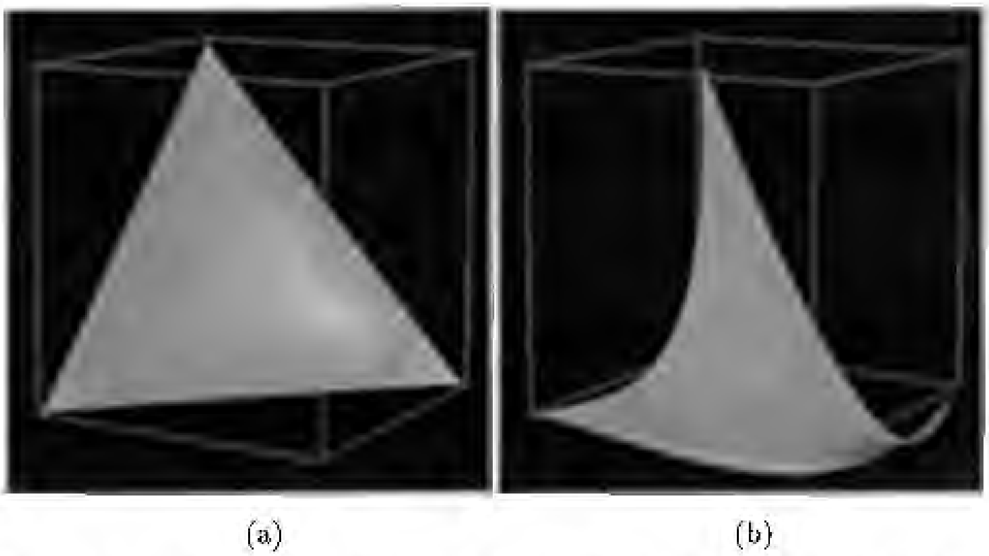


Figure 9.24: Interactive sculpting of a cubic patch (a) into another shape (b).

Chapter 10

Conclusion

We will now summarize the thesis and discuss future research topics.

10.1 Summary

The major contribution of this thesis is that it develops Dynamic Non-Uniform Rational B-Splines, or D-NURBS, and presents a new physics-based design paradigm. We also generalize our D-NURBS formulation to incorporate geometric constraints. The D-NURBS 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. Our D-NURBS formulation extends naturally to solids, albeit at proportionately greater computational cost.

When working with D-NURBS, a designer need not manipulate the individual degrees of freedom of an object. Instead, the designer can employ force-based "tools" to perform direct manipulation and interactive sculpting. Sculpting forces may be applied interactively to move the object or refine its shape. The physics-based model responds to applied simulated forces with natural and predictable dynamics, while the underlying geometric parameters, including the control points and weights, are determined automatically. Additional control over the shape is available through the modification of physical parameters. Elastic energy functionals allow the qualitative imposition of fairness criteria through quantitative means. Linear or nonlinear constraints may be imposed either as hard constraints that must not be violated, or as soft constraints to be satisfied approximately in the form of simple forces. Energy-based shape optimization for surface fairing is an automatic consequence of the dynamic model achieving static equilibrium subject to data constraints.

Our prototype interactive modeling system based on D-NURBS provides a promising approach to advanced surface design problems. We demonstrated the flexibility of our models in a variety of applications, including constraint-based optimization for surface design and fairing, automatic settings of weights in surface fitting, and interactive sculpting through applied forces. Furthermore, D-NURBS promise to serve as a basis for future research endeavors in physics-based CAGD, graphics, virtual reality, vision, and scientific visualization.

Since our models are built on the industry-standard NURBS geometric substrate, designers working with them can continue to make use of the existing array of geometric design toolkits. With the advent of high-performance graphics systems, the physics-based framework is poised for incorporation into commercial design systems to interactively model and sculpt complex shapes in real-time.

10.2 Future Research Topics

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. In CAD and graphics, interesting topics for future work include:

- Optimal knot vector selection in parametric free-form representations in accordance with aesthetic criteria and functional requirements. This goal may be achieved through the incorporation of the knot vector into the generalized coordinates of free-form physics-based models. In D-NURBS, for example, the Jacobian matrix entries (Chapter 4, 5, and 6) with respects to the (non-uniform) NURBS knot vector may not be explicitly formulated primarily due to piecewise rational basis functions of geometric NURBS which involve certain subsets of knots implicitly. Extra research endeavor is necessary to determine the optimal knot vector automatically for the NURBS formulation.
- Dynamic generalizations of other geometric forms including recursive subdivision surfaces and implicit functions.
- Design of new efficient and robust algorithms by leveraging the attractive properties of blossoming.
- Advanced user interaction techniques such as force-based sculpting tools that allow dynamic manipulations of shape positions, normals, and curvatures of objects.

The new, physics-based toolkits we have developed can be readily incorporated within commercial modeling systems in order to interactively model and sculpt extremely complex shapes in a real-time fashion. This thesis has demonstrated the feasibility of D-NURBS in a variety of CAGD problems such as constraint-based optimization, automatic parameter selection, variational parametric design, scattered data fitting, etc. New application topics may include shape interpolation/approximation with arbitrary topology subject to aesthetic criteria and shape interrogation, shape control with non-linear (usually curvature related) fairness functionals, and the integration of standard CAGD geometries and finite element analysis (FEA) for feature-based design. D-NURBS can also serve as a basis for future research in virtual reality and user interaction. With advanced special-purpose graphics processors and devices, it is both necessary and desirable to implement virtual reality environments (hardware and software) that provide designers 3D real-time force-based sculpting toolkits. Such a system would allow dynamic interaction between the physical world and virtual world.

