Hierarchical D-NURBS Surfaces and Their Physics-based Sculpting

Meijing Zhang and **Hong Qin**
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794–4400, USA
zmeijing | qin@cs.sunysb.edu

Abstract

In this paper, we present Hierarchical D-NURBS as a new shape modeling representation that generalizes powerful, physics-based D-NURBS for interactive geometric design. Our Hierarchical D-NURBS can be viewed as a collection of standard D-NURBS finite elements, organized hierarchically in a tree structure and subject to continuity constraints across the shared boundaries of adjacent D-NURBS elements at different levels. Hierarchical D-NURBS are amenable to the flexible and powerful representation of curves, surfaces, as well as higher dimensional geometric entities. The layered and composite construction of Hierarchical D-NURBS affords users the effective creation of local features and their flexible, global/local control at different level of details. Within the framework of Hierarchical D-NURBS, users can interactively sculpt NURBS geometry more intuitively and conveniently through both global and local toolkits. Based on the data structure of Hierarchical D-NURBS, we have developed a prototype software equipped with various physics-based toolkits in the form of geometric constraints and simulated forces. Our modeling system allows users to undertake the design tasks of point manipulation, normal editing, curvature control, curve fitting, and area sculpting in interactive graphics and CAD/CAM.

Keywords:
Physics-based Modeling and Sculpting, Shape Modeling, Computer Graphics, CAGD, NURBS, Deformable Models, Dynamics, Hierarchical Splines and Editing, Geometric Constraint.

1 Introduction

Shape modeling is critical to a wide range of areas including real-time interactive graphics, computer-aided geometric design (CAGD), scientific visualization, medical imaging, and virtual environments. NURBS have become a *de facto* standard in commercial modeling systems because of their power of representing both free-form shapes and commonly-used analytic functions.

Conventional techniques for free-form surface modeling is kinematic, and they are usually associated with the tedious and indirect shape manipulation through time-consuming operations on a large number of control variables such as control points and weights. In contrast, physics-based modeling offers a superior approach to traditional free-form shape design that can overcome most of the limitations associated with conventional geometric modeling techniques. Within the physics-based framework, free-form geometric models are equipped with mass contributions, internal deformation energies, and other material properties. Users can interact with the model geometry directly by exerting virtual forces. The model responds to user interaction naturally subject to physical laws and geometric constraints.

D-NURBS is a physics-based representation that augments NURBS geometry and its modeling techniques. Governed by Lagrangian mechanics, NURBS' dynamic behavior can produce physically meaningful, and hence intuitive shape variation. This permits users to interactively manipulate the shape geometry not only through the traditional indirect fashion, such as adjusting control vertices and setting weights, but also through direct physical manipulation, such as exerting simulated force and enforcing local and global shape constraints.

Despite many unsurpassed advantages of D-NURBS, current state-of-the-art of D-NURBS techniques are yet to support the local control and the fine editing of localized regions of interest on D-NURBS primarily because all the control points and weights are organized only at one level. Although local refinements based on the principle of knot-insertion are possible, knot-insertion techniques oftentimes generate more control points and weights than what designers actually need in order to maintain the tensor-product nature of NURBS geometry. In addition, currently available point-based sculpting toolkits for D-NURBS are essentially very primitive. To further ameliorate D-NURBS potential in shape modeling and geometric design, in this paper we develop Hierarchical D-NURBS, which can enhance both the geometric power and the dynamic flexibility of conventional D-NURBS. In particular, our Hierarchical D-NURBS offer the following advantages:

- It has a hierarchical structure, and each element within this hierarchy is a NURBS patch (or a collection of several NURBS patches).
- In a nutshell, it is a composite NURBS whose control points and weights can be defined independently.
- It satisfies continuity criteria everywhere across the NURBS parametric domain.
- Its parametric domain is the same as that of a tensor-product NURBS.
- It permits the flexible creation of local features, and facilitates both global and local control in a hierarchical way.
- It is applicable for the effective representation of both curves and surfaces, and can be generalized to higher dimensional primitives.
- At each level, extra new degrees of freedom (e.g., control points and weights) can be obtained through the *localized* knot-insertion algorithm applied to certain sub-regions at the parent's level. In contrast to knot-insertion, however, control variables as well as new NURBS patches are *not* introduced outside the regions of interest.

In this paper, we also develop interactive techniques and implement a prototype software system to facilitate the direct manipulation and interactive sculpting of D-NURBS surfaces. The key contribution of this paper is that we systematically formulate Hierarchical D-NURBS as a novel shape modeling representation that generalizes well-established D-NURBS substrate. Our software system and its associated toolkits are all founded upon the novel formulation of Hierarchical D-NURBS. Using our Hierarchical D-NURBS sculpting system, users can intuitively apply various interactive tools for NURBS editing, including directly manipulating normal/curvature at arbitrary location, enforcing local and global constraints at different levels, achieving feature-based curve fitting, sculpting surface regions directly with templates, etc. With the formulation of Hierarchical D-NURBS, users can easily edit the fine detail in a specified, local region of NURBS while maintaining other nearby regions unchanged. The hierarchical structure provides users a more intuitive and efficient way to undertake surface editing and sculpting. For example, users can select an arbitrary point on a surface of Hierarchical D-NURBS, and drag the point to any desired location via mouse. The surface will deform accordingly subject to the point manipulation. Users can then modify the normal/curvature of arbitrary point on the surface, and the surface will converge to the shape with the desired normal/curvature. Furthermore, users can choose various curve tools and/or surface tools and apply them to sculpt the surface directly.

The remainder of this paper is structured as follows. Section 2 briefly reviews the prior work. In Section 3, we detail the formulation of Hierarchical D-NURBS. Section 4 presents physics-based techniques of directly manipulating D-NURBS surfaces with various forces and constraints. We outline our system implementation and present our experimental results in Section 5. Finally, Section 6 concludes the paper.

2 Prior Work

Forsey and Bartels [6] proposed hierarchical B-splines and pioneered the technique for local refinement of smooth surfaces in surface design. A geometric model of hierarchical B-splines can by defined by locally offsetting a standard tensor-product B-spline surface with a set of new, refined B-spline basis functions. In hierarchical B-splines, additional interior control points yield new degrees of freedom at the refined level within the surface hierarchy. Welch and Witkin [19] developed the variational surface modeling method. Gonzalez-Ochoa and Peters [8] formulated the localized hierarchical surface splines (LeSS), which can automatically maintain tangent continuity across the surface and can be subsequently converted to either NURBS form or a set of cubic triangular Bezier patches. Various methods have been developed to generate fair surfaces which satisfy multiple constraints and optimize an energy-based objective function [19]. It is also possible to construct dynamic surfaces with natural behavior governed by physical laws [15]. Terzopoulos and Fleischer [16] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [3] developed an interesting prototype system for interactive design based on the finite-element optimization of energy functionals. Bloor and Wilson [2] used similar energies which can be optimized through numerical methods for B-splines. Celniker and Welch [4] investigated deformable B-splines subject to linear constraints. Thingvold and Cohen [18] proposed to use elasto-plastic mass-spring-hinge models on the B-spline control points. Moreton and Sequin [11] interpolated a minimum energy curve network with quintic Bezier patches by minimizing the variation of curvature. Stewart and Beier [14] demonstrated a direct curve manipulation technique which allows the direct control of position, normal, and curvature. Halstead et al. [9] implemented smooth interpolation with Catmull-Clark surfaces using a thin-plate energy functional. Grimm and Ayers [7] proposed a framework for curve editing by maintaining multiple representations of the same curve. Zheng et al. [20, 21] presented methods for users to modify a free-form curve by using a curve sculpting tool and deform a free-form surface to follow the shape of a predefined feature surface.

