International Journal of Shape Modeling

A NOVEL OPTIMIZATION APPROACH TO THE EFFECTIVE COMPUTATION OF NURBS KNOTS

HUI XIE
Department of Computer Science, State University of New York at Stony Brook
Stony Brook, New York 11794/4400, United States of America

and

HONG QIN
Department of Computer Science, State University of New York at Stony Brook
Stony Brook, New York 11794/4400, United States of America

Received (Aug, 2001)
Revised (Nov, 2001)
Communicated by (Alexander Pasko)

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. NURBS have become a *de facto* industry-standard primarily because of their power to represent free-form shapes as well as commonly-used analytic shapes. Although many geometric algorithms have been developed for NURBS, existing techniques primarily concentrate on NURBS control points. 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 key advantage of energy-based approaches is that they can evolve both the control points and the weights in response to NURBS deformation resulted from the numerical optimization of a set of energy functionals. These energy functionals have the capability to quantify user-centered aesthetic criteria, qualitative constraints, and functional requirements for a large variety of applications in a unified fashion. In this paper, we further augment our NURBS modeling capabilities by incorporating NURBS' non-uniform knot sequence into our shape parameter set intuitively controlled by energy functionals. We also have implemented a software environment that supports a large variety of functionals ranging from simple quadratic energy forms to non-linear curvature-based objective functionals.

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

1. Introduction and Motivation

In 1975, Versprille [24] proposed the Non-Uniform Rational B-Splines or NURBS. This shape modeling representation for geometric design generalized Riesenfeld's B-splines. NURBS quickly gained popularity and were incorporated into several

commercial modeling systems [16] primarily 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. NURBS are very flexible and powerful because of their large number of control variables (or degrees of freedom) which comprise control points, non-unity weights, and non-uniform knot sequence. In essence, through the manipulation of control points, weights, and/or knots, users can design a vast variety of shapes using NURBS. A wide array of techniques for NURBS have been developed for geometric design [5, 6, 17, 14, 15, 16, 23]. Typical design techniques include interactive editing, (regular or scattered) data interpolation, shape approximation, cross-sectional design, optimization, etc. Despite the diversity of NURBS-based modeling techniques, current state-of-the-art mainly concentrates on NURBS control points and their variations as well as the quantified effects of control points on NURBS. Existing algorithms may have some modeling difficulties:

- Users are oftentimes 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. In principle, indirect geometric design can be clumsy and laborious.

- A particular shape can often be represented non-uniquely, with different values of knots, control points, and weights. For instance, the "geometric redundancy" of NURBS tends to make shape refinement *ad hoc* and ambiguous; it often requires designers to make nonintuitive decisions: to adjust a shape, should the designer move a control point, change a weight, move two control points, or adjust several knots?

- The design requirements of engineers and stylists can be different. Whereas engineers focus on technical and functional issues, stylists emphasize aesthetically-driven conceptual design. Thus, typical design requirements may be posed in both quantitative and qualitative terms. Therefore, it can be very frustrating to design via the indirect approach, say, a "fair" surface that approximates unorganized 3D data.

To ameliorate the geometric design with NURBS, researchers have been widely employing the energy optimization technique in shape modeling, geometric design, and interactive graphics. 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. The key of energy-based techniques is to effectively formulate the designer's modeling requirements in terms of energy functionals and to seek the accurate solution which minimizes the corresponding functionals.

Mature NURBS modeling techniques such as interpolation and approximation are amenable to the computational framework of energy-based optimization. For ex-

ample, interpolation requirements on both discretized points and (isoparametric/non-isoparametric) curve network of NURBS can be considered as a set of geometric constraints that the final shape must satisfy. More sophisticated interpolation algorithms support cross-sectional design, skinning operation [7], and scattered data fitting [8, 4, 22, 1, 12, 13]. When the number of DOFs for NURBS is less than that of the geometric constraints, approximation techniques must be applied instead. In general, shape approximation has several advantages such as removing data redundancy, reducing data size, supporting data exchange, and fostering hierarchical modeling [19]. Typical approximation techniques for NURBS include approximated parameterization, least-squares fitting, knot reduction, and hierarchical refinement [10, 21, 20, 9, 2, 3, 18]. In principle, arbitrary approximation requirements can be expressed as the minimization of an error metric: $\epsilon(\mathbf{s})$ defined over $\mathbf{s}$ which is a NURBS curve/surface.

Nonetheless, prior techniques on energy optimization only focus on the optimal solution of relevant functionals whose variables are either control points or non-unity weights of NURBS. In principle, the gradient computation of functionals with respect to control points and weights poses no severe difficulties primarily because NURBS are a rational combination of these variables. However, the extra shape flexibility resulted from the non-uniform knots is yet to be fully investigated.

We generalize the prior approaches and develop novel algorithms that support the automatic determination of NURBS knots in terms of many popular well-behaved functionals such as the minimization of curvature and/or the variation of curvature. The core technology of our modeling algorithms is that we systematically transform the NURBS geometry into a set of equivalent rational Bezier patches in which NURBS knots become transparent to all the Bezier basis functions. Subsequently, NURBS knots are used to derive the new control points as well as the new weights of rational Bezier patches. This idea results from our following observation on NURBS. Unlike (polynomial and rational) Bezier splines, all the basis functions of NURBS are defined in terms of their non-uniform knots. The parametric domain of each basis function are directly associated with these knots. Changing NURBS knot vector will inevitably alter the parametric domain of each basis function. This presents a grand challenge to compute the NURBS Jacobian matrix [19] with respect to the knots either numerically or symbolically. The aforementioned transformation can greatly ameliorate the computation for NURBS Jacobian matrix, hence facilitating the energy optimization technique that provides the accurate solution for unknown knots through the numerical derivation of (oftentimes) non-linear energy functionals. We have developed cost-effective numerical algorithms to support the energy optimization process that determines the final, desirable shape of NURBS. In addition, we have implemented a prototype, scalable software system equipped with a large variety of energy-based constraints and functionals.

The remainder of this paper is structured as follows. Section briefly reviews NURBS geometry and its properties. In Section , we discuss the variational optimization method as well as outline a number of commonly-used functionals that

our system supports. Section presents our novel formulation and algorithm that incorporates the knot vector into the optimization framework towards the realization of NURBS' representation potential. In Section , we discuss the relevant numerical techniques. A brief description of our system architecture and application examples are illustrated in Section and Section , respectively. Finally, we conclude the paper in Section .

2. NURBS Geometry

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

2.1. *Curves*

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

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

where u is the parametric variable and $B_{i,k}(u)$ are B-spline basis functions. Assuming basis functions of degree $k-1$, a NURBS curve has $n+k$ 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-spline basis functions *implicitly*. Note that, re-parameterization will not change the shape of a NURBS curve, therefore, we can normalize the knot vector: $t_1 = 0$ and $t_{n+k} = 1$ without loss of generality. In addition, we can define knot intervals: $u_i = t_{i+1} - t_i$ and observe that all B-spline basis functions are in fact functions of u_i's because only the relative position of knots is used to formulate basis functions. 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 $\mathbf{p}_n$.

2.2. *Surfaces*

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

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

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 $s_l \leq v \leq s_{n+1}$. Based on the same rationale explained above, a re-parameterization process will result in $t_1 = 0$, $t_{n+k} = 1$; and $s_1 = 0$, $s_{m+l} = 1$. Furthermore, $B_{i,k}(u)$'s and $B_{j,l}(v)$'s are functions of knot intervals u_i's and v_j's, respectively, where $u_i = t_{i+1} - t_i$ and $v_j = s_{j+1} - s_j$. 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.