Physics-based techniques have proven to be useful for realistic 3D computer animation. The generative power of the D-NURBS developed in this thesis is applicable for constrained shape synthesis, nonrigid object animation, and interactions. In particular, there are opportunities to apply D-NURBS models to the modeling and simulation of biological objects such as human bodies and their motions.

The production of sophisticated animation systems necessitates the development of automatic motion control techniques and higher level behavioral control mechanisms. Recent work includes constraints and physical simulation, hierarchical control, and behavioral control. We would investigate the application of control theory to D-NURBS, interactive path planning and automatic motion control with NURBS-based schemes, and automatic collision detection and obstacle avoidance through dynamic constraints. This promises to be critical to the creation of complicated animations in a user-friendly way.

In computer vision, robotics, and multidimensional medical imaging, it is important to systematically develop integrated model-based approaches for various tasks including shape modeling, motion analysis, matching, geometric reasoning, and realistic simulation with full dynamic animation and visualization capabilities. D-NURBS, as well as common algebraic functions promise to be critical to the development of such techniques for both shape reconstruction and shape recognition. One valuable investigation is to work towards the development of a prototype visual system based on popular splines for various visual tasks such as 3D shape fitting to sparse noise-corrupted visual data, nonrigid shape estimation and motion tracking of 3D objects from time-varying observations, and recovery of articulated figures.

Appendix A

Other Geometric Representations

This appendix reviews other geometric formulations which are not discussed in Chapter 2, including recursive subdivision surfaces, implicit forms, and variational splines.

A.1 Subdivision Forms

Because the planar parametric domain is an open surface, it is almost impossible to model an object of arbitrary genus with a single non-degenerate B-spline surface. Thus, extra boundary continuity constraints must be enforced. Moreover, singularity seems to be inevitable in modeling real-world objects. Although trimming surfaces offer an alternative, it can destroy the compact and concise representation of spline surfaces. The domain-less subdivision schemes (see Fig. A.1) have proven to be very promising for modeling extremely complex objects.

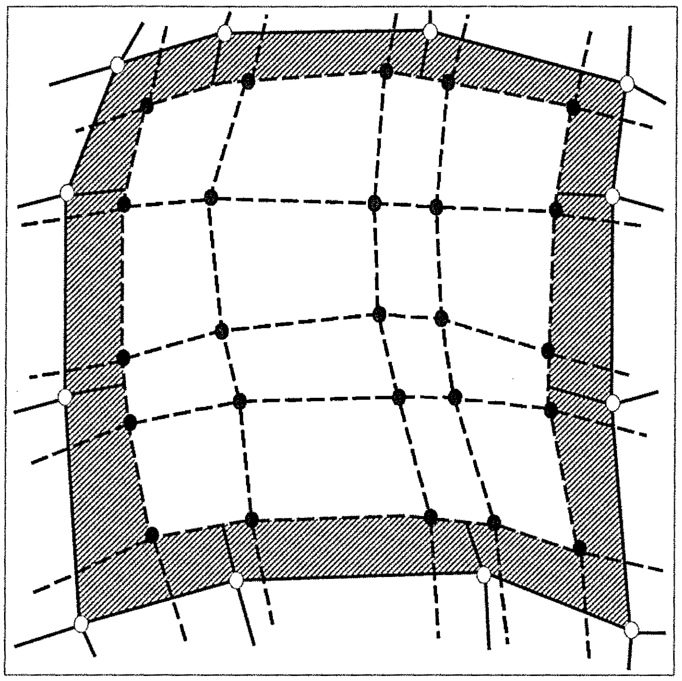


Figure A.1: Recursive subdivision surface.

In 1974, Chaikin described an efficient algorithm which permits the recursive subdivision of an initial polygon and makes a series of new polygons converge to a smooth curve. In 1978, Catmull and Clark generalized Chaikin's approach and constructed a smooth surface with arbitrary topology through the recursive subdivision of an arbitrarily shaped polyhedron (Catmull and Clark, 1978). This surface reduces to a G^2 B-spline surface except at a finite number of extraordinary points where only tangent plane continuity is achieved. The essential part of these new schemes is the subdivision rule which can guide the generation of an infinite set of consecutively refined meshes based on an initial polyhedron. The subdivision surface is the limit to which this infinite corner-chopping process converges. Note that, different subdivision rules determine distinct shapes of the converged smooth surface/curve.

The key advantage of subdivision surfaces is that smooth surfaces of arbitrary topology can be obtained based on topologically complex initial meshes. However, recursive subdivision surfaces lack parametric domains, which precludes their immediate pointwise evaluation and hence may limit the applicability of these schemes. In general, subdivision surfaces can be reduced to regular B-spline surfaces except at a finite number of singular points. Since the singular points need special treatment, subdivision surfaces are not equivalent to the previously described standard spline formulations. Instead, they are often referred to as B-spline-like surfaces. The behavior of those singular points are determined by eigenvalues of a set of matrices (Doo and Sabin, 1978).