3 Hierarchical D-NURBS System

This section formulates Hierarchical D-NURBS and presents its dynamic equation.

3.1 D-NURBS Decomposition

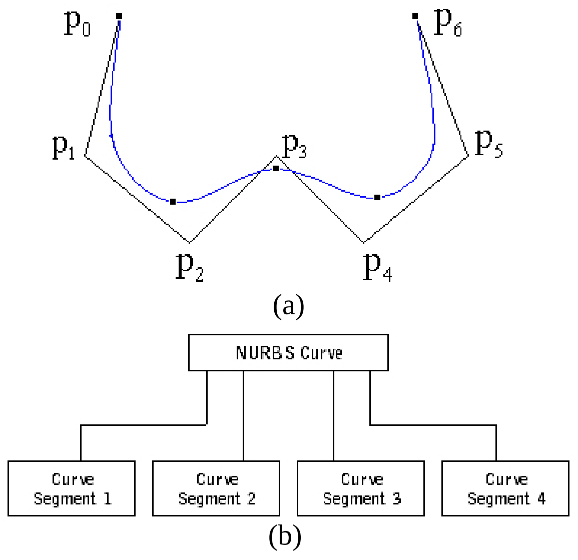


Figure 1: A NURBS curve: (a) A NURBS curve consists of 4 curve segments; (b) We consider it as a collection of 4 finite elements.

Conventional geometric NURBS curve is defined as a function of its parametric variable u:

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

where $\mathbf{p}_i$'s are control points, and the w_i's are associated nonnegative weights, and $B_{i,k}(u)$'s are basic functions. If we examine the geometric structure of NURBS curves from physics' point of view, we can decompose the entire geometric NURBS curve into a number of physical finite elements (or geometric spans). Each NURBS curve segment can be considered as a physical element. Adjacent elements share certain geometric information (such as some common control points and weights). These shared geometric information determines geometric continuity across adjacent curve segments.(see Fig. 1).

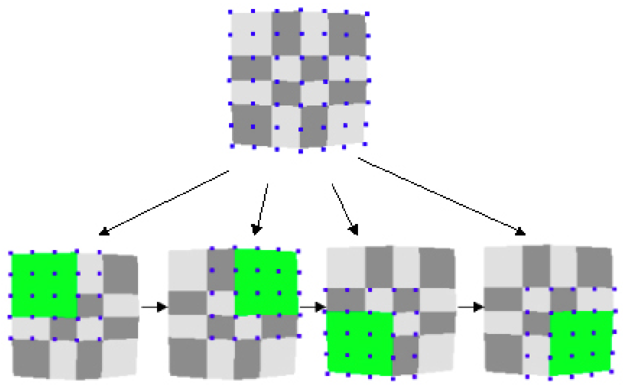


Figure 2: The NURBS surface, which consists of 4×4 patches and controlled by 7×7 control points (colored in blue), can be decomposed into 4 sub-regions (highlighted in green), each sub-region is controlled by 5×5 control points.

We now generalize this concept to the surface case. A conventional NURBS surface is defined as a function of parametric variables u and v:

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

in analogy, $\mathbf{p}_{i,j}$'s are control points, $w_{i,j}$'s are associated nonnegative weights, and $B_{i,k}(u)$ and $B_{j,l}(v)$ are basic functions along two parametric directions, respectively. Again, we can view a NURBS surface as a collection of several physical elements, the accurate number of NURBS elements depends on how users would decompose the surface geometry. In general, the number of elements has no direct relationship with the number of NURBS patches. Fig. 2 demonstrates that we can decompose a NURBS surface with 4×4 NURBS patches and 7×7 control points into 4 elements. Again, elements share geometric information, which determines the geometric continuity across the shared boundaries.

For the curve case, if we are interested in introducing local details, we can resort to all kinds of geometric algorithms such as knot-insertion. This way, we increase the degrees of freedom and re-organize the curve span distribution. However, it is important to point out that knot insertion in principle produces exactly the same geometric representation of the original curve. In addition, knot insertion permits the change of control vertices and the subsequent editing of fine details in a more localized region of the curve while maintaining other neighboring regions unaffected. After knot insertion, more curve spans (physical elements) will be introduced. Following the previous procedure, we can collect the newly-generated elements and replace the old elements with the new ones. Note that the control points for the adjacent element must be updated accordingly due to knot insertion, and the data structure must be modified significantly with great care, as showed in Fig. 3.

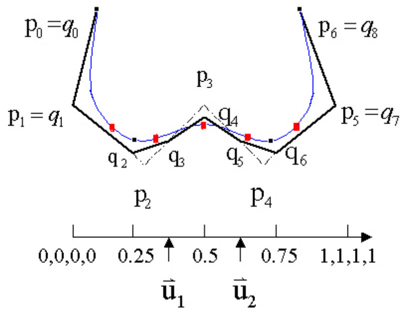


Figure 3: The new NURBS curve and its span decomposition after the insertion of two additional knots.

For the surface case, unfortunately, when we undertake the task of knot insertion in regions of interest, we must also divide other regions. This process will significantly increase the number of elements as shown in Fig. 4.

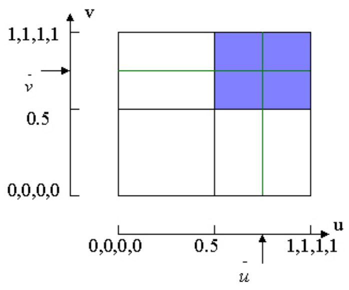


Figure 4: We insert one knot along u and v directions, respectively, in the region of interest (colored blue). As a result, we introduce more patches than what we actually need in the other neighboring regions (colored white).

To overcome the drawback associated with knot insertion for NURBS, we shall formulate Hierarchical D-NURBS. It increases the degrees of freedom of one region by introducing as many new patches as users want only in the specified region without modifying the geometric structure of adjacent regions. The key is to apply knot-insertion algorithm locally. As a result, it allows users to edit the fine details in one region of the D-NURBS surface without deforming any other regions, as shown in Fig. 5.

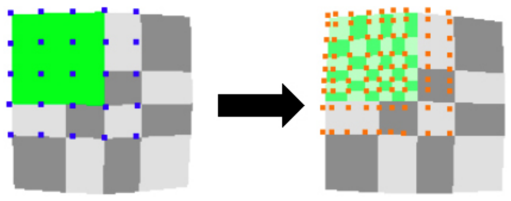
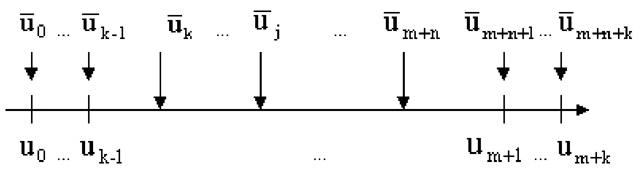


Figure 5: We subdivide the selected region (controlled by 5×5 control points) into 6×6 patches (controlled by 9×9 control points) without introducing new patches in any other regions.

