Automatic Knot Determination of NURBS for Interactive Geometric Design

Hui Xie and Hong Qin
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400, USA
xhui@cs.sunysb.edu, qin@cs.sunysb.edu

Abstract

This paper presents a novel modeling technique and develops an interactive algorithm that facilitates the automatic determination of non-uniform knot vectors as well as other control variables for NURBS curves and surfaces through the unified methodology of energy minimization, variational principle, and numerical techniques. Many geometric algorithms have been developed for NURBS during the past three decades, Recently, the optimization principle has been widely studied, which affords designers to interactively manipulate NURBS via energy functionals, simulated forces, qualitative and quantitative constraints, etc. The existing techniques primarily concentrate on NURBS control points. In this paper, we further augment our NURBS modeling capabilities by incorporating NURBS' non-uniform knot sequence into our shape parameter set. The automatic determination of NURBS knots will facilitate the realization of the full geometric potential of NURBS. We also have developed a modeling framework which supports a large variety of functionals ranging from simple quadratic energy forms to non-linear curvature-based (or area-based) objective functionals.

Keywords: *NURBS, CAGD, Deformable Models, Constraints, Interactive Techniques, Energy Optimization.*

1 Introduction and Motivation

During the past two decades, Non-Uniform Rational B-Splines (NURBS) have gained popularity for shape modeling and geometric design and were incorporated into several commercial modeling systems [16] mainly because they have many attractive properties. NURBS offer a unified mathematical formulation for representing not only free-form curves and surfaces, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution. Through the manipulation of control points, weights, and/or knots, users can design a vast variety of shapes using NURBS. Despite NURBS' power and potential, users are faced with the tedium of non-intuitively manipulating a large number of geometric variables. Moreover, a particular shape can often be represented non-uniquely, with different values of knots, control points, and weights. The "geometric redundancy" of NURBS tends to make shape refinement *ad hoc* and ambiguous.

To ameliorate the geometric design with NURBS, A wide array of techniques for NURBS manipulation have been developed [6, 7, 16, 21]. Typical design techniques include interactive editing, (regular or scattered) data interpolation, shape approximation, cross-sectional design, optimization, etc. Recently, energy optimization techniques have been widely studied in shape modeling especially shape fairing. In a nutshell, energy-based algorithms offer designers a feasible and powerful solution that can alleviate the burden of interactively manipulating degrees of freedom (DOFs) of NURBS and a metric to evaluate the extent to which the final shape satisfies certain design requirements.

Prior work on energy optimization only focuses on functionals whose variables are either control points or non-unity weights of NURBS. The computation of functionals with respect to the additional shape flexibility resulted from the non-uniform knots is yet to be fully investigated.

Because the knot variation will generally violate the local support property of NURBS, this makes the direct evaluation of the gradient with respect to knots non-intuitive in principle. In our modeling algorithms, the NURBS geometry is systematically transformed into a set of equivalent rational Bezier patches [2]. This idea offers a new parametrization for the same NURBS shape, in which NURBS knots no longer affect the domain boundary for a specific curve/surface patch. Hence, the new formulation imposes no difficulties for the gradient derivation with respect to knots. Consequently, the geometric modeling potential of NURBS can be fully exploited in an intuitive and uniform fashion.

2 NURBS Geometry

First, we review the formulation of NURBS curves and surfaces. We then briefly describe their analytic and geometric properties.

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

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

where u is the parametric variable and $B_{i,k}(u)$'s are B-spline basis functions. Assuming basis functions of degree $k-1$, a NURBS curve has $n+k$ knots t_i in non-decreasing sequence: $t_1 \leq t_2 \leq \ldots \leq t_{n+k-1} \leq t_{n+k}$. The basis functions are defined recursively using non-uniform knots as

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

with

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

The parametric domain is $t_k \leq u \leq t_{n+1}$. From users' point of view, the NURBS knots are used to define B-splines' basis functions *implicitly*. In many applications, the end knots are repeated with multiplicity k in order to interpolate the first and last control points $\mathbf{p}_1$ and $\mathbf{p}_n$.

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

$$\mathbf{s}(u,v) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{p}_{ij} w_{ij} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} B_{i,k}(u) B_{j,l}(v)}, \quad (2)$$

where $B_{i,k}$ and $B_{j,l}$ are B-spline basis functions of degree $k-1$ and $l-1$, respectively.

A NURBS surface has $m \times n$ control points $\mathbf{p}_{ij}$ and weights w_{ij}. The number of knots is $(m+k)+(n+l)$. The non-decreasing knot sequence can be explicitly expressed as: $t_1 \leq t_2 \leq \ldots \leq t_{m+k-1} \leq t_{m+k}$ along the u-axis and $s_1 \leq s_2 \leq \ldots \leq s_{n+l-1} \leq s_{n+l}$ along the v-axis, respectively. The parametric domain is: $t_k \leq u \leq t_{m+1}$ and $s_l \leq v \leq s_{n+1}$. If all the end knots have multiplicity k and l in the u axis and v axis, respectively, the NURBS surface will interpolate the four corners of the boundary control points.