2.3. *Properties*

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. This enables NURBS to satisfy different smoothness requirements. They inherit many properties from B-splines, such as the strong convex hull property, variation diminishing property, local control, and invariance under standard geometric transformations. Moreover, they have additional important 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. This powerful modeling flexibility is achieved through the specific combinations of control points, weights, and knots.
- 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 presents typical modeling techniques for NURBS and details a set of popular energy functionals commonly used in a large variety of modeling and design applications. These functionals have been implemented as powerful design tools in our current environment.

3.1. *Optimization*

Most frequently used NURBS design techniques are the specification of a control point, or interpolation/approximation of a set of data points to generate an initial

shape. The initial shape can then be refined into the final desired shape through the interactive adjustment of control points, weights, and possibly the insertion or deletion of knots. This refinement process is *ad hoc* and oftentimes time-consuming in general. Other available techniques include the specification of cross-sectional profiles such as trimming, extruding, and filleting; as well as surface fitting of a network of boundary curves.

Existing techniques are unable to realize the full modeling potential of NURBS because all the control variables have not been handled collectively and simultaneously in a unified way. For example, a straightforward question is: can certain analytic shapes be precisely reconstructed from a set of (dense or scattered) sample data of the prescribed shape using the available modeling tools? This question is of great importance in reverse engineering. We shall resort to the optimization process and incorporate knots (as well as control points and weights) into the generalized coordinates of NURBS to be determined automatically through the effective numerical simulation of energy functionals. Using the optimization technique, users can transform both aesthetic criteria (e.g., fairness) and functional requirements (e.g., interpolation or continuity constraints) to a certain unified form of energy functionals which can be further decomposed to a weighted linear combination of functional primitives. A typical set of basic functionals include the integral forms of parametric derivatives (up to order n), surface normal, curvature, differential area, the variation of curvature, etc. For instance, the linear, thin-plate under tension elastic energy [19] is no longer applicable to recover standard analytic shapes such as spheres and tori from a sample dataset. Instead, a functional that involves the variation of curvature and its integration is more appropriate because its global minimum results in a circular shape of the modeled object (whose curvature is constant). To make our design tools more attractive to the CAD/CAM industry, we have developed many energy-based tools whose functionals may take a wide range of forms to cover various design scenarios. Our variational technique is both flexible and powerful as it can easily satisfy qualitative and subjective criteria as well as quantitative requirements. In essence, the optimization approach provides the designer extra flexibility to enforce hard geometric constraints. The optimization method can accommodate diverse (sometimes conflicting) requirements from both engineers and stylists. Therefore, this new framework is universally applicable to a wide range of applications without the need to change its underlying architecture. Through the optimization process, the imposition of modeling criteria automatically evolve the shape of NURBS, and subsequently derive the optimal solution of all the unknown control variables. This methodology permits users an intuitive control on and natural interaction with NURBS, because all the design criteria can be expressed as a set of local and/or global energy functionals. Additional shape deformation can be achieved through the interactive manipulation of individual DOF of NURBS. This capability is especially valuable during the shape initialization phase. When a functional is non-quadratic, its gradient becomes a non-linear function. In this case, the initial guess is critical to reach the final desirable result, because only

the local minimum is guaranteed. Moreover, many complicated modeling criteria must be satisfied in order to achieve a desired shape. This leads to a properly weighted combination of different energy functionals.

3.2. *Curve Functionals*

Our system is general in the sense that it supports the computation of a diverse set of energy functionals and their linear combinations. The fundamental set of functional primitives in our system consists of various energy-based tools commonly used in CAD/CAM:

- Simple quadratic forms (i.e., derivatives up to order n):
$$\int_a^b (\mathbf{c}^{(i)}(u))^2 du$$
- Curvature integration:
$$\int_a^b \kappa^2[\mathbf{c}(u)]du$$
- Arc-length integration:
$$\int_a^b l[\mathbf{c}(u)]du$$
- Variation of curvature:
$$\int_a^b (\frac{d\kappa}{du})^2[\mathbf{c}(u)]du$$
- Variation of arc-length:
$$\int_a^b (\frac{dl}{du})^2[\mathbf{c}(u)]du$$
- Torsion-based functionals

Other available tools are expressed as any user-specified combination of the above basic modules. The generic formulation can be expressed as:

minimize $f(x)$ (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. *Surface Functionals*

Energy functionals of NURBS surfaces are more complicated than those of NURBS curves because there are two independent parametric variables. We proceed in a similar fashion to provide users a set of functional primitives for NURBS surfaces. Various design tools can be obtained through their linear weighted combination. Currently, our system can minimize the following functionals:

- 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.

3.4. *Least Motion of Control Variables*

The DOFs of NURBS are highly redundant in the sense that a specific shape may be represented by different combinations of shape control variables. Therefore, it will greatly facilitate the design intention of users and make our system more natural and intuitive if we incorporate the least motion constraint into the set of functional primitives. Essentially, the least motion constraint affords the change of the shape control variables to be as small as possible, aiming to reduce the data redundancy of NURBS. Typical constraints for the least motion principle may be expressed as:

$$\sum_i \sigma_i (p_i^* - p_i^0)^2, \tag{3}$$

where p_i^* denotes the optimal value of p_i, while p_i^0 denotes the initial value of p_i. Note that, Equation (3) is equivalent to the incorporation of extra material inertia

into NURBS geometry. A more intuitive analogy is that each generalized coordinate in the DOF vector is virtually connected with its initial position through a spring whose rest length is zero. The stiffness distribution of springs controls different weights for each entity in the sum of squared distance. Therefore, we can assign different (discretized) material properties to different regions across the NURBS parametric domain in an intuitive way, resulting in the local control through pseudo-physics.

The least motion principle can be straightforwardly generalized from curves to surfaces for NURBS. In addition, it can be extended from the discretize objective function in Equation (3) to the continuous integral of general energy functionals that are defined everywhere across the NURBS parametric domain. One typical example is that we can express the final value of $\mathbf{c}(u_0)$ as $\mathbf{c}^*(u_0)$ and constrain it to its original value $\mathbf{c}^0(u_0)$. More complicated examples involve the normal constraint of $\mathbf{N}(\mathbf{s}^0(u_0))$, the curvature constraint of $\kappa(\mathbf{s}^0(u_0))$, where curvature distribution κ and normal distribution $\mathbf{N}$ are differential operators on $\mathbf{c}(u)$. Our system supports both the discretized objective function and the continuous functionals (expressed as the integral form) for normal and curvature constraints based on the least motion principle.

Furthermore, our system is open-architectured and extensible in the sense that it permit users to define arbitrary objective functions of their own interest, our numerical algorithm will dynamically minimize the user-defined functional if users can provide the evaluation function of the functional (that can result in any meaningful geometric configuration for NURBS) as the new API in our system. Simple linear constraints on NURBS control variables can be easily imposed using the similar technique explained in [19] through the dimensional reduction of DOFs. General geometric constraints can be formulated and solved using techniques of Lagrange multipliers or penalty functions.

General functionals are continuous integral operators defined over certain function space. In our NURBS modeling environment, all modeled objects are considered to be NURBS which can be decomposed into a set of NURBS basis functions. Thus, the underlying function space over which our functionals are defined is limited to NURBS space. In addition, energy functionals become functions of shape control variables such as control points and knots due to the parametric integration:

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

where u is eliminated by the functional operator λ. We will use L as the generic objective function in the following sections in the interest of brevity.

4. Gradient with respect to NURBS Knots

This section details the transformation from NURBS to a set of rational Bezier splines that aims to disassociate NURBS knots with their basis functions in order to facilitate the gradient computation of energy functionals.

4.1. *Gradient Computation*

Our optimization algorithm is based on Polak-Ribiére's Conjugate Gradient (CG) method [25] which computes the gradient information in order to speed up the convergence. We now address the concept of re-parameterization and its effect on NURBS knots.