3.2 Hierarchical Formulation

To arrive at a hierarchical structure for D-NURBS, let us first examine the mathematics of knot-insertion. The basis functions $B_{i,k}(u)$ and $B_{j,l}(v)$ should be refinable in the sense that each one can be re-expressed as a linear combination of one or more new, and ”smaller” basis functions $N_{r,k}(u), N_{s,l}(v)$ defined over the new knot sequence:


$\bar{u}_0$ ... $\bar{u}_{k-1}$ $\bar{u}_k$ ... $\bar{u}_j$ ... $\bar{u}_{m+n}$ $\bar{u}_{m+n+1}$ ... $\bar{u}_{m+n+k}$
u_0 ... u_{k-1} ... u_{m+1} ... u_{m+k}

$$B_{i,k}(u) = \sum_r \alpha_{i,k}(r) N_{r,k}(u) \tag{3}$$

$$B_{j,l}(v) = \sum_s \alpha_{j,l}(s) N_{s,l}(v) \tag{4}$$

where α is defined as a recursive function:

$$\alpha_{i,l}(j) = \begin{cases} 1 & u_i \leq \bar{u}_j \leq u_{i+1} \\ 0 & otherwise \end{cases} \text{ and}$$

$$\alpha_{i,l}(j) = \frac{\bar{u}_{j+l-1} - u_i}{u_{i+l-1} - u_i} \alpha_{i,l-1}(j) + \frac{u_{i+l} - \bar{u}_{j+l-1}}{u_{i+l} - u_{i+1}} \alpha_{i+1,l-1}(j) \tag{5}$$

where, u_i's are the knot series before knot-insertion, and $\overline{u}_i$'s are the knot series after knot-insertion, $l = 2, 3, ..., k$, and k is the order of the spline. In the interest of the space, we refer readers to [1] for the details of knot-insertion process. After knot-insertion, we can obtain a new formulation of the same NURBS surface with the *smaller* basis functions and a larger number of control points and weights, which will introduce more degrees of freedom to the NURBS surface:

$$\mathbf{s}(u,v) = \frac{\sum_r \sum_s \overline{\mathbf{p}}_{r,s}\overline{w}_{r,s}N_{r,k}(u)N_{s,l}(v)}{\sum_r \sum_s \overline{w}_{r,s}N_{r,k}(u)N_{s,l}(v)} \quad (6)$$

$$\overline{\mathbf{p}}_{r,s} = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{p}_{i,j}w_{i,j}\alpha_{i,k}(r)\alpha_{j,l}(s)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{i,j}\alpha_{i,k}(r)\alpha_{j,l}(s)} \quad (7)$$

$$\overline{w}_{r,s} = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}\alpha_{i,k}(r)\alpha_{j,l}(s) \quad (8)$$

where $\mathbf{p}_{r,s}$ and $w_{r,s}$ are the control points and weights before knot-insertion, $\overline{\mathbf{p}}_{r,s}$ and $\overline{w}_{r,s}$ are the control points and weights after knot-insertion. Above are the standard knot insertion process for NURBS surface. The major shortcoming of knot-insertion algorithm is that it generates more control points $\overline{\mathbf{p}}_{r,s}$'s and weights $\overline{w}_{r,s}$'s than what users are actually interested in. What are required are only those $\overline{\mathbf{p}}_{r,s}$'s and $\overline{w}_{r,s}$'s that actually control the NURBS geometry in user-specified local regions for editing purposes. Therefore, if our goal is to only edit the local detail of a region, we shall retain those relevant $\overline{\mathbf{p}}_{r,s}$'s and $\overline{w}_{r,s}$'s and ignore the remaining control points and weights which only define the nearby regions. Towards this objective, we therefore resort to a tree structure and formulate our Hierarchical D-NURBS. The root of this D-NURBS tree (level 0) is the entire NURBS surface with control points $\mathbf{p}_{i,j}$'s and weights $w_{i,j}$'s, the level 1 of the tree only consists of all the subdivided regions of interest from level 0, level 2 only collects all elements which are further subdivided from the regions of interest at level 1, and this hierarchical structure will continue recursively (if necessary) to expand to level n as illustrated in Fig. 6.

Now, it sets a stage for us to introduce another important concept: *layer*. A layer is a physical element (consists of one or several NURBS patches) at any level. Let us use Fig. 6 as an example, the root of the D-NURBS tree (level 0) is the entire D-NURBS surface which consists of all the initial geometric patches. At level 1, each of the two subdivided regions of interest forms a layer. In essence, a layer is a localized region which corresponds to a portion of its parent-level substrate. With the tree structure, if users want to edit the local detail on a specified region, they only need to subdivide the region, which will generate a new layer at the next consecutive level. Consequently, users can edit regions

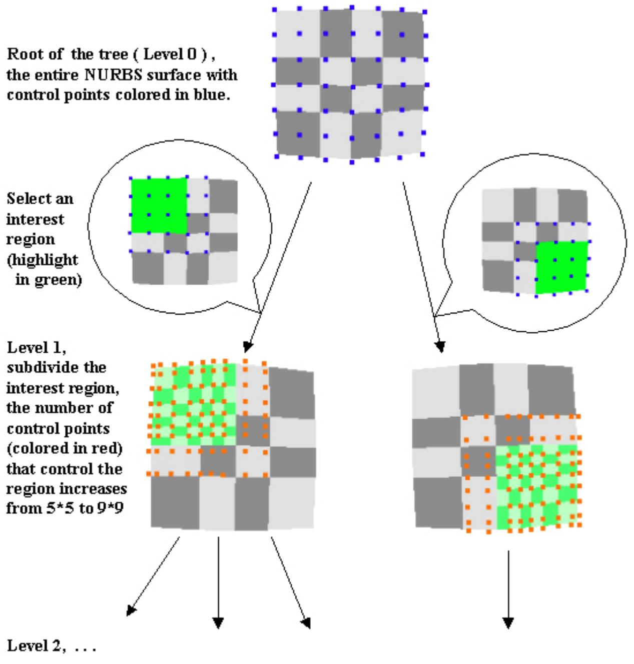

Root of the tree (Level 0), the entire NURBS surface with control points colored in blue.
Select an interest region (highlight in green)
Level 1, subdivide the interest region, the number of control points (colored in red) that control the region increases from 5*5 to 9*9
Level 2, . . .