Due to the lack of explicit parameterization, the fast evaluation of subdivision surfaces is restrained. Peters developed an algorithm which can generate a bivariate C^1 surface with an explicit parameterization from an irregular control point mesh (Peters, 1993). His approach produces a refined mesh of points through the use of the subdivision procedure. The new control point mesh is then used for generating a C^1 surface consisting of a set of quadratic and cubic patches. The major advantage of this algorithm is that it combines the intuitive subdivision of the irregular meshes with low-degree parameterization. The key constituents of this technique are (i) refining the initial mesh and generating the control points of box spline, (ii) converting the box spline surface into Bezier form, and (iii) filling the remaining holes with cubic triangular patches. Although open or closed surfaces of general topological structure can be obtained, this algorithm is complicated and *ad hoc*. Furthermore, a large number of patches are necessary to describe a complex shape.

A.2 Algebraic Functions

Although the most popular representation in CAGD is the parametric form, the traditional representation of geometric entities (in classical analytic geometry) is the implicit function. While parametric forms are well suited for shape editing and rendering, implicit forms are useful for point membership classification. Fig. A.2 shows that a two-dimensional implicit

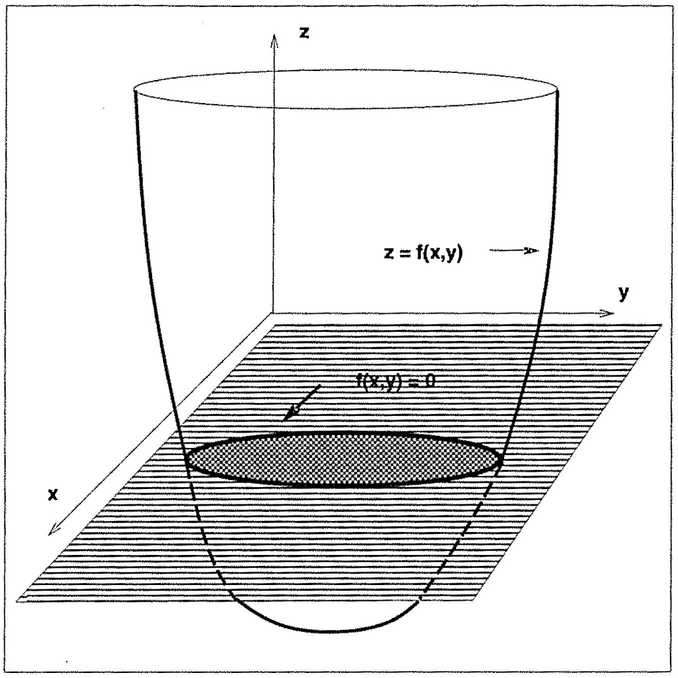


Figure A.2: The construction of a two-dimensional implicit function.

function $f(x,y) = 0$ is obtained through a planar cross-section of the three-dimensional scalar function $z = f(x,y)$. The most simple form for implicit functions is the power basis expression of degree n

$$\sum_{i,j,k,i+j+k=n} a_{ijk}x^i y^j z^k = 0. \tag{A.1}$$

Like the monomial form in (2.1), coefficients in (A.1) provide neither direct geometric interpretation nor intuitive insight into the underlying shape because the power basis implicit function is algebraic, and not geometric. It is equally difficult to predict the influence on the shape caused by the coefficient perturbation. Also there are no convenient tools for the intuitive shape control of this algebraic surface.

It can be shown that the set of implicit algebraic surfaces is actually larger than that of rational surfaces. This set is also closed under certain geometric operations. Every rational parametric curve/surface can be represented by an implicit algebraic equation, but not vice versa. Despite their representation power, implicit algebraic equations have shortcomings from the perspective of computational geometry. First, digitizing and rendering an implicit function is always difficult. Second, the derived shape may have separate components which are not indicated in the empirical data, therefore, extra polynomial constraints are necessary to ensure no singularities or self-intersections.

Despite these shortcomings, algebraic functions can be used for free-form modeling. Typical applications include forcing an algebraic surface to interpolate a set of points or spatial curves, and using piecewise algebraic patches to form a complex shape satisfying certain continuity requirements across patch boundaries. Sederberg discussed the modeling techniques for cubic algebraic surfaces (Sederberg, 1990a; Sederberg, 1990b). Hoffmann systematically reviewed the implicit function including the implicitization, parameterization, and the parametric/implicit conversion in CAGD (Hoffmann, 1993). Bajaj and Ihm presented an efficient algorithm to implement Hermite interpolation of low-degree algebraic surfaces with C^1 or G^1 continuity (Bajaj and Ihm, 1992). The interpolation is essentially reduced to a homogeneous linear system where the unknowns are coefficients of the algebraic surface. Note that, neither point nor curve interpolation is an attractive mechanism for defining an implicit surface because it is difficult for designers to predict the surface behavior beyond interpolating curves and points.