In the definitions to NURBS curves and surfaces, we can notice that only the relative positions of consecutive knots (i.e., knot intervals) are actually used to determine its geometry. The translation or scaling operations on knot vector only re-parameterize the same shape without perturbing its geometry.

NURBS generalize polynomial-based parametric representations for shape modeling. Analogous to B-splines, the rational basis functions of NURBS sum to unity, they are infinitely smooth in the interior of a knot interval provided the denominator is not zero, and at a knot they are at least C^{k-1-r} continuous with knot multiplicity r. They inherit many properties from B-splines, such as the strong convex hull property, variation diminishing property, local support, and invariance under affine geometric transformations.

Moreover, NURBS have additional properties. NURBS offer a unified mathematical framework for both implicit and parametric polynomial forms. In principle, they can represent **analytic** functions such as conics and quadrics precisely, as well as free-form shapes.

NURBS include weights as extra degrees of freedom which influence their local shape. NURBS are attracted toward a control point if the corresponding weight is increased and it is pushed away from a control point if the weight is decreased. If a weight is zero, the corresponding rational basis function is also zero and its control point does not affect the NURBS shape.

3 Modeling Techniques and Energy-based Tools

This section outlines the typical optimization approaches in geometric modeling and documents a set of popular energy functionals we developed, which are commonly used in a large variety of modeling and design applications.

3.1 Optimization Approach

Directly manipulating a variety of NURBS shape variables is non-intuitive and laborious. To ameliorate the geometric design with NURBS, researchers have been widely employing the energy optimization techniques in shape modeling, geometric design, and interactive graphics. In a nutshell, an energy optimization approach includes two steps: (1) formulating modeling requirements of the object shape in terms of energy functionals and (2) seeking a solution which minimizes a weighed combination of these energy functionals with appropriate optimization techniques. Optimization approach offers a unified way for enforcing both aesthetic criteria (e.g., fairness) and functional requirements (e.g., interpolation or continuity constraints). In general, user-specified energy functionals can be further decomposed into a combination of popular functional primitives.

Many mature NURBS modeling techniques such as interpolation and approximation are amenable to the computational framework of energy-based optimization. For example, interpolation algorithms such as cross-sectional design, skinning operation [8], and scattered data fitting [9,

5, 20, 1, 14, 15] can be considered as a set of geometric constraints that the final shape must satisfy. Typical approximation techniques for NURBS including approximated parametrization, lease-squares fitting, knot reduction, and hierarchical refinement [11, 19, 18, 10, 3, 4, 17]. can be expressed as the minimization of an error metric: $\epsilon(\mathbf{s})$.

3.2 Primitive Functionals

Currently, energy functionals in most CAD/CAM applications can be expressed in terms of a certain combination of functional primitives. A typical set of these basic functionals include the integral forms of parametric derivatives (up to order n), curvature, differential area, the variation of curvature, etc. Below are a set of commonly used primitives for NURBS surfaces. Functional primitives for NURBS curves can be similarly constructed.

- Simple quadratic forms (the subscript stands for partial derivatives with respect to parameters):

$$\int\int (c_1\,\mathbf{s}_u^2(u,v) + c_2\,\mathbf{s}_v^2(u,v))dudv$$

$$\int\int (c_1\,\mathbf{s}_{uu}^2(u,v) + c_2\,\mathbf{s}_{uv}^2(u,v) + c_3\,\mathbf{s}_{vv}^2(u,v))dudv$$

- Principal curvatures κ_1 and κ_2:

$$\int\int (c_1\,\kappa_1^2(\mathbf{s}(u,v)) + c_2\,\kappa_2^2(\mathbf{s}(u,v)))dudv$$

- Surface area function $A(\mathbf{s}(u,v))$:

$$\int\int A(\mathbf{s}(u,v))dudv$$

- Variation of two principal curvatures κ_1 and κ_2, where u_1 and u_2 represent u and v, respectively:

$$\int\int (\sum_{i,j}(\frac{\partial \kappa_i}{\partial u_j})^2(\mathbf{s}(u,v)))dudv$$

- Variation of surface area:

$$\int\int (\sum_{i}(\frac{\partial A}{\partial u_i})^2(\mathbf{s}(u,v)))dudv$$

- Functionals that enforce the area-preserving constraint, the convex-preserving constraint, etc.

- Torsion-based functionals.

The generic formulation for a constrained functional can be expressed as: minimizing $f(x)$ (where $x \in R^n$) subject to

$$\begin{cases} c_i(x) = 0, & i = 1,\ldots,k \\ c_i(x) >= 0, & i = k+1,\ldots,m \end{cases}$$

3.3 Least Motion and Uniform Distribution Constraints

NURBS are highly redundant in the sense that a specific shape may be represented by different combinations of shape control variables. This results in numerical unstability problem. Least motion constraint can be incorporated to ensure a unique solution. Essentially, the least motion constraint affords the change of the shape control variables to be as small as possible. A typical constraint for the least motion principle may be expressed as:

$$\sum_i \sigma_i (p_i^* - p_i)^2, \tag{3}$$

where superscripts $*$ denote optimal shape variables. The least motion constraint can be physically charaterized as that each generalized coordinate in the DOF vector is virtually connected with its initial position through a spring whose rest length is zero. The springs' stiffness distribution controls different weights for each entity within the sum of squared distance.