Figure 6: The D-NURBS tree and its hierarchical structure.

of arbitrary layer. To maintain the requirement of geometric continuity across the shared boundary of different layers, we shall be careful to manipulate only the central control points and weights, and keep the peripheral control points and weights constrained whenever we manipulate localized regions of interest. In addition, when the editing takes place at one layer, regions at all of its child layers should be modified correspondingly. Let $\mathbf{p}_{i,j}^l$ and $w_{i,j}^l$ be the control points and weights in a layer of level l. $\mathbf{p}_{i,j}^l$ and $w_{i,j}^l$ are functions of times, controlling the deformation of D-NURBS. When we manipulate a layer which is a leaf node of this tree, it does not result in any additional changes in neighboring regions. However, if the current layer is a non-leaf node (i.e., its child layers do exist), then this manipulation must propagate to all of its child layers, which means that the changes of $\mathbf{p}_{i,j}^l$ and $w_{i,j}^l$ will influence the value of $\mathbf{p}_{r,s}^{l+1}$ and $w_{r,s}^{l+1}$, where the superscript stands for the level index. In essence, the editing operation on parents' levels will have a *global* effect on their child layers, hence Hierarchical D-NURBS support both global and local deformation. Furthermore, the localized and detailed features defined by $\mathbf{p}_{r,s}^{l+1}$ and $w_{r,s}^{l+1}$ must be properly maintained. Our system achieves this goal by employing a special data structure that records the localized and detailed features as the time-varying displacements with respect to its parent's layer:

$$\mathbf{r}_p = \mathbf{p}_{r,s}^{l+1}(t_1) - \mathbf{p}_{r,s}^{l+1}(t_0), \quad (9)$$

$$r_w = w_{r,s}^{l+1}(t_1) - w_{r,s}^{l+1}(t_0), \tag{10}$$

where $\mathbf{p}_{r,s}^{l+1}(t_1)$, $w_{r,s}^{l+1}(t_1)$ are the control points and weights that determine the localized and detailed features; $\mathbf{p}_{r,s}^{l+1}(t_0)$, $w_{r,s}^{l+1}(t_0)$ are the control points and weights which are initially created from $\mathbf{p}_{i,j}^{l}(t_0)$, $w_{i,j}^{l}(t_0)$ in terms of Equation (7). We can express them as $\mathbf{p}_{r,s}^{l+1}(t_0) = f_p(\mathbf{p}_{i,j}^{l}(t_0))$ and $w_{r,s}^{l+1}(t_0) = f_w(w_{i,j}^{l}(t_0))$, where f_p and f_w are known functions which are determined by the *localized* knot-insertion. Now, if $\mathbf{p}_{i,j}^{l}(t_0)$ and $w_{i,j}^{l}(t_0)$ change to $\mathbf{p}_{i,j}^{l}(t)$ and $w_{i,j}^{l}(t)$ because of the editing operations on the current layer, we must recompute $\mathbf{p}_{r,s}^{l+1}(t), w_{r,s}^{l+1}(t)$ as:

$$\mathbf{p}_{r,s}^{l+1}(t) = f_p(\mathbf{p}_{i,j}^{l}(t)) + \mathbf{r}_p \tag{11}$$

$$w_{r,s}^{l+1}(t) = f_w(w_{i,j}^{l}(t)) + r_w \tag{12}$$

The global update algorithm must proceed recursively in order to traverse all of its child layers. In this way, when editing takes place at one layer, regions at all of its child layers will be modified correspondingly without losing their current localized and detailed features, facilitating both global and local deformations.

Now we can analytically represent our Hierarchical D-NURBS surface as:

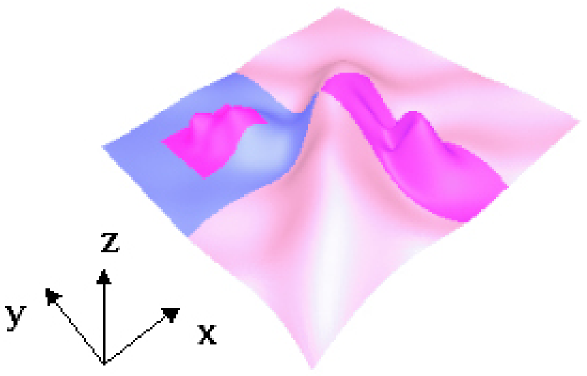
$$\mathbf{s}(u,v) = \mathbf{s}^0(u,v)+\mathbf{s}^1(u,v)+\mathbf{s}^2(u,v)+...+\mathbf{s}^\infty(u,v) \tag{13}$$

$$\mathbf{s}^l(u,v) = \sum_i \mathbf{s}_i^l(u,v) \tag{14}$$

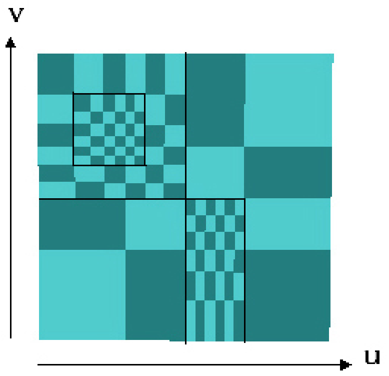
where $\mathbf{s}^l(u,v)$ is the collection of all layers (physical elements) at level l, $\mathbf{s}_i^l(u,v)$ is one of the layers in level l. Besides major advantages previously mentioned at the beginning of this paper, our Hierarchical D-NURBS are both powerful and flexible because:

- It inherits a lot of nice properties from NURBS geometry, as each element within the tree hierarchy is a single NURBS surface. Therefore, existing matured algorithms and modeling techniques are amenable to Hierarchical D-NURBS.

- Its geometry results from the localized operation of knot-insertion algorithms on the specified layer. Therefore, it can overcome typical drawbacks associated with the standard process of knot-insertion, and it does not introduce new degrees of freedom and new patches in regions of non-interest. Each level is a collection of NURBS patches, and each level introduces new degrees of freedom.



(a)



(b)

Figure 7: One example of a Hierarchical D-NURBS surface: (a) Different colors represent different layers, x, y, z are world coordinates; (b) The patch distribution of (a), and its subdivision process are: (1) the entire NURBS surface consists of 4×4 patches, (2) the top-left 2×2 patches are subdivided into 6×6 patches which generates a new layer $s_1^1(u,v)$, (3) the central 3×3 patches of $s_1^1(u,v)$ are further subdivided into 6×6 patches which generates a new layer $s_1^2(u,v)$, (4) we go back to layer $s^0(u,v)$ and subdivide one 1×2 patches into 6×6 patches which generates a new layer $s_2^1(u,v)$.

- More importantly, its hierarchical structure supports both global and local editing toolkits.

The trade-off for this extra geometric flexibility is that Hierarchical D-NURBS require rather complicated data structure and extra book-keeping operations. We shall take advantage of matured algorithms and latest advances in areas of Discrete Mathematics (such as Graph Theory) and Data Structure.