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

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

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

$$\mathbf{p} = \{p_i\}_{i=1..5n+k} = [\{p_{jd}\}_{j=1..n,d=1..3}\ ,\ \{w_j\}_{j=1..n}\ ,\ \{t_j\}_{j=1..n+k}]\,, \tag{4}$$

The gradient of the generic objective function L is now expressed as:

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

We shall investigate the contents of gradient $\mathbf{g}$.

Note that, it is tremendously challenging and far from trivial to derive the gradient with respect to non-uniform knots. This is primarily because NURBS knots are directly employed to define all the basis functions. As a result, they are *hidden* inside basis functions and become transparent to the integral operators. Another important observation is that only the relative positions of consecutive knots (i.e., knot intervals) are actually used to determine NURBS geometry. The transformation of knot distribution based on either translation or scaling operations only re-parameterize the same shape without perturbing its geometry. We can normalize the parametric domain $[t_k, t_{n+1}]$ to $[0, 1]$ by assuming t_k and t_{n+1} to be 0 and 1, respectively, without affecting the true DOFs.

We further investigate our new domain parameterization after the normalization. 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 $t_{i-k+1}, \cdots, t_{i+k}$, and more precisely, $t_{i-k+2}, \cdots, t_{i+k-1}$ (see Fig. 2). We move the knot t_i slowly towards its right neighbor and pass the u, now $\mathbf{c}(u)$ is jumping to the adjacent span. In this scenario, the relevant control polygon that defines $\mathbf{c}(u)$ is modified. In general, this may not present the modeling difficulty to designers except for interactive manipulation. However, the point shift from one span to its neighboring span resulted from the time-varying knots is counter-intuitive in principle. It is our hope that each point on NURBS is only a function of a fixed subset of the control polygon.

4.2. *Curve Re-parameterization*

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

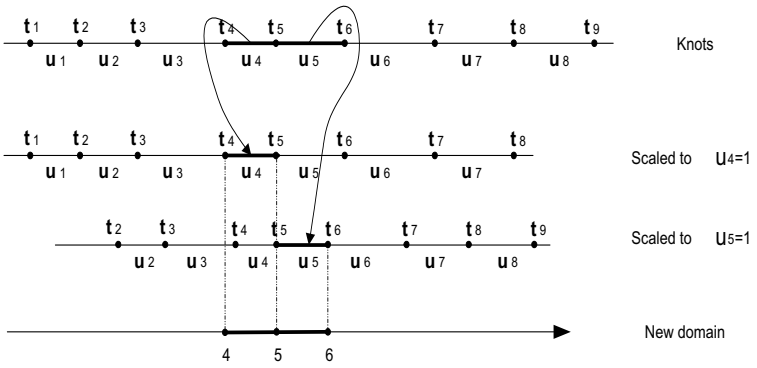


Figure 1: Uniformly spaced B-spline curve spans.

Consider each knot interval $t_i \leq u < t_{i+1}$, 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 parameterization scheme, for the i^{th} B-spline span, i.e., $\mathbf{c}(u)$, $t_i \leq u < t_{i+1}$:

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

where $i \leq \hat{u} < i + 1$, 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 $\mathbf{c}(u), t_k \leq u \leq t_{n+1}$ (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. Fortunately, the same curve still maintains G^{k-1-r} continuity.

We can further transform these B-spline curve spans into Bezier curves. A B-spline curve span $\mathbf{c}(u)$, $t_i \leq u < t_{i+1}$ determined by knots $t_{i-k+1}, \cdots, t_{i+k}$ and control polygon $\mathbf{p}_{i-k+1}, \cdots, \mathbf{p}_i$ is also a Bezier curve. The control polygon of this Bezier curve span can be derived from the preceding control polygon and its relevant knots. Note that, all the Bezier basis functions are defined over a standard domain $[0, 1]$, and they are only a function of parameter u', hence non-uniform knots are no longer associated with the new basis functions. Instead, non-uniform knots are used to define the new control points and their associated weights in the Bezier formulation. We denote the local Bezier parameter as u'. By indexing u' with the

span identifier (e.g., i), we develop a globally unique parameter for each point on the Bezier/B-spline curve, for example we let $\breve{u} = u' + i$. 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 pre-computed with appropriate sampling density in parameter domain [0..1] and these values can be stored in a look-up table for any future references.

$$\mathbf{c}_2(\breve{u}) = \mathbf{c}_{bez}(u', \mathbf{p}'_1, \cdots, \mathbf{p}'_4); \quad \mathbf{p}'_i = \mathbf{p}'_i(\{\mathbf{p}_j\}, \{t_j\}), \tag{6}$$

The control polygon $\mathbf{p}'_i$ for the i-th Bezier curve is determined by the relevant B-spline control polygon and its corresponding knot sequence. We now use curly brackets for vector components to simplify our mathematical derivation. Thus we eliminate the B-spline knot sequence by converting it into the Bezier control polygon. One significant contribution of this paper is that we can transform arbitrary NURBS curve into a set of geometrically equivalent rational Bezier curves whose control points and weights are functions of NURBS generalized coordinates (i.e., their control points, non-unity weights, and non-uniform knots). To simplify our discussion, we focus on cubic B-splines and demonstrate our transformation technique from B-splines to a set of piecewise Bezier curves. NURBS is a homogeneous representation of four-dimensional B-splines, and therefore, can be formulated analogously.

Fig. 2 illustrates the procedure of converting a cubic B-spline curve and their control points to a set of cubic Bezier curve and their accompanying control points. With control polygon $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$ and knot sequence $t_1, \cdots, t_9$, we can obtain two Bezier curves between parametric domain $t_4 \leq u < t_5$ and $t_5 \leq u < t_6$, respectively, shown as curve AD (solid) and curve DG (dash). In this example, the Bezier control points of curve AD and curve DG are $ABCD$ and $DEFG$, respectively.

Now, let us focus on the derivation of Bezier control polygon $ABCD$ which is uniquely determined by $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$, and knots $t_1, \cdots, t_8$. We divide each edge using certain ratios as shown in Fig. 2, where $u_2, \cdots, u_6$ are knot intervals, i.e., $u_i = t_{i+1} - t_i$. We observe that B-spline knots have been implicitly incorporated into the Bezier polygons which do not involve knots at all, because all Bezier curves are defined on parametric domain $[0, 1]$.

Now, let us explicitly derive the formulation of Bezier control polygon for a cubic B-spline curve:

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

This cubic B-spline curve is equivalent to a cubic Bezier curve:

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

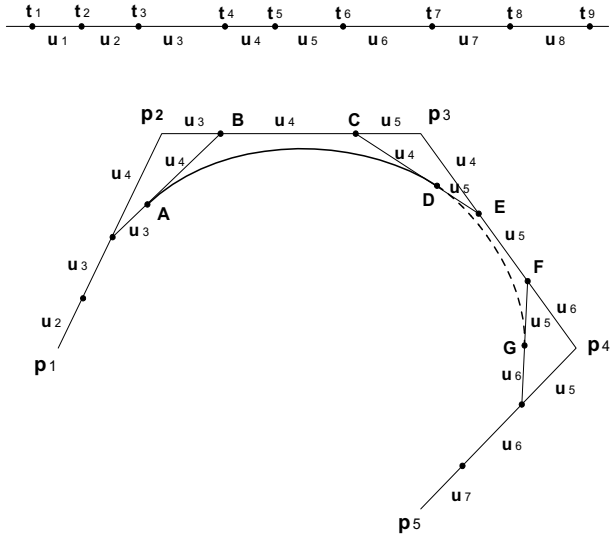


Figure 2: Conversion of a cubic B-spline curve to a set of Bezier curves.