Another useful constraint to ensure numerical stability is the regularity constraint which requires the knots and/or control points uniformly distributed when there is a continuous set of local minima (for example, straight line may be represented by arbitrary vertices on a straight line), please refer to [12] for details.

The underlying function space over which our functionals are defined is limited to NURBS space. Thus, energy functionals over infinite dimensional shape space become functions of finite dimensional shape control variables:

$$\lambda[s(u,v,p_1,\cdots,p_{n+k})] = L(p_1,\cdots,p_{n+k})$$

where u,v are eliminated by the functional operator λ.

4 Computation of NURBS' Gradient

The key to most optimization techniques is to effectively derive the gradient of certain functionals and seek the accurate solution of zero-gradient for the corresponding functionals. This section details the transformation from NURBS to a set of rational Bezier splines that aims to facilitate the gradient computation of energy functionals.

4.1 Gradient Calculation

Our optimization algorithm is based on Polak-Ribiere's Conjugate Gradient (CG) method [22] which computes the gradient information in order to speed up the convergence. We now address the concept of re-parametrization and its effects on NURBS knots.

To simplify notation, we decompose the NURBS DOFs into three separate vectors:

$$\mathbf{p}^b = \{p_{id}\}_{i=1..n,d=1..3} = \left[\begin{array}{ccc} \mathbf{p}_1^\top & \cdots & \mathbf{p}_n^\top \end{array}\right]^\top,$$
$$\mathbf{p}^w = \{w_i\}_{i=1..n} = \left[\begin{array}{ccc} w_1 & \cdots & w_n \end{array}\right]^\top,$$
$$\mathbf{p}^k = \{t_i\}_{i=1..n+k} = \left[\begin{array}{ccc} t_1 & \cdots & t_{n+k} \end{array}\right]^\top,$$

where subscript d differs the x, y, and z coordinates for each control point. Let us collect all DOFs into a single vector:

$$\mathbf{p} = \{p_i\}_{i=1..5n+k} =$$

$$[\{p_{jd}\}_{j=1..n,d=1..3}\ ,\ \{w_j\}_{j=1..n}\ ,\ \{t_j\}_{j=1..n+k}], \quad (4)$$

The gradient of any generic objective funtion L is now expressed as:

$$\mathbf{g} = \frac{\partial L(\mathbf{p}^b, \mathbf{p}^w, \mathbf{p}^k)}{\partial \mathbf{p}} = \left[\begin{array}{ccc} \partial L/\mathbf{p}^b & \partial L/\partial \mathbf{p}^w & \partial L/\partial \mathbf{p}^k \end{array}\right]$$

It is trivial to evaluate the gradients with respect to NURBS control points and weights. However, the gradient evaluation with respect to NURBS knots is rather challenging. One desirable advantage of our NURBS geometry is its local control property which allows better editing capability. Arbitrary point $\mathbf{c}(u)$, $t_i < u < t_{i+1}$ is only related to a subset of the knot vector $t_{i-k+1}, \cdots, t_{i+k}$ (see Fig. 2). We move the knot t_i slowly towards its right neighbor and pass the u, now $\mathbf{c}(u)$ jumps to the adjacent span. In this case, the relevant control polygon that defines $\mathbf{c}(u)$ is modified. This leads problems in certain applications. For instance, moving a curve point may not result in a strong change of the shape, but the point sliding to its nearby patch along the curve tangent. The point shift from one span to its neighboring span resulted from the varying knots is counter-intuitive. It is our hope that each point on NURBS is only a function of a fixed subset of its control polygon.

4.2 Curve Reparametrization

Note that, because NURBS are a homogeneous representation of four-dimensional B-splines, we first describe our procedure for B-splines. The underlying concept can be trivially extended to NURBS using homogeneous transformation.

Consider each knot interval $t_i \le u < t_{i+1}$, the curve span $\mathbf{c}(u)$ is uniquely determined by a subset of control parameters mentioned above. We can decompose a B-spline curve as a set of B-spline spans defined on the domain of a single knot interval (i.e., every two consecutive knots) and formulate the parametric representation for each span. We normalize the parameter domain for each span by scaling and translating the domain $[t_i, t_{i+1}]$ for span i to $[i, i+1]$. Thus we obtain a new parametrization scheme, for the i^{th} B-spline span, i.e., $\mathbf{c}(u),\ \ t_i \le u < t_{i+1}$:

$$\mathbf{c}_1(\hat{u}) = \mathbf{c}(u) = \mathbf{c}(t_i + (\hat{u} - i)(t_{i+1} - t_i)), \quad (5)$$

where $i \le \hat{u} < i+1$, and $\hat{u}$ is the new parameter. Concatenating all of these curve spans together, we derive a new parameterization in domain $[k, n+1]$ for the same B-spline curve $\mathbf{c}(u), t_k \le u \le t_{n+1}$ (see Fig. 1). Moreover, we can derive the explicit transformation, $u = t_i + (\hat{u} - i)(t_{i+1} - t_i)$. The shape is preserved with the normalized parameter domain for each B-spline span. However, we lose the C^{k-1-r} continuity at integer value on the parametric domain. but G^{k-1-r} continuity still holds.