3.3 Physical Sculpting of Hierarchical D-NURBS

Because our Hierarchical D-NURBS can be decomposed into a linear combination of a set of layers at different levels, we shall concentrate on one layer and derive its motion equation. Analogous to [5, 13], we represent one layer as a continuous NURBS surface:

$$\mathbf{s}(u,v,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{p}_{i,j}(t)w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{i,j}(t)B_{i,k}(u)B_{j,l}(v)}, \tag{15}$$

where t denotes time, and $\mathbf{p}_{i,j}(t)$ and $w_{i,j}(t)$ are control points and weights which are time variables. The control point and weight vector $\mathbf{p}$ is the concatenation of all 3D control points $\mathbf{p}_{i,j} = [x,y,z]^T$ and weights $w_{i,j}$:

$$\mathbf{p} = \left[\begin{array}{cccccccc} \mathbf{p}_{0,0}^T & w_{0,0} & \cdots & \mathbf{p}_{i,j}^T & w_{i,j} & \cdots & \mathbf{p}_{m,n}^T & w_{m,n} \end{array}\right]^T \tag{16}$$

where ^{T} denotes matrix transposition. To facilitate the task of real-time surface editing with various toolkits, we discretize the continuous dynamic surface to a set of parametrically uniformed $g \times h$ points $\mathbf{d}$, which forms $(g-1)\times(h-1)$ quadrilateral grid. Therefore, our dynamic surface will have a dual representation in mathematical domain $(\mathbf{p}, A)$ and physical space $(\mathbf{d})$. The two formulations are tightly coupled through the non-linear equation:

$$\mathbf{d} = A\mathbf{p} \tag{17}$$

where A is the transformation matrix whose entries are discretized Jacobian quantities (see [5, 13, 17] for the details) evaluated at the sampled parametric values. The discretized dynamic model has material quantities such as mass, damping, and stiffness distribution. To improve the modeling efficiency, we consider the discretized surface as a mass-spring model (i.e., grid-points are mass points connected by a network of springs between their nearest neighbors). Alternatively, the dynamic surface can be approximated using finite element method based on $\mathbf{d}$. The finite element formulation derived from $\mathbf{d}$ are mathematically equivalent to our current implementation. We use a mass-spring model instead because of its simplicity for real-time manipulation. We formulate the motion equation of all mass-points using a discrete simulation of Lagrangian dynamics:

$$\mathcal{M}\ddot{\mathbf{d}} + \mathcal{D}\dot{\mathbf{d}} + \mathcal{K}\mathbf{d} = \mathcal{F} \tag{18}$$

The force at every mass-point in the discretized grid is the sum of all possible external forces: $\mathcal{F} = \sum f_{ext}$. The internal forces are generated by the connected springs, where each spring is modeled with force: $f = kl$. The rest length of each spring is determined upon initialization, but it is free to vary if plastic deformations or other nonlinear phenomena are more desirable.

Because all discretized points and springs are constrained by the D-NURBS surface, we shall formulate the motion equation of physical behavior for all the control points:

$$A^T\mathcal{M}A\ddot{\mathbf{p}} + A^T\mathcal{D}\dot{\mathbf{d}} + A^T\mathcal{K}\mathbf{d} = A^T\mathcal{F} \tag{19}$$

$$A^T\mathcal{M}A\ddot{\mathbf{p}} = A^T\mathcal{F} - A^T\mathcal{D}\dot{\mathbf{d}} - A^T\mathcal{K}\mathbf{d} \tag{20}$$

Therefore, we can directly compute the acceleration of the control point and weight vector based on the sculpting forces on the discretized mesh.

Note that, in general A is not constant due to the nonlinear nature of weights, and it changes every timestep. In particular, if all the weights are fixed, A becomes constant, then we can pre-compute its inverse and sculpt the surface more efficiently. In our system, users can switch the weights from fixed to unfixed or vice versa during sculpting. Our system makes use of both Conjugate Gradient(CG) method and QR decomposition method for the time integration of the above equation. CG method offers a rather general solution for solving the $N \times N$ linear system: $A \cdot \mathbf{x} = \mathbf{b}$, it is very attractive for a large, sparse system because it only references A through its multiplication with a vector, or the multiplication of its transpose with a vector. And these operations can be very efficient for a properly stored sparse matrix. In our system, however, A is large, sparse, but not square in general. In addition, $A^T\mathcal{M}A$ is large, symmetric, but not sparse. Therefore, computing $\ddot{\mathbf{p}}$ is less efficient with CG method in most cases. On the other hand, QR decomposition: $A = Q \cdot R$, where R is upper triangular and Q is orthogonal, can also be used to solve a system of linear equations. To solve $A \cdot \mathbf{x} = \mathbf{b}$ is equivalent to solve $R \cdot \mathbf{x} = Q^T \cdot \mathbf{b}$ through back substitution. QR decomposition can be applied to nonsquare matrix, so using QR, we only need to solve

$$\mathcal{M}A\ddot{\mathbf{p}} = \mathcal{F} - \mathcal{D}\dot{\mathbf{d}} - \mathcal{K}\mathbf{d} \tag{21}$$

In our Hierarchical D-NURBS system, whenever we want to manipulate the local detail of a region, we subdivide the region, which generates a new layer and introduces new control points and weights. However, control points and weights in the vicinity of boundary areas are constrained, in order to satisfy the continuity across the region boundary. So we can only modify all those free ones and keep the peripheral ones unchanged. We explicitly decompose

$$\ddot{\mathbf{d}} = A\ddot{\mathbf{p}} \tag{22}$$

into

$$\begin{bmatrix} \ddot{\mathbf{d}}_f \\ \ddot{\mathbf{d}}_s \end{bmatrix} = \begin{bmatrix} A_{ff} & A_{fs} \\ A_{sf} & A_{ss} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{p}}_f \\ \ddot{\mathbf{p}}_s \end{bmatrix} \tag{23}$$

where $\ddot{\mathbf{d}}_s$ and $\ddot{\mathbf{p}}_s$ stand for the acceleration of static surface point vector and control point and weight vector, respectively, which should be 0; $\ddot{\mathbf{d}}_f$ and $\ddot{\mathbf{p}}_f$ represent the acceleration of free surface point vector and control point and weight vector, respectively. So $A_{sf} = 0$ and

$$\ddot{\mathbf{d}}_f = A_{ff} \cdot \ddot{\mathbf{p}}_f \tag{24}$$

At time t,

$$\mathbf{d}_f(t) = A_{ff} \cdot \mathbf{p}_f(t) + A_{fs} \cdot \mathbf{p}_s \tag{25}$$

Through iteration at time $t + \Delta t$,

$$\mathbf{d}_f(t + \Delta t) = \mathbf{d}_f(t) + A_{ff} \cdot (\mathbf{p}_f(t + \Delta t) - \mathbf{p}_f(t)) \tag{26}$$

Thus we only need to compute a very small portion of the surface every timestep, it makes the sculpting more efficiently.

4 Constraints

Based on the formulation of our Hierarchical D-NURBS, we develop useful physics-based toolkits and constraint techniques to enable the efficient sculpting of Hierarchical NURBS. Before we detail our system functionalities on toolkit implementation, we shall first review the related work on geometric constraints. Many methods have been proposed to implement constraints. Hsu et al. [10] solved a spline curve for point constraints using the matrix pseudo-inverse. The pseudo-inverse has the property of finding the least-squared error when the system becomes over-constrained. Welch and Witkin [19] utilized Lagrange multipliers to enforce a least-square solution to a constraint matrix. Moreton and Sequin [11] used a minimum-energy network to optimize a system of linear and nonlinear constraints. Terzopoulos et al. [15] used the penalty method to drive a dynamic deformation for animation. Qin and Terzopoulos [13] used linear constraint techniques to deform physical models for design purposes. Platt and Barr [12] discussed various constraint methods for deformable models including the penalty method, reaction constraints, Lagrange constraints, and augmented Lagrange constraints. Among various techniques to handle constraints, penalty methods exhibit the property of simplicity, but suffer from inexact solutions and the need for small timesteps. Reaction constraints improve the penalty method by enforcing constraints exactly in the presence of external forces.