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

$$\begin{bmatrix} \mathbf{p}'_1{}^\top \\ \mathbf{p}'_2{}^\top \\ \mathbf{p}'_3{}^\top \\ \mathbf{p}'_4{}^\top \end{bmatrix} = \begin{bmatrix} \frac{u_4}{u_3+u_4} & \frac{u_3}{u_3+u_4} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{u_5}{u_4+u_5} & \frac{u_4}{u_4+u_5} \end{bmatrix}.$$

$$\begin{bmatrix} \frac{u_4}{u_2+u_3+u_4} & \frac{u_2+u_3}{u_2+u_3+u_4} & 0 & 0 \\ 0 & \frac{u_4+u_5}{u_3+u_4+u_5} & \frac{u_3}{u_3+u_4+u_5} & 0 \\ 0 & \frac{u_5}{u_3+u_4+u_5} & \frac{u_3+u_4}{u_3+u_4+u_5} & 0 \\ 0 & 0 & \frac{u_5+u_6}{u_4+u_5+u_6} & \frac{u_4}{u_4+u_5+u_6} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \\ \mathbf{p}_4^\top \end{bmatrix}$$

We can rewrite the previous derivation using the original knot vector $\{t_i\}$ instead of knot interval vector $\{u_i\}$,

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

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

In general, for arbitrary B-spline curve, denoting the product of the first two matrices above as $\mathbf{M}$, we can further simplify the previous formulations:

$$\mathbf{c}(\breve{u}, \mathbf{p}) = [N_{1,k}(u'), \cdots, N_{k,k}(u')]\, \mathbf{M}(t_{\bar{u}-k+1}, \cdots, t_{\bar{u}+k}) \cdot \left[\mathbf{p}_{\bar{u}-k+1}^{\top}, \cdots, \mathbf{p}_{\bar{u}}^{\top}\right]^{\top}, \quad \bar{u} = \lfloor \breve{u} \rfloor, \tag{7}$$

where, $\bar{u} = \lfloor \breve{u} \rfloor$ serves as the curve span index, and $u' = \breve{u} - \bar{u}$ is the local Bezier parameter.

For simplicity, we further abbreviate it to $\mathbf{c}(\breve{u}, \mathbf{p}) = \mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\mathbf{P}$ or simply $\mathbf{NMP}$, where $\mathbf{N}$ is the set of Bezier basis functions of degree $k-1$, $\mathbf{P}$ is the subset of the control polygon $\{\mathbf{p}_i\}$ influencing Bezier curve $\bar{u}$. Note that, for each curve, there are different $\mathbf{M}$ and $\mathbf{P}$. Now we can calculate the gradient with respect to knots.

$$\frac{\partial \mathbf{c}_2(\breve{u}, \mathbf{p})}{\partial \mathbf{p}^k} = N(u') \frac{\partial \mathbf{M}}{\partial \mathbf{p}^k} \mathbf{P}.$$

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

$$\mathbf{c}(u) = \frac{\sum_{j=1}^{n} \mathbf{p}_j w_j B_{j,k}(u)}{\sum_{j=1}^{n} \mathbf{w}_j B_{j,k}(u)} = \frac{\mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\{w_i \mathbf{p}_i\}}{\mathbf{N}(u')\mathbf{M}(\mathbf{p}^k)\{w_i\}}.$$

4.3. *Surface Re-parameterization*

Re-parameterization of a NURBS surface can be similarly derived, following the prior discussion on the curve procedure. We shall first consider the process of converting B-splines to Bezier pieces, and then proceed with a NURBS formulation for arbitrary surface:

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

The non-decreasing knot sequence is $t_1 \leq t_2 \leq \ldots \leq t_{m+k}$ along the u-axis and $s_1 \leq s_2 \leq \ldots \leq s_{n+l}$ along the v-axis. The parametric domain is $t_k \leq u \leq t_{m+1}$ and $s_l \leq v \leq s_{n+1}$. Again, the generalized coordinates of NURBS surface consist of the control points, weights, and two knot sequences which are 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, explained in details below. We denote each element of matrix $\mathbf{M}$ in (7) 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. Our new parameterization can be expressed as $\check{u} = \bar{u} + u'$ and $\check{v} = \bar{v} + v'$ similar to our curve formulations:

$$\mathbf{s}(u, v, \mathbf{p}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{p}_{ij} w_{ij} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} B_{i,k}(u) B_{j,l}(v)} \Rightarrow \mathbf{s}_2(\check{u}, \check{v}, \mathbf{p})$$

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

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, $M_{ij}^{k} = M_{ij}^{k}(k_{\bar{u}-k+1}, \cdots, k_{\bar{u}+k})$ is a square matrix whose entities are functions of the relevant knots for a specific Bezier patch.

5. Numerical Techniques

This section presents the numerical techniques used in our modeling system. The basic numerical operations in our system are functionals and their first-order and second-order derivative evaluations. Efficient computation of these fundamental operations is critical for the overall performance of the optimization process. We resort to analytic and/or numerical techniques for different cases in order to balance accuracy and efficiency. Also, we exploit the *Conjugate Gradient* technique for our multi-dimensional optimization problem where derivative information is available.

5.1. *Functional and Derivative Evaluation*

In general, allowing the knots to be time-varying control variables will make the evaluation of NURBS more time-consuming. We consider the following numerical aspects:

- **NURBS Evaluation**. Through re-parameterization, 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 pre-processing 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 heavily depends on the effective computation of NURBS derivatives (up to order n). NURBS derivatives with respect to control points and weights can be easily formulated symbolically. Although possible, however, the analytic computation of NURBS derivatives with respect to their non-uniform knots is extremely complicated, especially for those higher-order derivatives. By contrast, directly resorting to numerical solutions for NURBS derivatives with respect to knots appears to be more feasible in practice, oftentimes offering better performance with

satisfactory precision bound. We have developed a large variety of numerical routines which can either analytically or numerically evaluate NURBS derivatives. Based on these numerical algorithms, we also 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. For instance, the NURBS evaluation and its first-order derivative are computed analytically with ease. However, NURBS area, second-order derivative, curvature, as well as the variation of curvature should be computed numerically due to their complexities. This is mainly because NURBS are based on homogeneous coordinates, and their higher-order derivatives require many multiplication and division operations.

- **Numerical Integration**. Continuous functionals require function integration. The closed-form analytic solution for the integral of arbitrary functions is almost impossible except for very few cases such as polynomials. For piecewise rational polynomials such as NURBS and their derivatives, however, numerical integration appears to be the only computational means. In our system, in particular, we take advantages of different numerical methods for integration which are applicable to different cases. Among them, Gaussian quadrature [25] 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 discontinuity at NURBS knots occur. 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 where singularities do occur, we use Simpson's quadrature instead, which also samples the boundary value for the evaluated functional. Fig. 3 shows the case where Gaussian quadrature fails to offer a good approximation for the integral computation of arc-length. The solid dots on x-axis are sampling points of Gaussian quadrature.

5.2. *Conjugate Gradient Method*

Our optimization problem falls into the category of multi-dimensional nonlinear system, where, the objective function is continuous and has continuous first-order derivatives. In applied mathematics, this class of problems are primarily addressed by iterative approaches, in which, each step is a one-dimensional optimization problem. More formally speaking, we employ an iterative procedure, which generates a sequence of points $\mathbf{x}^k$ converging to the optimum $\mathbf{x}^*$ of the objective function f:

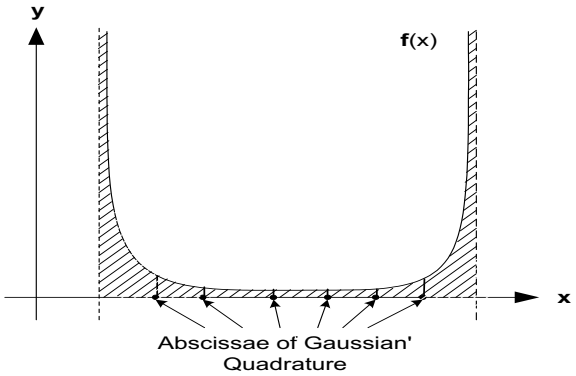


Figure 3: Gaussian quadrature fails to offer a good solution when the underlying function becomes singular at the vicinity of two boundary points.