We know a B-spline curve span $\mathbf{c}(u),\ t_i \le u < t_{i+1}$ determined by knots $t_{i-k+1}, \cdots, t_{i+k}$ and control polygon $\mathbf{p}_{i-k+1}, \cdots, \mathbf{p}_i$ is a Bezier curve of the same degree, refer to [2] for details. The control polygon of this Bezier curve can be expressed in terms of the preceding B-spline control polygon and its relevant knots.

Note that, all the Bezier basis functions are defined over a fixed domain $[0, 1]$. This leads to another reparametrization scheme which offers each curve span a fixed boundary independent of knots changing. Here, non-uniform knots are used to derive the new Bezier control points. With Bezier representation, it is no longer necessary to re-evaluate B-spline basis functions after the modification of knots. In general, this can significantly improve time performance because all Bezier basis functions can be precomputed with appropriate sampling density in parameter domain $[0..1]$.

We now use cubic B-splines to briefly illustrate the transformation technique from B-splines to a set of piecewise Bezier curves. Detailed derivation can be found in Boehm's paper [2]. In Fig. 2, Control polygon $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ and knot sequence $t_1, \cdots, t_8$, determines curve span AD(solid) defined on parametric domain $t_4 \le u < t_5$. Control polygon $\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$ and knot sequence $t_2, \cdots, t_9$ determines curve spanDG (dash) defined on parametric domain $t_5 \le u < t_6$. In this example, the Bezier control points of curve AD and DG are $ABCD$ and $DEFG$, respectively. The Bezier control polygon $ABCD$ and $DEFG$ can be derived by dividing each edge using certain ratios as shown in Fig. 2, where $u_2, \cdots, u_6$ are knot intervals, i.e., $u_i = t_{i+1} - t_i$.

Now, let us focus on the derivation of Bezier parametrization for the B-spline curve span $\mathbf{c}(u),\ t_i \le u < t_{i+1}$. Each B-spline point is associated with a corresponding local Bezier parameter, denoted as $u' \in [0, 1]$. We introduce Bezier global parameter as $\check{u} = i + u'$, where u' is the local Bezier parameter, and i is the index of the span where the point resides.

The cubic B-spline curve span $\mathbf{c}(u)$ is equivalent to a cubic Bezier curve $\mathbf{c}'(u')$. For the first span, $\mathbf{c}(u),\ t_4 \le u < t_5$, we have,

$$\mathbf{c}(u) = \sum_{i=1}^{4} \mathbf{p}_i B_{i,4}(u) = \sum_{i=1}^{4} \mathbf{p}'_i N_{i,4}(u'),$$

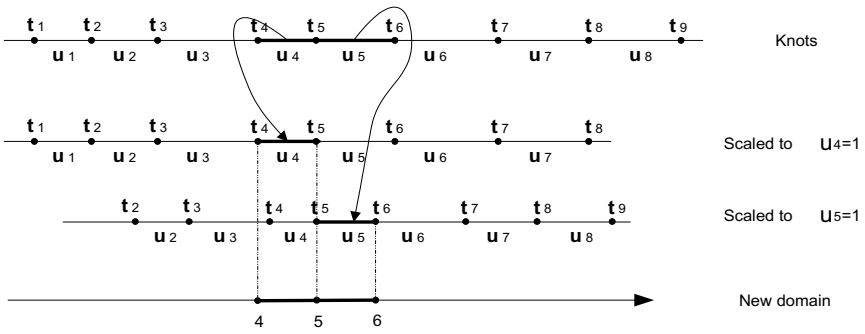

Figure 1. Uniformly spaced B-spline curve spans.

where each $\mathbf{p}'_i$ is a control point for the corresponding Bezier form. Each $N_{i,4}$ is a cubic Bezier basis function whose domain is $[0, 1]$, and u' is a local parameter for the Bezier curve. Bezier control points are algebraic functions of B-spline controls points and their associated knot.

$$\begin{bmatrix} \mathbf{p}'_1{}^\top \\ \mathbf{p}'_2{}^\top \\ \mathbf{p}'_3{}^\top \\ \mathbf{p}'_4{}^\top \end{bmatrix} = \begin{bmatrix} \frac{t_5-t_4}{t_5-t_3} & \frac{t_4-t_3}{t_5-t_3} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{t_6-t_5}{t_6-t_4} & \frac{t_5-t_4}{t_6-t_4} \end{bmatrix}.$$

$$\begin{bmatrix} \frac{t_5-t_4}{t_5-t_2} & \frac{t_4-t_2}{t_5-t_2} & 0 & 0 \\ 0 & \frac{t_6-t_4}{t_6-t_3} & \frac{t_4-t_3}{t_6-t_3} & 0 \\ 0 & \frac{t_6-t_5}{t_6-t_3} & \frac{t_5-t_3}{t_6-t_3} & 0 \\ 0 & 0 & \frac{t_6-t_5}{t_6-t_4} & \frac{t_5-t_4}{t_6-t_4} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1{}^\top \\ \mathbf{p}_2{}^\top \\ \mathbf{p}_3{}^\top \\ \mathbf{p}_4{}^\top \end{bmatrix}$$