4.1 Normal Constraint

We can manipulate the surface normal at arbitrary point. The normal of the surface point on a continuous surface can be approximated by averaging all the normals from its surrounding triangles:

$$\mathbf{n}_d = \frac{1}{n} \sum_{i=0}^{n-1} N_i \tag{27}$$

where $\mathbf{n}_d$ is the normal of the surface point, and N_i is the normal of a surrounding triangle. When users modify the point normal to $\overline{\mathbf{n}}_d$, our system will convert the normal constraint into additional external forces applied at the vicinity of the surface point, then the surface will deform its shape and gradually converge to its equilibrium with the new normal vector through the computation of the following nonlinear equation. We use the minimum-energy method, the energy due to normal manipulation is $E = \frac{k_n}{2}(\mathbf{n}_d - \overline{\mathbf{n}}_d)^2$. The force $\mathbf{f}$ that arises from this energy and exerts on each neighboring point is $\mathbf{f}(\mathbf{x}) = -\frac{\partial E}{\partial \mathbf{x}}$, where the point vector $\mathbf{x} = a, b, c, d, e, f$. Fig. 8 illustrates a D-NURBS patch, which can be considered as a layer at any level of the Hierarchical D-NURBS structure, consisting of 9×9 control points. This layer is further discretized to 19×19 surfaces points. The peripheral 6×6 control points are static in order to ensure geometric continuity across the boundary, so only the central 3×3 control points are free to move in this example.

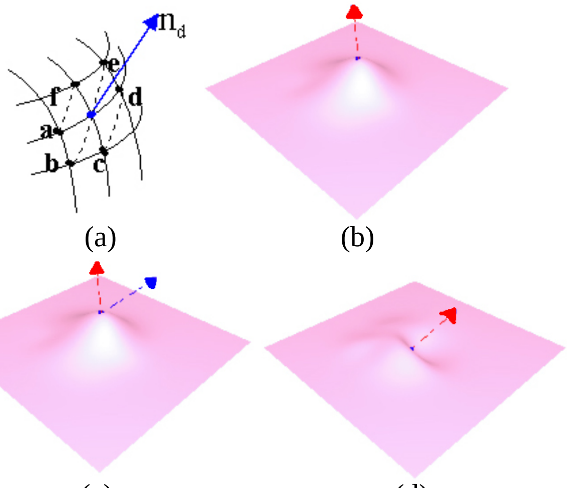
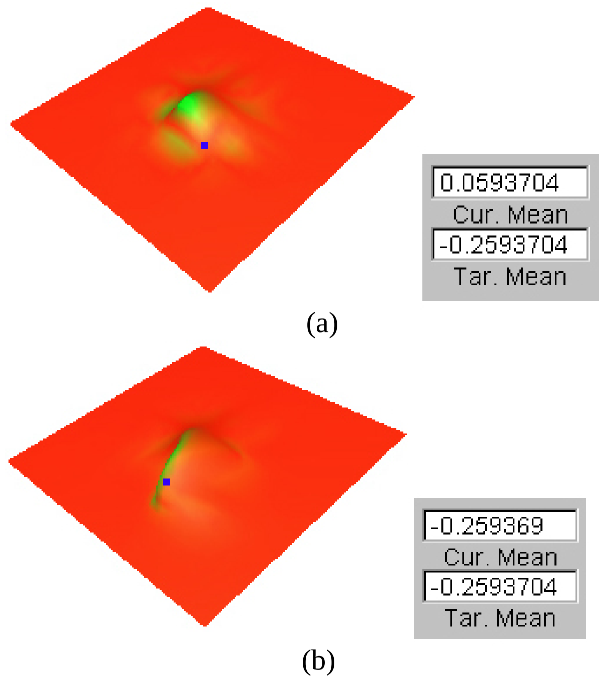


Figure 8: Normal constraint: (a) Changing the point normal will exert additional forces at the vicinity of the surface point: a,b,c,d,e,f; (b) The current normal (represented as the red arrow) is (-0.04,0.07,-0.92); (c) The blue arrow represents the desired normal which is (0.45,-0.73,-0.52), the red arrow represents the current normal; (d) The actual normal (red arrow) after surface deformation is (0.36,-0.64,-0.63).

4.2 Curvature Constraint

Users can also intuitively change the shape of the D-NURBS by modifying the mean curvature at arbitrary point. We approximate the mean curvature at arbitrary point by computing

$$c_d = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{NE - 2MF + LG}{EG - F^2} \tag{28}$$



(a)

0.0593704	Cur. Mean
-0.2593704	Tar. Mean

(b)

-0.259369	Cur. Mean
-0.2593704	Tar. Mean

Figure 9: Curvature constraint: (a) Curvature map, green represents high curvature, red represents low curvature; the blue point is the selected point with which users want to modify its curvature, the accompanying table lists the current and target curvatures of the blue point; (b) The curvature map after users modify the blue point's curvature; the accompanying table shows that the current curvature equals to the target curvature after surface deformation.

where $L = \mathbf{n}\mathbf{x}_{uu}, M = \mathbf{n}\mathbf{x}_{uv}, N = \mathbf{n}\mathbf{x}_{vv}, E = \mathbf{x}_u\mathbf{x}_u$, $F = \mathbf{x}_u\mathbf{x}_v$, and $G = x_v x_v$; $\mathbf{n}$ is the surface normal at the point. When users change the curvature to $\bar{c}_d$, our system will convert the curvature constraint into additional external forces applied on the neighborhood of the surface point. We use the minimum-energy method in our system, the energy due to curvature deformation is $E = \frac{k_c}{2}(c_d - \bar{c}_d)^2$. The force $\mathbf{f}$ that arises from this energy and exerts on every neighboring point is $\mathbf{f}(\mathbf{x}) = -\frac{\partial E}{\partial \mathbf{x}}$, where $\mathbf{x}$ is the neighboring point vector.

Note that the neighboring points are connected by springs in our mass-spring system, therefore, it is almost impossible to enforce the exact curvature constraint unless we temporarily disable the forces resulted from all connected springs. For sculpting purpose, we can first set the curvature to what we want and disable spring forces temporarily. Then when the curvature reaches the desired value, we can re-enforce the spring forces exerted from the neighboring points. Fig. 9 shows a D-NURBS surface, which can be considered as a layer at any level of the Hierarchical D-NURBS. This surface consists of 11×11 control points and is discretized into 25×25 surfaces points. The peripheral 6×6 control points become static to ensure geometric continuity across the surface boundary, only the central 5×5 control points are free to move.