Choose a starting point $\mathbf{x}^0, k := 0$
DO
  $\mathbf{g}_k := -\nabla f(\mathbf{x}^k)$
  $\mathbf{d}_k := func(\mathbf{g}_0, \cdots, \mathbf{g}_k)$
  Find λ_k such that
    $f(\mathbf{x}^k + \lambda_k \mathbf{d}_k) = \min_{\lambda \geq 0} f(\mathbf{x}^k + \lambda \mathbf{d}_k)$
  Set $\mathbf{x}^{k+1} := \mathbf{x}^k + \lambda_k \mathbf{d}_k$
  Set $k = k + 1$
UNTIL $\|\mathbf{x}^k - \mathbf{x}^*\| < \epsilon$

The choice of the function to derive the displacement direction $\mathbf{d}_k$ in each step from the gradient information gives different optimization techniques. Taking $\mathbf{d}_k$ as $-\mathbf{g}_k$ results in the *steepest descent method*. Despite its simplicity, the number of necessary steps to minimize ill-conditioned functions of the "narrow and elongated valley" type (see Fig.4) may be very large.

The *Conjugate Gradient Method* and *Quasi-Newton's Method* can alleviate this difficulty with great satisfaction. Both of them offer quadratic convergence rate. Our system uses *Polak-Ribiére* method, a variant of the *Conjugate Gradient* method, where

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \frac{\mathbf{g}_{k+1}^\top [\mathbf{g}_{k+1} - \mathbf{g}_k]}{\mathbf{g}_k^\top \mathbf{g}_k} \mathbf{d}_k.$$

We refer readers to [11] for a detailed description of these two families of algorithms.

5.3. *Hard Constraint Enforcement for NURBS*

NURBS shape variables are subject to certain inherent hard constraints, e.g., the non-decreasing property of knot vectors, and the positivity of weights. To enforce these hard constraints by using Lagrange multipliers or by adding penalty terms

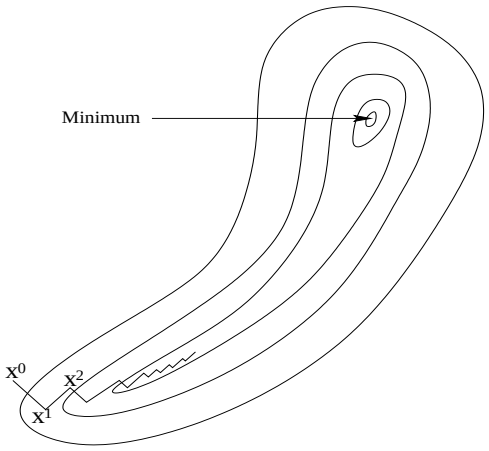


Figure 4: Steepest descent method within a narrow and elongated valley.

to the objective functional is not efficient. Lagrange multiplier method introduces additional variables to be determined, while adding penalty terms oftentimes leads the solver to travel along a long narrow valley in the domain, which inevitably involves many small steps in the optimization process while approaching the optimum. Fig.5 shows the impact of introducing non-decreasing penalty terms to the objective function, which is represented as isolines; the parametric domain is limited to $t_1 < t_2$ half plane. The penalty terms form a steep wall near line $t_1 = t_2$; the sum of the penalty term and objective function forms a deep, narrow, minute curved valley toward the minimum.

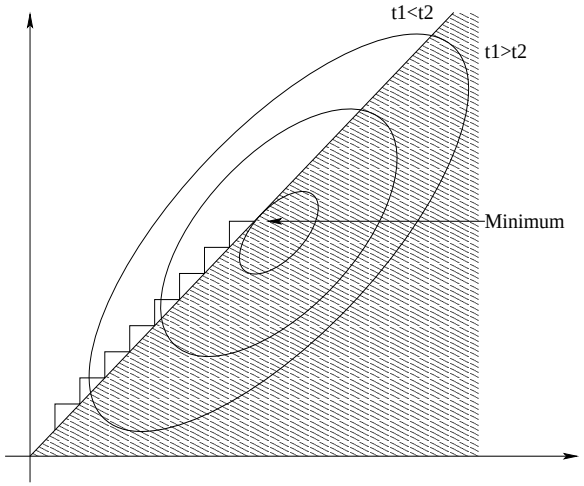


Figure 5: Penalty term forms a steep wall.

To alleviate "efficiency penalty" for hard constraint enforcement, in our system, we intentionally tune the displacement direction in each step to be parallel to the wall. This results in the following two fine-tuned algorithms for the computation of gradient with respect to weights and gradient with respect to knots respectively. The tuning of gradient with respect to weights are quite simple:

```
FOR i=1 TO m
    IF w_i < ε AND d_i < 0 THEN SET d_i = 0
END
```

The tuning process of gradient with respect to knots involves three main stages, collision detection, inverse movement detection and averaging operations:

```
Step 1, Collision detection:
  FOR i=1 TO N - 1
    FOR j = i+1 TO N
      IF t_{i+1} − t_i < ε CONTINUE
      ELSE GOTO 1
    END
1:  OUTPUT group [i..j-1]
    SET i := j
  END
```

```
Step 2, Inverse movement detection:
  FOR EACH output group of Step 1
    FOR i = groupstart TO groupend
      FOR j = i TO groupend
        IF avg(d_i..d_j) < avg(d_{j+1}..d_{groupend})
          OUTPUT group[i..j]
      END
      SET i = j + 1
    END
  END
```

```
Step 3, Averaging gradient:
  FOR EACH output group of Step 2
    FOR EACH i IN group
      SET d̄_i = avg(d_{groupbegin}..d_{groupend})
    END
  END
```

where $\bar{d}_i$'s are the components of the new displacement direction. The gradient tuning can effectively enforce the inherent hard constraints upon NURBS variables with very little additional expense.

At each stage of the CG method, we also need to set a upper-bound that specifies how far the line-minimizer can proceed. To be more detailed, assuming the new tuned displacement direction for each iteration of line-minimizing 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 set a limit along this direction as $Min_i\{(t_{i+1} - t_i)/(d_i - d_{i+1})\}$ in order to enforce the geometric constraints for knots. Similarly, the same principle can be applied to weights.

6. System Organization

This section presents a brief description of our system organization, which includes two aspects: (1) the data structure, and (2) the organization of algorithm modules.

6.1. *Data Structure*

NURBS curves and surfaces can be partitioned into spans or patches which are determined by only a small subset of the shape variables. We can view each curve span or surface patch as an element. The FEM data organization for NURBS can afford users better topological adaptability (see Fig.6).