Denote the composite transformation matrix of the matrices in the above equation as $\mathbf{M}$. In general, for arbitrary B-spline curve span, it is a function of its corresponding subset of the knot vector.

$$\mathbf{c}(\breve{u}, \mathbf{p}) = [N_{1,k}(u'), \cdots, N_{k,k}(u')]\, \mathbf{M}(t_{i-k+1}, \cdots, t_{i+k}) \cdot \left[\mathbf{p}_{i-k+1}^\top, \cdots, \mathbf{p}_i^\top\right]^\top, \quad (6)$$

where, i is the span index. For simplicity, let us write the right side of this formula as: $\mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\mathbf{P}$ or simply $\mathbf{NMP}$. Note that, for each curve span, there is a different $\mathbf{M}$. Now we can calculate the gradient with respect to knots.

$$\frac{\partial \mathbf{c}(\breve{u}, \mathbf{p})}{\partial \mathbf{p}^k} = N(u') \frac{\partial \mathbf{M}}{\partial \mathbf{p}^k} \mathbf{P}$$

It may be noted that the close-form analytic expression of $\partial \mathbf{M}/\partial \mathbf{p}^k$ can be derived though the exact form is extremely complicated. As four-dimensional homogeneous B-spline, NURBS curve can be converted into a set of rational Bezier curves with the same geometry:

$$\mathbf{c}(u) = \frac{\sum_{j=0}^{n} \mathbf{p}_j w_j B_{j,k}(u)}{\sum_{j=0}^{n} \mathbf{w}_j B_{j,k}(u)} = \frac{\mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\{w_i \mathbf{p}_i\}}{\mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\{w_i\}}$$

4.3 Surface Reparametrization

Reparametrization of a B-spline surface can be similarly derived, following the prior discussion on the curve procedure. For a NURBS surface:

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

Again, the generalized coordinates of NURBS surface consist of the control points, weights, and two knot sequences which can be assembled into vectors $\mathbf{p}^b$, $\mathbf{p}^w$, $\mathbf{p}^s$ and $\mathbf{p}^t$, respectively. Finally, we assemble them into a single generalized coordinate vector $\mathbf{p}$.

Similar to a NURBS curve, a NURBS surface consists a set of Bezier surface patches. To simplify the complicated subscripts for the tensor product NURBS surface, we use individual matrix entity instead of the previous matrix form in our derivation. We denote each element of matrix $\mathbf{M}$ in (6) as $M_{ij}^k(\{t_i\})$ where $i, j \in [1..k]$. Let u' and v' be the corresponding Bezier parameters for u and v.

$$\mathbf{s}(u, v, \mathbf{p}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{p}_{ij} w_{ij} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} B_{i,k}(u) B_{j,l}(v)} =$$

$$\frac{\sum_{i=1}^{k} \sum_{i'=1}^{k} N_{i'}^k(u') M_{i'i}^k (\sum_{j=1}^{l} \sum_{j'=1}^{l} N_{j'}^l(v') M_{j'j}^l w_{ij} \mathbf{p}_{ij})}{\sum_{i=1}^{k} \sum_{i'=1}^{k} N_{i'}^k(u') M_{i'i}^k (\sum_{j=1}^{l} \sum_{j'=1}^{l} N_{j'}^l(v') M_{j'j}^l w_{ij})} \quad (8)$$

where $N_{i'}^k$ and $N_{j'}^l$ are the Bezier basis functions of degree k and l, respectively, $M_{i'i}^k$ and $M_{j'j}^l$ are the matrices transforming B-spline control points to Bezier control points for parameter u and v, respectively, Both M's are square matrices whose entities are functions of the relevant knots for a specific Bezier patch.

5 Numerical Techniques

This section addresses the numerical techniques used in our modeling software. In particular, we have developed a

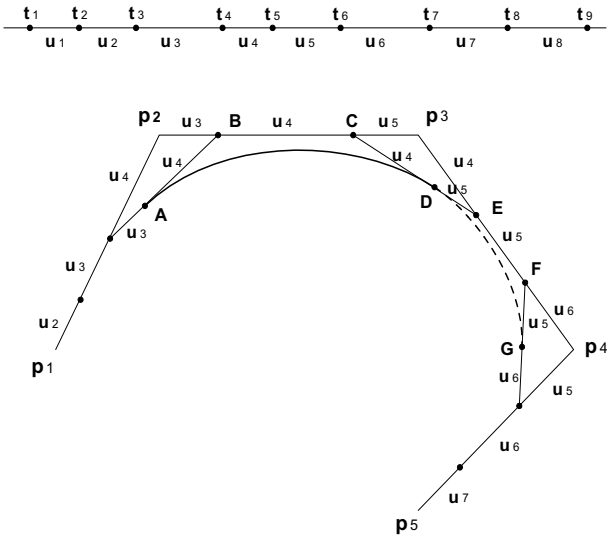

Figure 2. Conversion of a cubic B-spline curve to a set of Bezier curves.

large variety of numerical algorithms for efficient function evaluation, derivative evaluation, and energy optimization.