4.3 Curve Constraint

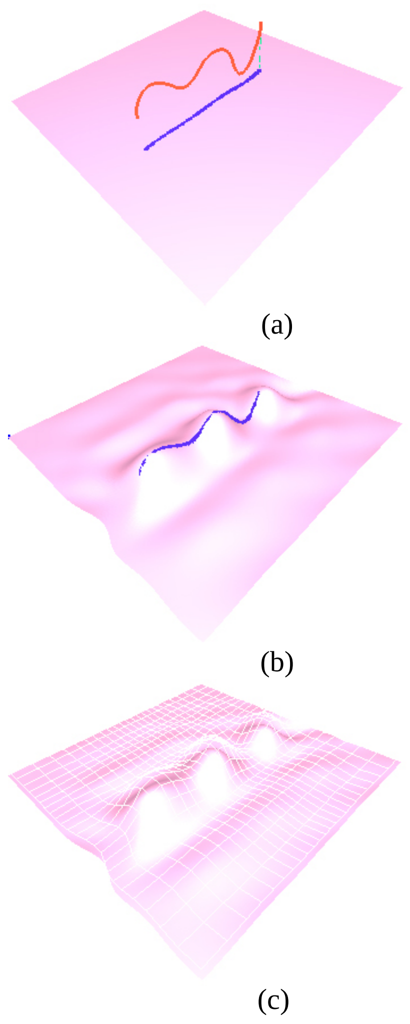


Figure 10: Curve constraint: (a) Select a spline curve tool colored in red; (b) The surface deforms in terms of the shape of the curve tool, the curve colored in blue is the corresponding curve on the D-NURBS surface; (c) The new, deformed NURBS surface satisfying the constraint of the curve feature.

Although the aforementioned point-based sculpting provides designers useful manipulation tools, point editing is less effective, hence less appealing, especially when users are faced with complicate design requirements. To ameliorate, we develop sculpting tools that afford the intuitive specification of curve-based constraints. First of all, users can pick a curve tool from the system menu (or define their own curve tool by specifying a set of line segments). Second, users can apply the curve tool to deform the surface, the

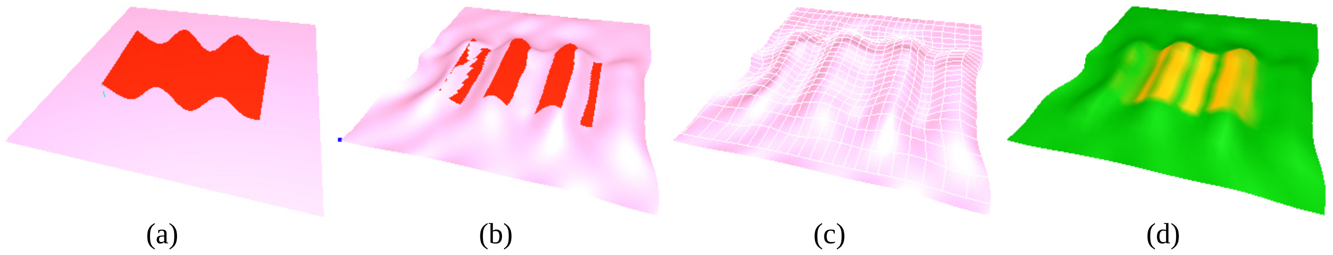


Figure 11: Surface constraint: (a) Pick a spline surface tool as a sculpting template; (b) The surface deforms in terms of the shape of the surface tool, the sculpting template is colored in red; (c) The new, smooth surface subject to the surface constraint; (d) The error map: the area colored in green represents small error distribution, and the area colored in yellow represents large error distribution. The error shown in (d) is characterized by the distance between the tool template and the deformed D-NURBS surface. The maximum error in this example is 1.88, and the average error is 0.60.

contacting region on the surface will be sculpted in terms of the tool shape, and the rest region of the surface will deform accordingly subject to physical laws and material properties. Our sculpting algorithm is:

- Discretize a continuous curve tool into a number of sample points,
- Raycast a line from every curve point along the sculpting direction (similar to parallel projection),
- Compute the intersection point of the line with the NURBS surface and also retrieve the corresponding triangle from the D-NURBS discretized grid,
- If the distance between a point on the curve tool and its corresponding triangle on the D-NURBS is less than a user-defined threshold, our system will connect the two points with a spring (whose rest length is zero and whose stiff constant can be set up interactively), which will attract the triangle along the raycasting direction of the curve tool towards the destination curve point.
- Repeat this process for every curve point, therefore, a feature-based force distribution will be applied to the D-NURBS surface at the corresponding region through the spring attachment between every curve point and the corresponding intersection point.

In our dynamic framework, a curve feature is converted into the external force distribution applied on the D-NURBS surface, the surface will then deform subject to the curve constraint. At the equilibrium, the D-NURBS surface exhibits the same feature as that of the curve sculpting tool.

4.4 Surface Constraint

Certain surface models may exhibit special features in specific regions, hence sometimes it is more desirable to make region-based editing tools available to designers towards the ultimate goal of feature-based design. Analogous to the aforementioned curve tool, our system can map a user-specified area onto a region of interest within the D-NURBS surface. Similar to our previous discussion about the curve sculpting tool, users can select a surface tool from our system menu (or interactively define a surface tool as a collection of connected polygons). Then, users can interactively move the designated surface tool to deform the corresponding region of the D-NURBS surface. The attached region in the D-NURBS surface will be sculpted intuitively in terms of the geometric shape of the tool template, and the other region of the D-NURBS will be deformed accordingly in the physically realistic fashion subject to other relevant geometric constraints. Throughout the enforcement of the surface constraint, our sculpting algorithm functions in a similar way as that of the curve tool. we use the similar sculpting algorithms explained above.

5 System Implementation

We have developed a prototype software environment that permits users to intuitively and interactively manipulate Hierarchical D-NURBS surfaces (either locally or globally) via various force-based sculpting tools and constraints. Our system is written in C++ and can run in both MS Windows and Unix operating systems.

With our Hierarchical D-NURBS, designers can interactively undertake local/global modifications on a D-NURBS surface by specifying a region of any level within the D-NURBS hierarchy and selecting an appropriate tool from the menu of various toolkits. Within physics-based modeling framework, users do not need to work with the mathematical parameters such as control points, weights and knot sequence directly because they are less intuitive and require

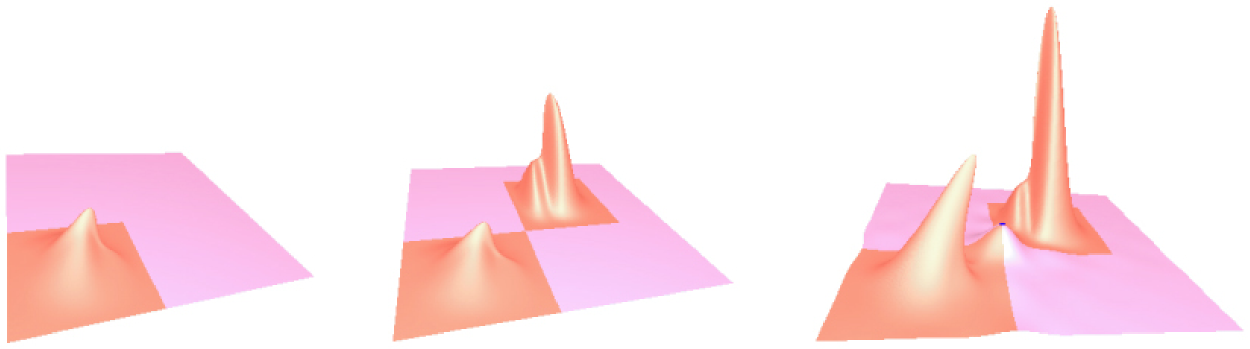


Figure 12: Examples of dynamic sculpting with Hierarchical D-NURBS.

strong mathematical sophistication. Instead, the desired shape features can be automatically achieved through the direct manipulation using force-based tools and constraints. The appropriate value of both control points and weights is evolved continuously subject to the time integration of D-NURBS dynamics.