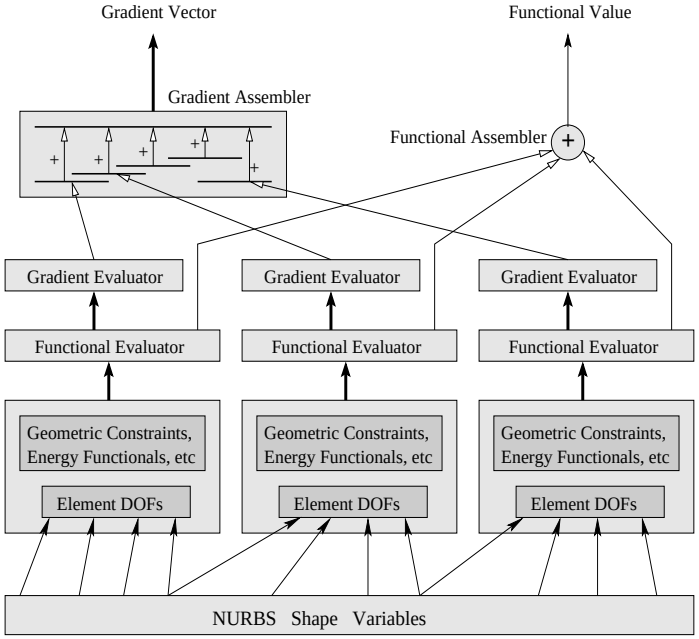


Figure 6: A FEM data organization for NURBS and its data flow.

All shape variables are stored in a global vector. Each element allocates a data structure containing pointers to its relevant shape variables. For better control on local properties, geometric and physical properties such as local material properties, local potential energy functionals, local geometric constraints, etc, are stored in the data structure of each element.

Fig.6 also outlines the evaluation of the functional values, and the gradient of potential energies, as an assembly process (or summation process) over each element.

We partition the parametric domain for each element, and all the quantities are computed on individual element. Therefore, the integral must be partitioned into a summation of the sub-integral over each element. This enables us to evaluate these quantities in a parallel fashion:

$$L(p_1, \cdots, p_{n+k}) = \int_{\Omega} \lambda[\mathbf{c}(u, p_1, \cdots, p_{n+k})]du$$

$$= \sum_i \int_{\Omega_i} \lambda[\mathbf{c}(u, \mathbf{p}_{relevant})]du, \tag{9}$$

and correspondingly

$$\frac{\partial}{\partial p_j} L = \sum_i \int_{\Omega_i} \frac{\partial}{\partial p_j} \lambda[\mathbf{c}(u, \mathbf{p}_{relevant})]du. \tag{10}$$

where Ω_i is the parametric domain of each element. We can calculate the gradient over each element in a parallel fashion and then assemble them together. Note that, a single element is only related to a subset of the shape variables, i.e., only a subvector of the gradient for that element is non-zero.

6.2. *Algorithm Modules*

Fig.7 illustrates the organization of our numerical system. The gradient of functionals ($\mathbf{g}_i$ in Section) are evaluated numerically by perturbing the input control mesh. The gradients with respect to shape variables are then adjusted to enforce inherent hard constraints as described in Section . The Conjugate Gradient generator are subsequently invoked to give a new displacement direction ($\mathbf{d}_i$ in Section) for the line-minimizer. Before calling the line-minimizer, we set an upper bound to specify how far the line-minimizer can go. The output of line-minimizer produces a sequence of solutions $\mathbf{x}_i$ converging to the optimum. The error controller determines when to stop.

Our system also offers users several modules and procedures that help the effective computation of Gaussian quadrature for numerical integration, normal and curvature evaluation, and NURBS derivatives of any order. Meanwhile, based on these fundamental numerical routines, our system provides users a large variety of basic functionals (detailed in the previous sections) in order to help users to construct more complicated ones.

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. This section explains several application examples only for NURBS surface design (because NURBS curve can be viewed as a simplier case than NURBS surface).

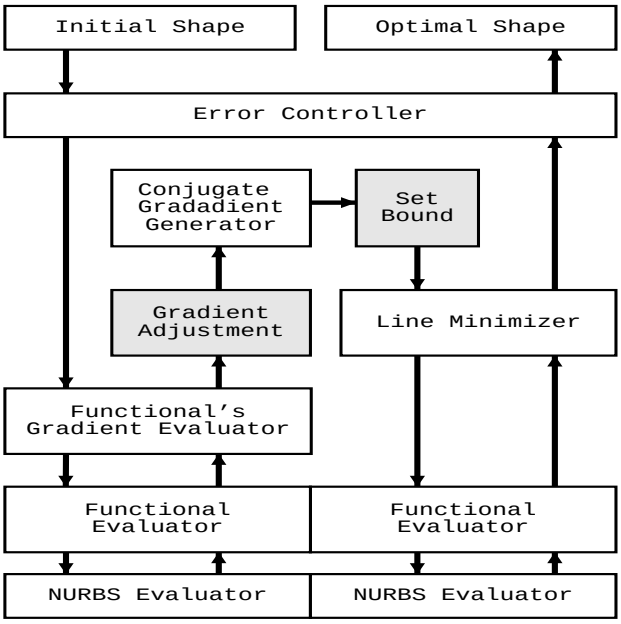


Figure 7: Conjugate optimizer structure.

- **Minimize Gaussian and mean curvatures at a user-specified point on the surface**.
  Fig.8(a) is a NURBS surface with 3×3 patches. The boundary control points are fixed, and meanwhile the end knots along both u and v direction are repeated with multiplicity 4. Fig.8(b) shows the NURBS shape after minimizing the Gaussian curvature at a user-specified point (refer to a big red dot near the top of the curve). Fig.8(d) and Fig.8(e) are the curvature maps for (a) and (b), respectively. In which, red denotes larger Gaussian curvature, green denotes larger mean curvature, and yellow is the mixture of red and green. Fig.8(g) and Fig.8(h) documents the NURBS knots before and after the operation, which demonstrate the time-varying nature of NURBS knots in our system.

- **Minimize the curvature variation**.
  Fig.8(c) shows the sculpting tool that can minimize the variation of curvature throughout the NURBS area. We fix the boundary control points as well as the four inner control points represented in red color near the top. Note that, sometimes fixing a subset of the 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.8(f) demonstrates the corresponding curvature map. We can clearly see the curvature contrast across different regions of the

NURBS surface. Fig.8(i) documents the new knot vectors for the resultant surface, subject to the minimization of curvature variation.
In Fig.9, a torus is deformed after moving its control points and knots, we can recover the torus geometry by minimizing the variation of curvature throughout the shape area. Note that we fix half of the control polygon (colored in red). In this example, we also introduce the cyclic knot vectors, that is, $u_i = u_{i+n}$ for some n.
In Fig.10, a coke bottle is initialized to have a fancy, interesting shape at the bottom. A two-step minimization approach is used to round the geometry at the bottom, in which, we first minimize the variation of the curvature by moving the control points and then we optimize the variation of the curvature by adjusting the NURBS knot vectors. This is an effective way to address the saddle point problem due to the redundancy of NURBS DOFs, especially for unknown knots of NURBS.

- **Interactively specify a surface point and manipulate this surface point directly to arbitrary location in 3-space.**
  Fig.11(a) and Fig.11(b) show two NURBS surface shapes before and after manipulating the user-specified point, respectively. Our system allows users to pick any point across the entire NURBS surface and edit its location, normal, and curvature directly. In this example, we use the NURBS denominator distribution as a texture map, or weight map (red color stands for larger NURBS denominator) to illustrate our system's capability of manipulating weights.

- **Enforce the area-preserving constraint on any specified region of interest**.
  Fig.11(c) shows the result of changing the area value for the shape in Fig.11(a) to a new value subject to the least-motion constraint. 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, in order to satisfy area-preserving constraints.

- **Physics-based shape manipulation.**
  In Fig.11, (d), (e), (f), and (g) illustrate the physics-based sculpting tools such as specifying material property, exerting external forces, etc. Our system can accommodate a large set of modeling criteria ranging from physical properties such as material, forces, to various geometric constraints.

- **Combine numerous sculpting tools together for real-time geometric design.**
  In Fig.11, (h), (i), (j), and (k) present an interesting example of sequentially using a set of sculpting tools to morph a goblet to another shape.