The benefits of using implicit functions are due to their low degree and computational efficiency. It is highly desirable to permit the interactive and direct manipulation of implicit algebraic surfaces. A piecewise algebraic surface patch can be defined with trivariate barycentric coordinates using a reference tetrahedron, and a regular lattice of control points and weights can be associated with this bounding tetrahedron. Consider a tetrahedron with noncoplanar vertices $\mathbf{v}_{n000}$, $\mathbf{v}_{0n00}$, $\mathbf{v}_{00n0}$, and $\mathbf{v}_{000n}$. Let (r, s, t, u) denote the local barycentric coordinates of a point $\mathbf{p}$ in the tetrahedron. By definition, we have

$$\mathbf{p} = r\mathbf{v}_{n000} + s\mathbf{v}_{0n00} + \mathbf{v}_{00n0} + u\mathbf{v}_{000n},$$

where $r+s+t+u=1$. Then, we define a set of $(n+1)(n+2)(n+3)/6$ control points $\mathbf{p}_{ijkl}$ such that

$$\mathbf{p}_{ijkl} = \frac{i\mathbf{v}_{n000} + j\mathbf{v}_{0n00} + k\mathbf{v}_{00n0} + l\mathbf{v}_{000n}}{n},$$

where $i,j,k,l \geq 0$, and $i+j+k+l=n$. A weight w_{ijkl} is assigned to each control point. The algebraic patch inside the tetrahedron can be formulated using Bernstein-Bezier basis functions as

$$\sum_i \sum_j \sum_k \sum_{l=n-i-j-k} w_{ijkl} \frac{n!}{i!j!k!l!} r^i s^j t^k u^l = 0, \tag{A.2}$$

where i,j,k, and l are non-negative, and (r,s,t,u) represents the local barycentric coordinates of arbitrary vertices on this algebraic patch. Fig. A.3 shows the construction of a quadratic algebraic patch in which $n=2$. The key advantage is that control points and weights provide a meaningful way to control the shape of the patch. Note that, unlike weights of rational splines, the weights of algebraic patches are not independent. Like Bezier curves and surfaces, algebraic patches have some nice properties such as local control of control points and weights, boundary interpolation, gradient control, and self-intersection avoidance (see (Sederberg, 1990a; Sederberg, 1990b; Bajaj and Ihm, 1992) for the details). Patches can be abutted smoothly for modeling complex shapes. By applying subdivision, additional degrees of freedom are generated, allowing patches to satisfy extra continuity conditions across the boundaries.

Implicit functions have been used in various applications in graphics and CAGD. Generalizing ordinary algebraic modeling methods, Muraki used the "blobby" models for volumetric data fitting (Muraki, 1991). Although this approach allows the automatic extraction of a symbolic shape description from the range data, it is too slow to be of practical importance. The convolution surface (Bloomenthal and Shoemake, 1991) is another example of graphical model representation with implicit functions, and this technique is based on skeletons. The convolution surface generalizes the "blobby" model because the continuous

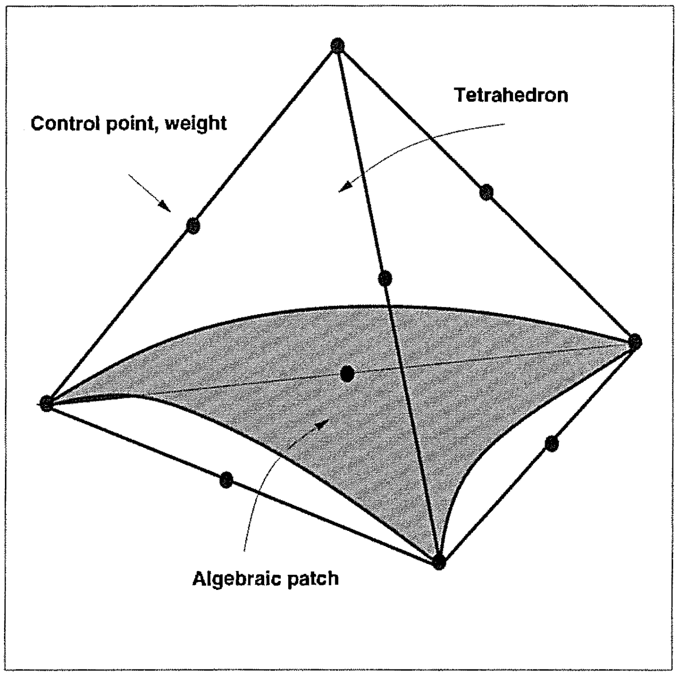


Figure A.3: A quadratic algebraic patch.

integral operator is used to replace the discrete summation of exponential-based operators. Thus, the shape of the convolution surface is smooth and blendings are well behaved. Dutta, Martin and Pratt (Dutta, Martin and Pratt, 1993) used piecewise Dupin's quartic cyclides for free-form surface modeling and blending because of the low algebraic degree, rational parametric form, and simple and intuitive geometric parameters of cyclides.

A.3 Geometric and Variational Splines

The main task in CAGD is to smoothly construct complex shapes with a parsimonious number of curve spans or surface patches. For instance, piecewise Bezier curves can be used to construct complex curves satisfying certain continuity requirements at conjunction points. Ordinary parametric continuity, however, has proven to be unnecessarily restrictive. Recently, a weaker continuity condition, geometric continuity, has received a great of deal