5.1 Functionals and gradient Evaluation

In general, allowing NURBS knots to vary makes NURBS evaluation more time-consuming. We consider the following numerical aspects:

- **NURBS Evaluation**. Through reparametrization, arbitrary NURBS curve is transformed to a set of Bezier curves. Bezier basis functions are Bernstein polynomials with fixed domain $[0, 1]$, we employ the preprocessing to store all relevant values of Bezier basis functions in a lookup table in order to speed up the function evaluation. So, the major run-time computational cost for NURBS evaluation is due to the calculation of Bezier control polygon and their weights.

- **Functional Evaluation**. Functional evaluation mainly depends on the effective computation of NURBS derivatives (up to order n). Because NURBS are based on homogeneous coordinates, their higher-order derivatives require many multiplication and division operations. In our system, only the NURBS value and its first-order derivative are computed analytically. However, NURBS area, second-order derivative, curvature, as well as the variation of curvature should be computed numerically due to their complexities. For NURBS derivatives with respect to knot vector, we also turn to numerical approaches. We implemented a wide range of commonly-used functionals as software modules. Users can concentrate on a meaningful combination of system-supplied functional procedures in order to achieve their design objectives.

- **Numerical Integration**. Continuous functionals requires function integration. The closed-form analytic solution for the integral of arbitrary functions of NURBS and their derivatives is almost impossible. In our system, in particular, we take advantages of different numerical methods for integration which are applicable to different cases. Among them, Gaussian quadrature [22] is the most accurate and efficient technique with few sampling points for most functionals available in our system. Nevertheless, Gaussian quadrature will be much less appealing to users when certain singularities such as dis-continuity at NURBS knots happen. This scenario may be caused by multiple knots concentrated in a very small region. In a nutshell, Gaussian quadratures are sampled in the interior of knot intervals. Function values at two end points of any knot interval are not taken into consideration during the numerical integration. In such cases, we use Simpson's quadrature instead, which also samples the boundary value. Fig. 3 shows a situation where Gaussian quadrature fails to offer a good approximation for the integral computation of arc-length. The solid dots on axis x are sampling points of Gaussian quadrature.

5.2 Optimization Techniques

Mathematicians have studied a large variety of approaches to solving optimization problems of various categories. The energy optimization problems of NURBS can be classified as multi-dimensional non-linear constrained

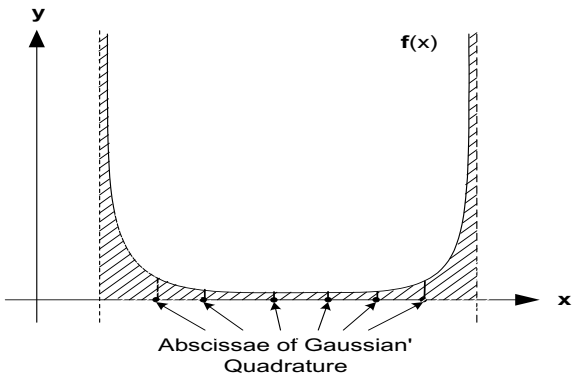

Figure 3. Gaussian quadrature fails to offer a good solution when the underlying function becomes singular at the vicinity of two boundary points.

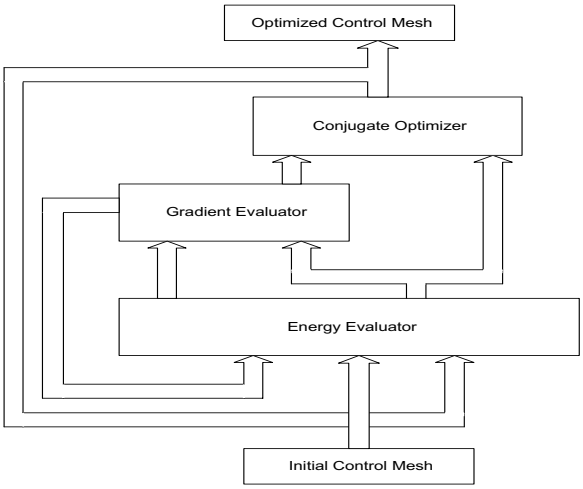

Figure 4. The system architecture of our energy-based optimization procedure.

optimization. In [13], many methods for this type of problems are analyzed in detail.

Our optimization algorithm is based on Polak-Ribiere's Conjugate Gradient (CG) method [22]. In the process of optimization, there are several hard constraints that NURBS must satisfy. First, the knot sequence must be non-decreasing: $t_1 \leq t_2 \leq \cdots \leq t_{n+k}$. Second, all the weights have to be non-negative. We modify the CG algorithm to accomodate these hard constraints.

At each CG iteration, a new direction based on the gradient is generated, and the optimizer sequentially invokes a line minimizer to obtain the local minimum along that direction. To enforce the hard constraint on knots, we first adjust the direction vector with respect to knots. If several adjacent knots which are very near to each other intend to collide during the numerical procedure, we modify the direction component of them to their average. A rigorous analysis of this technique can be derived through the use of Langrangian multiplier method. After the direction is adjusted, we set a upper-bound that specifies how far the line-minimizer can proceed. To enforce the non-decreasing constraint for NURBS knots, we simply set a minimum value so that the distance between two adjacent knots must not fall below this threshold. Assuming the new adjusted direction is $\mathbf{d} = \{d_i\}$, and the knot vector is $\mathbf{P}^k = \{t_1, t_2, \cdots, t_{n+k}\}$, we will have to explicitly limit the depth along this direction as:

$$min\{\ (t_{i+1} - t_i - \epsilon)/(d_i - d_{i+1})\ \}_{(d_i - d_{i+1})>0}$$

in order to enforce the geometric constraints for knots and weights. Similarly, If a weight value becomes very close to zero and the corresponding gradient with respect to this weight is negative.