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 NURBS Jacobian matrix through both symbolic and numerical computation. The new, improved NURBS formulation and its modeling framework based on the principle of energy optimization afford all 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.

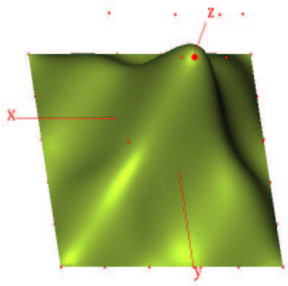
We have developed a prototype interactive modeling system based on the new NURBS formulation and have demonstrated the flexibility of our models in a variety of applications. In particular, we have extended our previous NURBS system by offering users a wide array of non-quadratic curvature-based functionals that are dynamically minimized during the user 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, significantly enhancing NURBS functionalities in various visual computing applications.

9. Acknowledgments

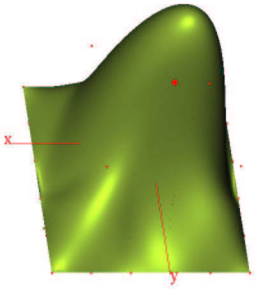
This research is supported in part by the NSF CAREER award CCR-9896123, the NSF grants DMI-9896170, IIS-0082035, IIS-0097646, and research grants from Ford Motor Company and Honda America, Inc.

1. R. Barnhill. A survey of the representation and design of surfaces. *IEEE Computer Graphics and Applications*, 3(7):9–16, 1983.
2. F. Cheng and B. Barsky. Interproximation: interpolation and approximation using cubic spline. *Computer-Aided Design*, 23(10):700–706, 1991.
3. J. Chou and L. Piegl. Data reduction using cubic rational B-splines. *IEEE Computer Graphics and Applications*, 12(3):60–68, 1992.
4. 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.
5. G. Farin. Trends in curve and surface design. *Computer-Aided Design*, 21(5):293–296, 1989.
6. G. Farin. From conics to NURBS: A tutorial and survey. *IEEE Computer Graphics and Applications*, 12(5):78–86, Sept. 1992.
7. D. Ferguson and T. Grandine. On the construction of surfaces interpolating curves: I. A method for handling nonconstant parameter curves. *ACM Transactions on Graphics*, 9(2):212–225, 1990.

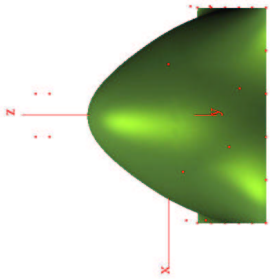
8. A. Forrest. Interactive interpolation and approximation by Bezier polynomials. *Computer-Aided Design*, 22(9):527–537, 1990.
9. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–77, 1992.
10. J. Hoschek. Approximate conversion of spline curves. *Computer Aided Geometric Design*, 4(1-2):59–66, 1987.
11. M. Minoux. *Mathematical Programming*. Wiley, New York, 1986.
12. 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.
13. 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.
14. L. Piegl. Modifying the shape of rational B-splines. part 1:curves. *Computer-Aided Design*, 21(8):509–518, 1989.
15. L. Piegl. Modifying the shape of rational B-splines. part 2:surfaces. *Computer-Aided Design*, 21(9):538–546, 1989.
16. L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, Jan. 1991.
17. L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485–498, 1987.
18. M. Pratt, R. Goult, and L. Ye. On rational parametric curve approximation. *Computer Aided Geometric Design*, 10(3-4):363–377, 1993.
19. Hong Qin and Demetri Terzopoulos. D-NURBS: A physics-Based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, March 1996.
20. B. Sarkar and C. Menq. Parameter optimization in approximating curves and surfaces to measurement data. *Computer Aided Geometric Design*, 8(4):267–290, 1991.
21. B. Sarkar and C.-H. Menq. Smooth-surface approximation and reverse engineering. *Computer-Aided Design*, 23(9):623–628, 1991.
22. L. Shirman and C. Sequin. Procedural interpolation with geometrically continuous cubic splines. *Computer-Aided Design*, 24(5):267–277, 1992.
23. W. Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, Sept. 1983.
24. K.J. Versprille. *Computer-Aided Design Applications of the Rational B-Spline Approximation form*. PhD thesis, Syracuse University, 1975.
25. William T. Vetterling Willian H. Press, Saul A. Teukolsky and Brian P. Flannery. *Nemerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition edition, 1992.



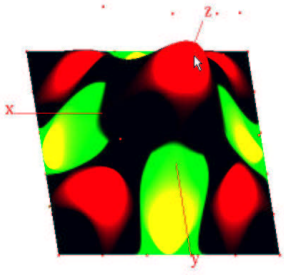
(a) A NURBS surface with 3×3 patches.



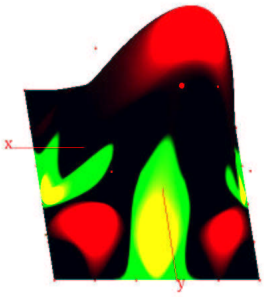
(b) Minimize the curvature at a given point (big red dot).



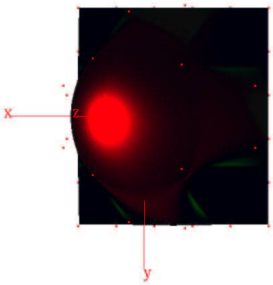
(c) Minimize the variation of the curvature over the entire surface area.



(d) Curvature map of (a).



(e) Curvature map of (b).



(f) Curvature map of (c).

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

(g) The NURBS knot vectors of surface (a).

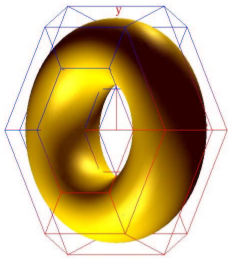
u	0.0000	0.0000	0.0000	0.0000	1.4356	1.7723	3.0000	3.0000	3.0000	3.0000
v	0.0000	0.0000	0.0000	0.0000	1.2435	1.3637	3.0000	3.0000	3.0000	3.0000

(h) The NURBS knot vectors of surface (b).

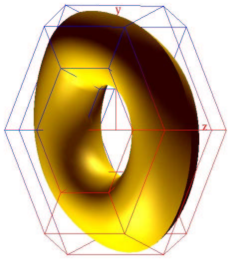
u	0.0000	0.0000	0.0000	0.0000	0.4513	2.5487	3.0000	3.0000	3.0000	3.0000
v	0.0000	0.0000	0.0000	0.0000	0.4513	2.5487	3.0000	3.0000	3.0000	3.0000

(i) The NURBS knot vectors of surface (c).

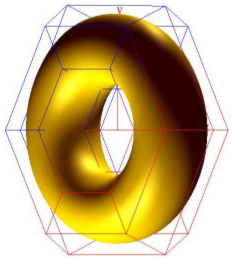
Figure 8: Curvature manipulation of a NURBS surface with $\mathbf{3} \times \mathbf{3}$ patches. The control points along are fixed, while the end knots of both **u** and **v** are repeated with multiplicity 4.



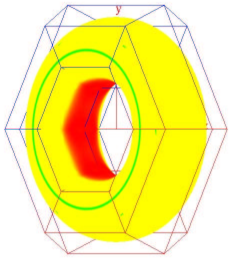
(a) A torus with 6×6 patches.



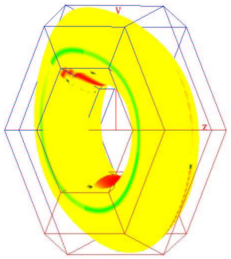
(b) Deformed torus.