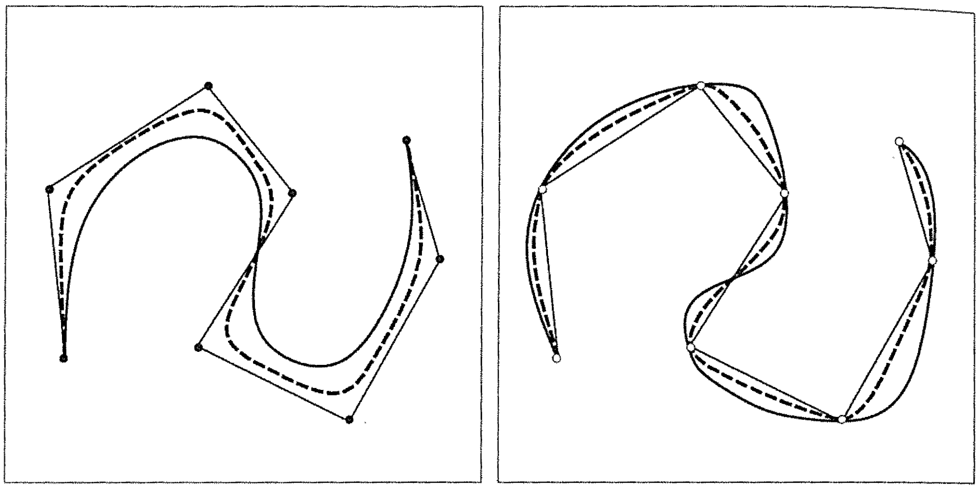

(a) (b)

Figure A.4: Tension effect on curve shape (a) the curve is generated by control polygon (b) the curve is generated through data interpolation.

of attention (see (Barsky and DeRose, 1989; Barsky and DeRose, 1990) for the details). By definition, two adjacent curves are said to have G^n continuity at a joint if and only if they meet with C^n continuity under arc-length parameterization. Obviously, G^1 means unit tangent vector continuity. G^2 implies curvature continuity in addition to G^1 continuity.

With geometric continuity, it is possible to introduce shape parameters independent of control points, which can provide designers more DOFs for shape design and control. It is always desirable to provide extra flexibility for geometric design. Recently, there has been a great deal of interest in the use of tension parameters as extra design handles beyond control vertices for piecewise polynomial (see Fig. A.4 for tension effects on the curve shape). Note that, tension parameters act as a parametric rescaling because the parametric continuity is reduced into geometric continuity. Many special splines involving geometric continuity have been developed during the past twenty years.

The Beta-spline (Barsky, 1983), in which adjacent curve spans satisfy geometric continuity, generalizes the uniform cubic B-spline. The uniformly shaped Beta-splines provide only two control parameters (bias and tension) for the whole curve, and thus, no local control is provided. Nonetheless, this generality allows designers to control the geometric shape with shape parameters in addition to control points. In general, a tension increase will pull the curve towards the control point, and a bias increase will pull the curve span towards the corresponding line segment of the control polygon. In particular, if the bias is one, and the tension is zero, the Beta-spline is reduced to the cubic B-spline. The effect of bias and tension and the necessary G^2 continuity conditions are discussed in (Goodman and Unsworth, 1986). Furthermore, a continuously-shaped Beta-spline has been developed (Barsky, 1983) where local control of bias and tension are obtained. Other forms of Beta-splines also exist. In (Barsky and DeRose, 1989; Barsky and DeRose, 1990), G^1 and G^2 Beta-splines are shown to be equivalent to composite quadratic and cubic Bezier splines, respectively.

DeRose and Barsky discussed a new class of splines, geometrically continuous Catmull-Rom splines (DeRose and Barsky, 1988), which can be expressed as

$$\mathbf{c}(u) = \sum_i \mathbf{a}_i(u) W_i(u), \tag{A.3}$$

where each $\mathbf{a}_i(u)$ is constructed to interpolate the $k+1$ vertices $\mathbf{v}_i, \ldots, \mathbf{v}_{i+k}$. Unlike the previously discussed spline schemes, $\mathbf{a}_i(u)$ is a vector-valued interpolating function. As a result, geometrically continuous Catmull-Rom splines can either interpolate or approximate data points. It has both control vertices and shape parameters which can be used for shape manipulation. When used in applications, however, its control vertices and shape parameters are often transformed into the equivalent Bezier control polygon to take advantage of various efficient Bezier toolkits.

Many special splines have also been defined via variational forms. In 1974, Nielson generalized the exponential related spline under tension. He proposed the Nu-spline as a G^2 (continuous curvature) piecewise polynomial minimizing the following functional over

$[t_0, t_n]$:

$$\int_{t_0}^{t_n} \|\mathbf{f}''(u)\|^2 du + \sum_{i=0}^{n} \nu_i \|\mathbf{f}'(u_i)\|^2, \tag{A.4}$$

subject to the interpolation $\mathbf{f}(t_i) = \mathbf{d}_i$ and necessary end conditions. Based on (A.4), Nielson further developed a geometric representation which allows convenient tension control while maintaining the global G^2 continuity (Nielson, 1984). This curve is composed of piecewise planar rational polynomials with tangent direction continuity and zero curvature at each end. Similar to the Nu-spline, it permits local control of tension parameters.

In 1985, Hagen proposed the Tau-spline (Hagen, 1985) which is a quintic spline that maintains both curvature and torsion continuity by minimizing

$$\int_{t_0}^{t_n} \left\|\mathbf{f}^{(k)}(u)\right\|^2 du + \sum_{i=0}^{n} \sum_{j=1}^{k-1} \nu_{i,j} \left\|\mathbf{f}^{(j)}(u_i)\right\|^2 \tag{A.5}$$

subject to interpolatory constraints. Pottmann presented a special spline with tension control (Pottmann, 1990) which generalizes the Tau-spline and possesses third-order parametric continuity and a smooth torsion plot. Lasser explicitly formulated a Bezier representation for the Tau-spline (Lasser, 1990). Thus, many efficient algorithms of Bezier splines are also applicable to Tau-splines.