Currently, we only employ the forward Euler method to solve the Lagrangian dynamics, and this computing scheme is inevitable to introduce errors (and sometimes instability) into our numerical simulation. Numerical errors are due to either a coarse timestep or a low-resolution discretization. The timestep is normally very small. If the model is very complicated, the timestep may become large in order to offset the large amount of numerical computation, and in this case, errors can easily creep into the simulation unless an adaptive timestep is used. However, these errors do not necessarily deteriorate the task of surface design since the system is continuously evolving towards an equilibrium of energy minimization. Temporary inconsistencies in dynamics do not appear to have a negative effect in our system towards the final stable shape. Meanwhile, coarse discretization also leads to potential errors, so an accurate bound for surface sampling rates is necessary in order to quantify the error effect on the surface quality.

Currently, our sculpting system based on Hierarchical D-NURBS only focuses on a single D-NURBS surface element at arbitrary level within the D-NURBS hierarchy. In the future, we will generalize our system and extend its functionalities to support the simultaneous sculpting of multiple NURBS surface elements (possibly from different layers). These more advanced and complex tasks are extremely useful and are far from trivial. Another challenging aspect is to effectively handle the detection/avoidance of self-collision during the sculpting session. This functionality will provide more realistic effects for D-NURBS and may facilitate our system to support the realistic cloth simulation and mechanical part assembly using Hierarchical D-NURBS.

6 Conclusion

We have proposed and formulated a new shape modeling representation—Hierarchical D-NURBS, and have developed a prototype software environment that supports the direct manipulation and interactive sculpting of Hierarchical D-NURBS via real-time physical interaction. Our novel formulation applies the knot-insertion algorithm only on the localized region of NURBS parameterization, producing new degrees of freedom whenever necessary and ameliorating the limitation of standard knot-insertion techniques. Through the hierarchical structure and physics-based modeling, we have further extended the geometric coverage of standard NURBS, making them more flexible and powerful in shape modeling, geometric design, and interactive graphics.

Our experimental software provides users a hierarchical sculpting interface for D-NURBS editing and a wide range of powerful toolkits such as point manipulation, normal editing, curvature control, feature-based curve constraint, as well as feature-based surface constraint. These new, physics-based capabilities permit users to model and manipulate D-NURBS surfaces intuitively. Our experiments have shown that the hierarchical structure of D-NURBS and the novel physics-based force tools and constraints offer users more freedom and a more natural interface to effectively manipulate D-NURBS surfaces in order to satisfy a set of design criteria and functional requirements.

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.H. Bartels, J.C. Beatty, and B.A. Barsky, "An Introduction to Splines for Use in Computer Graphics and Geometric Modeling," *Morgan Kaufmann Publishers, Inc*, 1987.

[2] M.I.G. Bloor and M.J. Wilson, "Representing PDE Surfaces in Terms of B-splines," *Computer-Aided Design*, Vol.22, no.6, 324-331, 1990.

[3] G. Celniker and D. Gossard, "Deformable Curve and Surface Finite Elements for Free-Form Shape Design," *Computer Graphics*, Vol.25, 257-266, 1991.

[4] G. Celniker and W. Welch, "Linear Constraints for Deformable B-spline Surface," *Computer Graphics (Proceedings of the 1992 Symposium on Interactive 3D Graphics*, Vol.26, no.2, 165-170, 1992.

[5] F. Dachille, H. Qin, A. Kaufman and J. El-Sana "Haptic Sculpting of Dynamic Surfaces," *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, 103-110, 1999.

[6] D.R. Forsey and R.H. Bartels, "Hierarchical B-Spline Refinement," *Computer Graphics*, Vol.22, no.4, 205-212, 1988.

[7] C. Grimm and M. Ayers, "A Framework for Synchronized Editing of Multiple Curve Representation," *Computer Graphics Forum (Proceedings of EUROGRAPHICS'98)*, Vol.17, no.3, 1998.

[8] C. Gonzalez-Ochoa and J. Peters, "Localized-Hierarchy Surface Splines(LeSS)," *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, 7-16, 1999.

[9] M. Halstead, M. Kass, and T. DeRose, "Efficient, Fair Interpolation Using Catmull-Clark Surfaces," *Computer Graphics*, Vol.27, 35-44, 1993.

[10] W.M. Hsu, J.F. Hughes, and H. Kaufman, "Direct Manipulation of Free-Form Deformations," *Computer Graphics*, Vol.26, 177-184, 1992.

[11] H.P. Moreton and C.H. Sequin, "Functional Optimization for Fair Surface Design," *Computer Graphics*, Vol.26, 167-176, 1992.

[12] J.C. Platt and A.H. Barr, "Constraint Methods for Flexible Models," *Computer Graphics*, Vol.22, 279-288, 1988.

[13] H. Qin and D. Terzopoulos, "D-NURBS: A Physics-Based Framework for Geometric Design," *IEEE Transactions on Visualization and Computer Graphics*, Vol.2, no.1, 85-96, 1996.

[14] P.J. Stewart and K.P. Beier, "Direct Manipulation of Free-Form Curves with Generalized Parametric Basis Functions," *Technical report, Personal Communication*, 1998.

[15] D. Terzopoulos, J. Platt, A. Barr and K. Fleischer, "Elastically Deformable Models," *Computer Graphics*, Vol.21, 205-214, 1987.

[16] D. Terzopoulos and K. Fleischer, "Deformable Models," *The Visual Computer*, Vol.4, no.6, 306-331, 1988.

[17] D. Terzopoulos and H. Qin, "Dynamic NURBS with Geometric Constraint for Interactive Sculpting," *ACM Transactions on Graphics*, Vol.13, no.2, 103-136, 1994.

[18] J.A. Thingvold and E. Cohen, "Physical Modeling with B-spline Surfaces for Interactive Design and Animation," *Computer Graphhics (Proceedings of the 1990 Symposium on Interactive 3D Graphics*, Vol.24, no.2, 129-137, 1990.

[19] W. Welch and A. Witkin, "Variational Surface Modeling," *Computer Graphics*, Vol.26, no.2, 157-166, 1992.

[20] J.M. Zheng, K.W. Chan and I. Gibson, "A New Approach for Direct Manipulation of Free-Form Curve," *Computer Graphics Forum (Proceedings of EUROGRAPHICS'98)*, Vol.17, no.3, 327-334, 1998.

[21] J.M. Zheng, K.W. Chan and I. Gibson, "Surface Feature Constraint Deformation for Free-form Surface and Interactive Design," *Proceedings of Fifth Symposium on Solid Modeling and Applications*, 223-233, 1999.