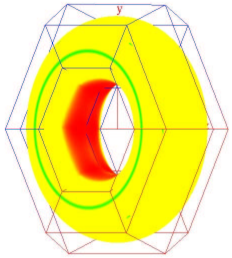
(c) Recovered torus.



(d) Curvature map of (a).



(e) Curvature map of (b).



(f) Curvature map of (c).

u	0.0000	0.1667	0.3333	0.5000	0.6667	0.8333	1.0000
v	0.0000	0.1667	0.3333	0.5000	0.6667	0.8333	1.0000

(g) The NURBS knot vectors of the original torus in (a).

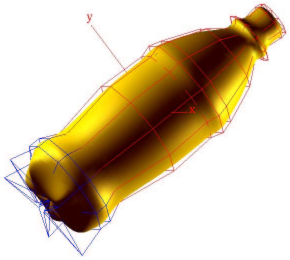
u	0.0000	0.1116	0.4815	0.4815	0.5656	0.8599	1.0000
v	0.0000	0.3556	0.3556	0.3556	0.6774	0.8297	1.0000

(h) The NURBS knot vectors of the deformed torus in (b).

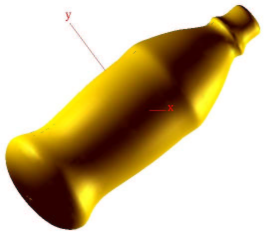
u	0.0000	0.1760	0.3405	0.4944	0.6682	0.8323	1.0000
v	0.0000	0.1827	0.3300	0.4635	0.6901	0.8269	1.0000

(i) The NURBS knot vectors of the restored torus in (c).

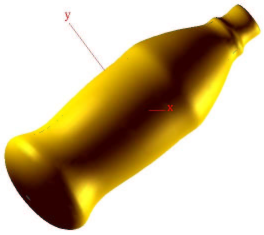
Figure 9: A torus is deformed by moving its control points and knots, we can accurately reconstruct it by minimizing the variation of the curvature. Half of the control polygon (colored in red) are fixed in both the deformation and reconstruction process. We use cyclic knot vectors, i.e., $\mathbf{u_i} = \mathbf{u_{i+6}}$ and $\mathbf{v_i} = \mathbf{v_{i+6}}$.



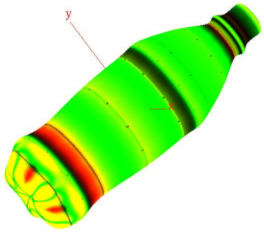
(a) A bottle with 10 × 13 patches, in which only the blue part of the control polygon is allowed to change.



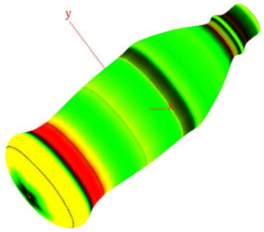
(b) Step 1: minimize the variation of the curvature by optimizing control points.



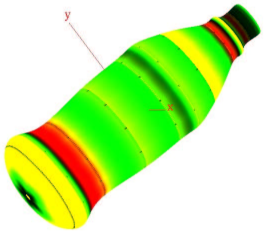
(c) Step 2: further minimize the variation of the curvature by optimizing knot vectors.



(d) Curvature map of (a).



(e) Curvature map of (b).



(f) Curvature map of (c).

u	.0000	.1000	.2000	.3000	.4000	.5000	.6000	.7000	.8000	.9000	1.000
v	.0000	.0000	.0000	.0000	.0769	.1538	.2308	.3077	.3846	.4615	.5385
	.6154	.6923	.7692	.8462	.9231	1.000	1.000	1.000	1.000		

(g) The NURBS knot vectors for the original bottle in (a).

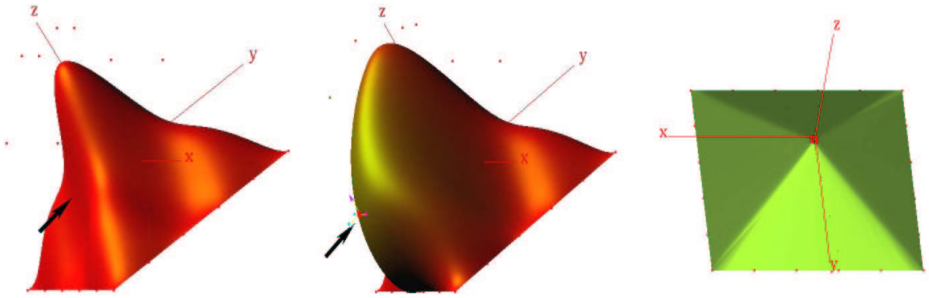
u	.0000	.1000	.2000	.3000	.4000	.5000	.6000	.7000	.8000	.9000	1.000
v	.0000	.0000	.0000	.0000	.0769	.1538	.2308	.3077	.3846	.4615	.5385
	.6154	.6923	.7692	.8462	.9231	1.000	1.000	1.000	1.000		

(h) In Step 1, the knot vectors are not changed.

u	.0000	.1000	.2000	.3000	.4000	.5000	.6000	.7000	.8000	.9000	1.000
v	.0000	.0000	.0000	.0000	.1158	.1312	.1812	.3157	.4402	.4403	.5282
	.6210	.6943	.7638	.8487	.9171	1.000	1.000	1.000	1.000		

(i) The NURBS knot vectors for the optimized bottle in (c).

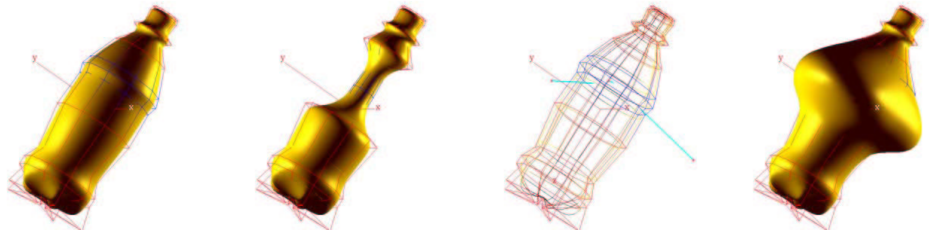
Figure 10: Minimize the variation of the curvature of a bottle, in which only a subset of the control points near the bottom (colored in blue in (a)) are allowed to change. Cyclic knot vector is used for **u**, while end knots of **v** are repeated with multiplicity 4.



(a) The weight map of a NURBS surface, red color stands for greater weight, while green for smaller one.

(b) The new weight map after moving a surface point. The system automatically figures out the new control points and weights.

(c) Impose the area-preserving constraint on the NURBS surface.

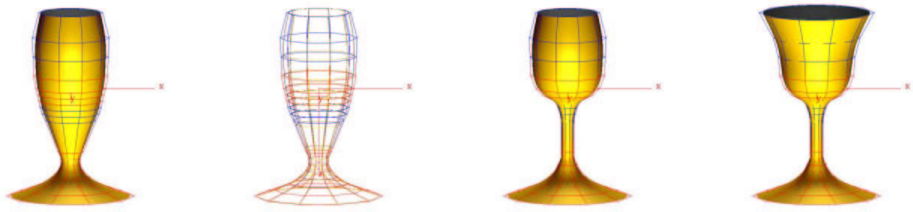


(d) A bottle, only the middle (colored in blue) part of its control polygon is allowed to change.

(e) By assigning elastic material, the movable control polygon shrinks.

(f) Exert external forces to some surface points as demonstrated by green line segments.

(g) The result surface after imposing external forces.



(h) A goblet.

(i) Red part of the control polygon is fixed.

(j) Shrink the lower part by assigning elastic material.

(k) Directly manipulate the normal at the opening.

Figure 11: This figure shows additional functionalities in our system. Essentially, by optimizing certain properly-formulated energy functionals, numerous geometric sculpting tools can be easily constructed.