In 1986, Nielson extended his Nu-spline to the surface case (Nielson, 1986). The straightforward tensor-product generalization of Nu-spline is not very useful because the arbitrary $n+m$ tension parameters will affect the entire curve network. In contrast to ordinary piecewise splines, Nu-spline provides extra tension parameters. When used for various modeling applications, however, Nu-splines are often converted into piecewise Hermite polynomials to expedite spline evaluation.

Foley presented a weighted Nu-spline interpolant (Foley, 1988) which is the C^1 piecewise cubic polynomials by minimizing

$$\sum_{i=0}^{n-1} w_i \int_{t_i}^{t_{i+1}} \|\mathbf{f}''(u)\|^2 du + \sum_{i=0}^{n} \nu_i \|\mathbf{f}'(u_i)\|^2, \tag{A.6}$$

subject to the interpolatory conditions $\mathbf{f}(t_i) = \mathbf{d}_i$. The weighted Nu-spline obtained from (A.6) can be used as a shape-preserving interpolant. Derivative constraints enforce the piecewise cubic interpolant to preserve local monotonicity. In addition, tension parameters can be used for shape modification. Furthermore, a cardinal basis for univariate weighted Nu-splines is formulated (Foley and Ely, 1989). Unlike cubic Hermite basis functions, cardinal basis of weighted Nu-splines are not local. They can not be evaluated easily and efficiently. These cardinal bases can be further used to construct a weighted Nu-spline surface which is a piecewise bicubic surface. The weighted Nu-spline surfaces can be interactively changed by manipulating the control points (interpolating points) and interval/point tension parameters. Note that, the surface boundaries are also weighted Nu-splines. In particular, when all weights are equal, and all tensions are zero, the weighted Nu-spline is reduced to a C^2 cubic spline.

Given $t_0 < t_1 < \ldots < t_n$, the weighted spline interpolant (Foley, 1986) is the C^1 piecewise cubic function $f(u)$ that minimizes

$$\int_{t_0}^{t_n} w(u)(f^{(2)}(u))^2 du \tag{A.7}$$

subject to $f(t_i) = d_i$ and relevant end conditions. Over each interval, weighted cubic splines have piecewise constant tension control $w(u)$. In addition, they generalize cubic B-splines. The B-spline-like basis functions can be formulated explicitly. This can facilitate interactive design. Later, based on (A.7), Foley derived weighted bicubic splines (Foley, 1987) which minimize the following functional:

$$\int_{t_0}^{t_n} \int_{s_0}^{s_m} w(u,v)(\mathbf{f}_{uu,vv}(u,v))^2 du dv \tag{A.8}$$

subject to the interpolation constraints $f(u_i, v_j) = d_{i,j}$. This scheme has been used for the interpolation of rapidly varying data. The surface is actually a piecewise bicubic Hermite interpolant whose unknown derivatives can be obtained by solving a linear system of equations.

Cohen formulated local basis functions for the locally tensioned splines (LT-spline) which are essentially piecewise C^0 cubics having continuous curvatures (Cohen, 1987). It is shown that the LT-splines have the variation diminishing property, the convex hull property, and a straightforward knot insertion algorithm. Both curves and individual basis functions can be easily evaluated. When used for shape modeling, however, LT-splines are often transformed into equivalent B-splines in order to benefit from various algorithms for efficient evaluation, refinement, and hierarchical modeling. Note that, the fact that the B-spline is employed as the underlying formulation for this and other special splines in various applications demonstrates that B-splines are more powerful and flexible schemes.

References

Auerbach, S., Meyling, R. G., Neamtu, M., and Schaeben, H. (1991). Approximation and geometric modeling with simplex B-splines associated with irregular triangles. *Computer Aided Geometric Design*, 8(1):67–87.

Bajaj, C. and Ihm, I. (1992). Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91.

Barnhill, R. (1983). A survey of the representation and design of surfaces. *IEEE Computer Graphics and Applications*, 3(7):9–16.

Barnhill, R. (1985). Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design*, 2(1):1–17.

Barnhill, R., Birhoff, G., and Gordon, W. (1973). Smooth interpolation in triangles. *J. Approximation Theory*, 8:114–128.

Barr, A. (1981). Superquadrics and angle preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23.

Barsky, B. (1983). Local control of bias and tension in Beta-splines. *ACM Transactions on Graphics*, 2(2):109–134.

Barsky, B. and DeRose, T. (1989). Geometric continuity of parametric curves: three equvilent characterization. *IEEE Computer Graphics and Applications*, 9(6):60–68.

Barsky, B. and DeRose, T. (1990). Geometric continuity of parametric curves: constructions of geometrically continuious plines. *IEEE Computer Graphics and Applications*, 10(1):60–68.

Bartels, R., Beatty, J., Booth, K., Bosch, E., and Jolicoeur, P. (1993). Experimental comparison of splines using the shape-matching paradigm. *ACM Transactions on Graphics*, 12(3):179–208.

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

Bloomenthal, J. and Shoemake, K. (1991). Convolution surfaces. *Computer Graphics*, 25(4):251–256.

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

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

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

Catmull, E. and Clark, J. (1978). Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355.

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

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

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

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

Cheng, F. and Barsky, B. (1991). Interproximation: interpolation and approximation using cubic spline. *Computer-Aided Design*, 23(10):700–706.

Chou, J. and Piegl, L. (1992). Data reduction using cubic rational B-splines. *IEEE Computer Graphics and Applications*, 12(3):60–68.