6 System Organization

The major components of our numerical algorithms for energy-based tools in our system are briefly documented in Fig.4. The entire system functions through the use of several temporal events:

1. Convert a NURBS curve/surface into Bezier patches.
2. Evaluate the scalar functional to be optimized.
3. Numerically evaluate the gradient of this functional by repeatedly applying Step 2 with perturbed shape variables (the number of evaluations depending on the required accuracy).
4. Employ appropriate multi-dimensional optimization techniques such as Conjugate Gradient, or Newton's method to evolve the shape.
5. Repeat the above procedures till the error bound are satisfied.

7 Results and Discussion

We provide users an interactive shape modeling and editing system, which can enforce various constraints for NURBS curves and surfaces in real-time. In the interest of space, this section explains several application examples only for NURBS surface design. We use a 6×6 control mesh for all the following examples. Without the loss of generality, the boundary control points and end knots are fixed for these examples in order to emphasize our novel tools and its associated ideas.

- **Inteactively specify a surface point and manipulate this surface point directly to arbitrary location in**

knots u	0.0000	0.0000	0.0000	0.0000	1.0000	2.0000	3.0000	3.0000	3.0000	3.0000
knots v	0.0000	0.0000	0.0000	0.0000	1.0000	2.0000	3.0000	3.0000	3.0000	3.0000

Figure 5. The two knot vectors for the NURBS surface appeared in Fig.7(a).

knots u	0.0000	0.0000	0.0000	0.0000	0.4513	2.5487	3.0000	3.0000	3.0000	3.0000
knots v	0.0000	0.0000	0.0000	0.0000	0.4513	2.5487	3.0000	3.0000	3.0000	3.0000

Figure 6. The two knot vectors for the NURBS surface appeared in Fig.7(g).

3-space.
In Fig. 7, (a) and (b) show two NURBS surfaces before and after the manipulation of a user-specified point, respectively. Our system allows users to pick any point (pointed with arrow in the figures) across the entire NURBS surface and edit its location, normal, and curvature.

- **Minimize Guassian and mean curvatures at a user-specified point on the surface**.
  Fig.7(c) and Fig.7(d) show two NURBS surface shapes before and after minimizing the Gaussian curvature at the user-specified point (pointed by arrow), respectively. Fig.7(e) and Fig.7(f) demonstrate the curvature maps, of Fig.7(c) and Fig.7(d) respectively, where the gray color denotes larger Gaussian curvature, the light color represents larger mean curvature, dark color for smaller value of both Gaussian and mean curvatures.

- **Minimize the curvature variation**.
  Fig.7(g) shows the sculpting tool that can minimize the variation of curvature throughout the NURBS area. The original shape without the minimization of this energy functional is the same as that in Fig.7(a). We fix four inner control points near the top (represented as grey points). Note that, sometimes fixing a subset of control mesh can be used to specify a rough shape as an effective initialization process. Other free control parameters will be determined through the interactive specification of appropriate energy functionals. Fig.7(h) and Fig.7(i) demonstrate the corresponding curvature maps. We can clearly see the curvature contrast across different regions of the NURBS surface. Note we rotate the shape by an angle in the curvature maps. Fig.5 and Fig.6 document the knot vectors before and after the minimization of the variaton of curvature for this example.

- **Enforce the area-preserving constraint on any specified region of interest**.
  Fig.7(j) shows the result of changing the area value in Fig.7(a) to a new value subject to the constraint of least motion. Note that, the control points near the center aggregate towards each other as this type of motions will change the area of NURBS surface significantly in this example.

- **Render the NURBS denominator distribution as a texture map on the NURBS geometry**.
  Besides knots, our system also supports the automatic determination of time-varying weights. To best demonstrate this functionality, we use the NURBS denominator distribution as a texture map, or weight map. Fig.7(k) and Fig.7(l) are two weight maps that show the two sets of non-unity weights and their variation due to the surface point manipulation. In this example, the dard color stands for larger NURBS denominator.

8 Conclusion

We have greatly enhanced the already-powerful modeling capabilities of NURBS through the energy-based optimization approach by incorporating non-uniform knots of NURBS into NURBS generalized coordinates. Our key contribution is that we have developed a novel modeling technique that systematically transforms general NURBS geometry (including both univariate curves and tensor-product surfaces) into a set of geometrically equivalent rational Bezier splines in order to facilitate the mathematical derivation of analytic formulation for NURBS Jacobian matrix through symbolic computation. The new, improved NURBS formulation and its modeling framework based on the principle of energy optimization afford all time-varying degrees of freedom of NURBS to be controlled by various commonly-used energy functionals. Within the framework of energy optimization, the system can allow users to interactively manipulate NURBS geometry in an intuitive fashion via a large variety of sculpting tools (e.g., geometric constraints, energy functionals) without worrying about how to set up control points, non-unity weights, and/or non-uniform knots.