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

Cohen, E. (1987). A new local basis for designing with tensioned splines. *ACM Transactions on Graphics*, 6(2):81–122.

Cox, M. (1972). The numerical evaluation of B-splines. *J. Institute of Mathematics and its Applications*, 10:134–149.

Dahmen, W. and Micchelli, C. (1982). On the linear independence of multivariate B-splines, I. triangulations of simploids. *SIAM J. Numer. Anal.*, 19(5):993–1012.

Dahmen, W. and Micchelli, C. (1983). Recent progress in multivariate splines. In Chui, C., Schumaker, L., and Ward, J., editors, *Approximation Theory IV*, pages 27–121. Academic Press, New York.

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

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

de Boor, C. (1976). Splines as linear combinations of B-splines. In Lorentz, G., Chui, C., and Schumaker, L., editors, *Approximation Theory II*, pages 1–47. Academic Press, New York.

DeRose, T. and Barsky, B. (1988). Geometric continuity, shape parameters and geometric constructions for Catmull-Rom splines. *ACM Transactions on Graphics*, 7(1):1–41.

Doo, D. and Sabin, M. (1978). Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360.

Dutta, D., Martin, R., and Pratt, M. (1993). Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications*, 13(1):53–59.

Dyn, N., Levin, D., and Gregory, J. (1990). A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169.

Farin, G. (1986). Triangular Bernstein-Bezier patches. *Computer Aided Geometric Design*, 3(2):83–127.

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

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

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

Farouki, R. and Hinds, J. (1985). A hierarchy of geometric forms. *IEEE Computer Graphics and Applications*, 5(5):51–78.

Faux, I. and Pratt, M. (1979). *Computational Geometry for Design and Manufacture.* Ellis Horwood, Chichester,UK.

Ferguson, D. and Grandine, T. (1990). On the construction of surfaces interpolating curves: I. A method for handling nonconstant parameter curves. *ACM Transactions on Graphics*, 9(2):212–225.

Foley, T. (1986). Local control of interval tension using weighted splines. *Computer Aided Geometric Design*, 3(4):281–294.

Foley, T. (1987). Weighted bicubic spline interpolation to rapidly varying data. *ACM Transactions on Graphics*, 6(1):1–18.

Foley, T. (1988). A shape perserving interpolant with tension controls. *Computer Aided Geometric Design*, 5(2):105–118.

Foley, T. and Ely, H. (1989). Surface interpolation with tension controls using cardinal bases. *Computer Aided Geometric Design*, 6(2):97–109.

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

Forrest, A. (1990). Interactive interpolation and approximation by Bezier polynomials. *Computer-Aided Design*, 22(9):527–537.

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

Fowler, B. (1992). Geometric manipulation of tensor product surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, pages 101–108.

Galyean, T. and Hughes, J. (1991). Sculptinng: An interactive volumetric modeling technique. *Computer Graphics*, 25(4):267–274.

Goodman, T. and Unsworth, K. (1986). Manipulating shape and producing geometric continuity in Beta-spline curves. *IEEE Computer Graphics and Applications*, 6(2):50–56.

Gordon, W. and Riesenfeld, R. (1974). B-spline curves and surfaces. In Barnhill, R. and Riesenfeld, R., editors, *Computer Aided Geometric Design*, pages 95–126. Academic Press.

Gossick, B. (1967). *Hamilton's Principle and Physical Systems.* Academic Press, New York and London.

Grandine, T. (1988). The stable evaluation of multivariate simplex splines. *Mathematics of Computation*, 50(181):197–205.

Greiner, G. and Seidel, H.-P. (1994). Modeling with triangular B-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60.

Hagen, H. (1985). Geometric spline curves. *Computer Aided Geometric Design*, 2(1-3):223–227.

Hoffmann, C. (1993). Implicit curves and surfaces in CAGD. *IEEE Computer Graphics and Applications*, 13(1):79–88.

Hollig, K. (1982). Multivariate splines. *SIAM J. Numer. Anal.*, 19(5):1013–1031.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–77.

Hoschek, J. (1987). Approximate conversion of spline curves. *Computer Aided Geometric Design*, 4(1-2):59–66.

Kardestuncer, H. (1987). *Finite Element Handbook.* McGraw–Hill, New York.

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

Lasser, D. (1990). Two remarks on Tau-splines. *ACM Transactions on Graphics*, 9(2):198–211.

Loop, C. and DeRose, T. (1989). A multisided generalization of Bezier surfaces. *ACM Transactions on Graphics*, 8(3):204–234.

Lounsbery, M., Mann, S., and DeRose, T. (1992). Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52.

Lowe, T., Bloor, M., and Wilson, M. (1990). Functionality in blend design. *Computer-Aided Design*, 22(10):655–665.

McInerney, T. and Terzopoulos, D. (1993). A finite element model for 3d shape reconstruction and nonrigid motion tracking. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 518–523, Berlin, Germany.

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

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

Miller, J., Breen, D., Lorensen, W., O'bara, R., and Wozny, M. (1991). Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics*, 25(4):217–226.

Minoux, M. (1986). *Mathematical Programming*. Wiley, New York.

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

Muraki, S. (1991). Volumetric shape description of range data using Blobby Model. *Computer Graphics*, 25(4):227–235.