We have developed a prototype interactive modeling system based on the new NURBS formulation and have demonstrated the flexibility of our models in a variety of

applications. In particular, we have extended our previous-developed NURBS system by offering users a wide array of non-quadratic curvature-based functionals that can be dynamically minimized during the sculpting session in real-time. Our novel formulation as well as its accompanying system permits NURBS to realize its full modeling potential in shape modeling, geometric design, and interactive graphics, greatly enhancing NURBS functionalities in various visual computing applications.

9 Acknowledgments

This research is supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, and a research grant from Ford Motor Company.

References

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

[2] W. Boehm. Generating the bezier points of b-spline curves and surfaces. *Computer-Aided Design*, 13(6):365 – 366, 1981.

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

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

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

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

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

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

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

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

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

[12] P. Laurent-Gengoux and M. Mekhilef. Optimization of a nurbs representation. *Computer-Aided Design*, 25(11):699–710, 1993.

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

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

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

[16] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, New York, NY, second edition, 1997.

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

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

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

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

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

[22] W. T. V. Willian H. Press, Saul A. Teukolsky and B. P. Flannery. *Nemerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition edition, 1992.

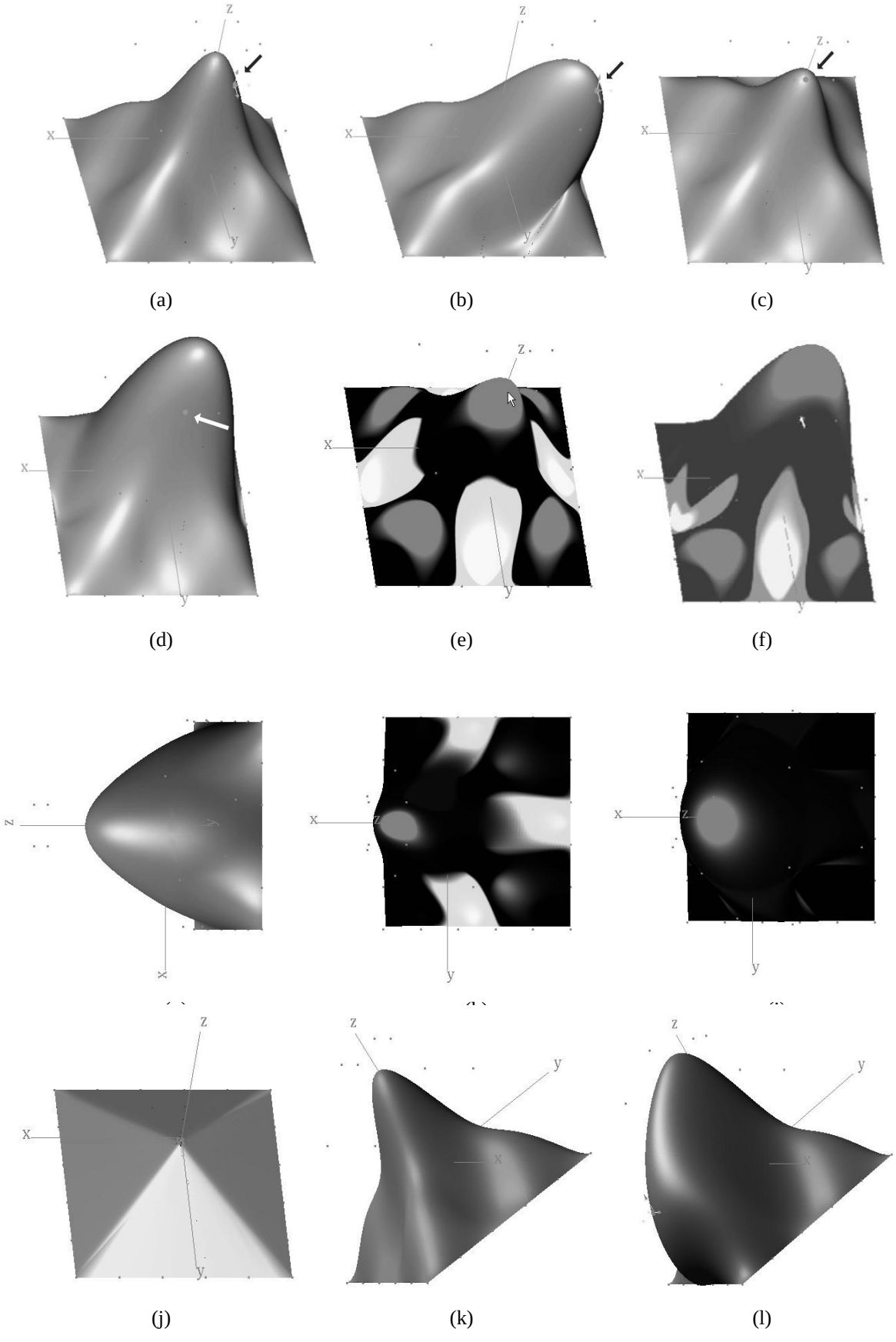

Figure 7. Result Snapshots