Nielson, G. (1984). A locally controllable spline with tension for interactive curve design. *Computer Aided Geometric Design*, 1(3):199–205.

Nielson, G. (1986). Rectangular ν-splines. *IEEE Computer Graphics and Applications*, 6(2):35–40.

Nielson, G. (1993). Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70.

Nielson, G., Foley, T., Hamann, B., and Lane, D. (1991). Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics and Applications*, 11(3):47–55.

Nielson, G. and Ramaraj, R. (1987). Interpolation over a sphere based upon a minimum norm network. *Computer Aided Geometric Design*, 4(1-2):41–57.

Parkinson, D. (1992). Optimisec biarc curves with tension. *Computer Aided Geometric Design*, 9(3):207–218.

Pentland, A. and Horowitz, B. (1991). Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742.

Pentland, A. and Sclaroff, S. (1991). Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729.

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

Peters, J. (1993). Smooth free-form surfaces over irregular meshes generalizing quadratic splines. *Computer Aided Geometric Design*, 10(3-4):347–361.

Piegl, L. (1985). Representation of quadric primitives by rational polynomials. *Computer Aided Geometric Design*, 2(1-3):151–155.

Piegl, L. (1986). The sphere as a rational Bezier surface. *Computer Aided Geometric Design*, 3(1):45–52.

Piegl, L. (1987a). Infinite control points–A method for representing surfaces of revolution using boundary data. *IEEE Computer Graphics and Applications*, 7(3):45–55.

Piegl, L. (1987b). On the use of infinite control points in CAGD. *Computer Aided Geometric Design*, 4(1-2):155–166.

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

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

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

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

Piegl, L. and Tiller, W. (1989). A menagerie of rational B-Spline circles. *IEEE Computer Graphics and Applications*, 9(5):48–56.

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

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

Pottmann, H. (1990). Smooth curves under tension. *Computer Aided Design*, 22(4):241–245.

Pratt, M., Goult, R., and Ye, L. (1993). On rational parametric curve approximation. *Computer Aided Geometric Design*, 10(3-4):363–377.

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

Qin, H. and Terzopoulos, D. (1993). NURBS with Lagrangian dynamics (abstract). In *Proceedings of Third SIAM Conference on Geometric Design*, page 14, Tempe, Arizona. SIAM.

Qin, H. and Terzopoulos, D. (1994). Physics-based NURBS swung surfaces. In *Proceedings of IMA Conference on Mathematics of Surfaces VI*, London, UK. Oxford University Press. in press.

Qin, H. and Terzopoulos, D. (1995a). Dynamic manipulation of triangular B-splines. In *Proceedings of Third ACM/IEEE Symposium on Solid Modeling and Applications*, pages 351–360, Salt Lake City. ACM Press.

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

Qin, H. and Terzopoulos, D. (1995c). Triangular NURBS and dynamic generalizations. under review for *Computer Aided Geometric Design.*

Riesenfeld, R. (1973). *Applications of B-spline Approximation to Geometric Problems of Computer-aided Design.* PhD thesis, Syracuse University.

Rogers, D. and Adlum, L. (1990). Dynamic rational B-spline surfaces. *Computer-Aided Design*, 22(9):609–616.

Sapidis, N. (1992). Controlling the curvature of a quadratic bezier curve. *Computer Aided Geometric Design*, 9(2):85–91.

Sapidis, N. and Farin, G. (1990). Automatic fairing algorithm for B-spline curves. *Computer-Aided Design*, 22(2):121–129.

Sarkar, B. and Menq, C. (1991a). Parameter optimization in approximating curves and surfaces to measurement data. *Computer Aided Geometric Design*, 8(4):267–290.

Sarkar, B. and Menq, C.-H. (1991b). Smooth-surface approximation and reverse engineering. *Computer-Aided Design*, 23(9):623–628.

Schoenberg, I. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics*, 4:45–99.

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

Sederberg, T. (1990a). Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Applications*, 10(4):14–25.

Sederberg, T. (1990b). Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Application*, 10(5):12–21.

Sederberg, T., Gao, P., Wang, G., and Mu, H. (1993). 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.

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

Seidel, H.-P. (1992). Representing piecewise polynomials as linear combinations of multivariate B-splines. In Lyche, T. and Schumaker, L., editors, *Curves and Surfaces*, pages 559–566. Academic Press, New York.

Seidel, H.-P. (1993). An introduction to polar forms. *IEEE Computer Graphics and Applications*, 13(1):38–46.

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

Shirman, L. and Sequin, C. (1992). Procedural interpolation with geometrically continuous cubic splines. *Computer-Aided Design*, 24(5):267–277.

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

Strang, G. (1986). *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, MA.

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

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

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

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

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

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

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

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

Traas, C. (1990). Practice of bivariate simplicial splines. In et al, W. D., editor, *Computation of Curves and Surfaces*, pages 383–422. Kluwer Academic Publishers.

Versprille, K. (1975). *Computer-Aided Design Applications of the Rational B-Spline Approximation form.* PhD thesis, Syracuse University.

Wahba, G. (1990). *Spline Models for Observational Data.* SIAM, Philadelphia, PA.

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

Wever, U. (1988). Non-negative exponential. *Computer-Aided Design*, 20(1):11–16.

Wever, U. (1991). Optimal parameterization for cubic splines. *Computer-Aided Design*, 23(9):641–644.

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

Zienkiewicz, O. (1977). *The Finite Element Method.* McGraw-Hill, London, third edition.