Interactive Virtual Tools for Manipulating NURBS Surfaces in a Virtual Environment

Brian P. Perles
DELMIA CORP,
5500 New King St.,
Troy Michigan, 48098
e-mail: Brian_Perles@delmia.com

Judy M. Vance
Department of Mechanical Engineering,
Virtual Reality Applications Center,
Iowa State University,
Ames, Iowa 50011
e-mail: jmvance@iastate.edu

DN-Edit is a virtual environment developed to allow the manipulation of non-uniform rational b-spline (NURBS) surfaces using virtual shaping tools. NURBS have become the industry standard for representation of free-form curves and surfaces. The contribution of the work presented here is in the development of shaping tools which are used to operate directly on the NURBS data and change the shape of the surfaces in a virtual environment. These shaping tools allow surface manipulations to be made using methods that match the shaping of real malleable objects. The virtual shaping tools are three-dimensional shapes that are controlled through a six degree-of-freedom tracking system that converts user hand motions into computer input. The NURBS surface updates itself in real time due to the effect of the tools on the surface. The new shape of the surface is dependent on the position and orientation of the shaping tool relative to the surface. Constraint-based surface manipulation is used to obtain multiple point direct manipulation of NURBS surfaces. In addition, computational methods to allow the user to have direct control of the first derivative of the surface over an area are implemented. This application has been developed using the C2 software libraries and Iowa State University's C2 surround screen virtual environment. [DOI: 10.1115/1.1464132]

Introduction

Virtual reality (VR) is an interactive, immersive computer interface system where the input and display is linked to the user's head and hand position, Durlach and Mavor [1]. This has the potential for allowing users to interact with computer generated objects as if they are physical three-dimensional (3D) objects. Computer aided drafting (CAD) models can be displayed at a one-to-one scale with the world, allowing users to place themselves in the world with the object and compare sizes and proportions. Design reviews conducted in a virtual environment allow interaction with designs without the cost of building physical mockups.

Design reviews usually result in suggested changes to the design. It would be advantageous to be able to edit the CAD data from within the virtual environment during the design review and see the results of the changes immediately. The virtual environment's six degree-of-freedom (DOF) input devices provide an interface to the computer that is not available on the standard desktop computer. The application presented in this paper, DN-Edit, is designed to allow interactive editing of predefined non-uniform rational b-spline (NURBS) surfaces from within a virtual environment. NURBS are a commonly used CAD format for free-form surfaces. The position and orientation of the user's hand in space is what controls the position and orientation of the virtual shaping tools. The workpiece is modeled as a NURBS surface and the user-controlled virtual shaping tools interact with the surface, making the workpiece respond like a malleable object. Methods are also provided for controlling the first derivative of the surface directly.

VR CAD Programs. There are two widely used methods of representing a 3D shape in a Computer-aided design (CAD) program: solid modeling and free-form surfaces. Commercial CAD packages make use of both these modeling types. Research-based CAD applications tend to focus on developing new techniques for either solid modeling or free-form surface modeling because the data storage and manipulation methods needed for each are so different.

Developers of VR-based solid modeling programs have focused on providing interfaces for defining Boolean operations and primitives. Hand motions in 3D space are used to input extrusion lengths, spoken words are recorded to enter numerical data, and head motions replace manipulating the view through zoom and pan commands. These intuitive interfaces have the potential of making the input of a computer model as intuitive as assembling wooden blocks to create an object, with the result being increased use of CAD programs during the conceptual design stage.

JDCAD [2] was developed with the goal of providing faster interaction with a solid modeling program. Modeling is performed by creating and attaching primitives. Liang and Green investigated many methods of interacting with the solid models which utilized the extra degrees of freedom provided by a 6 DOF wand. Available to the user are a wide array of virtual menus, dials, and handles for controlling a primitive's shape and position.

COVIRDS [3] is a VR solid modeling application designed to facilitate conceptual design in a virtual environment and to make 3D sketching with shape elements accessible to engineers, designers, and others. A solid modeling engine based on a shape-graph is used. Modeling operations are selected and quantified with voice input and positioned with a virtual ray cast from a 6 DOF glove. The command structure is patterned after verbal descriptions to make control intuitive.

Free-form surfaces are mathematical curves and surfaces that represent continuous shapes with calculable derivates [4]. Free-form surfaces are used to represent curves or surfaces with non-constant curvature. These surfaces may also have stringent requirements for surface properties. A mathematical model of a surface is stored by the computer and usually converted to polygons when rendered. Free-form surfaces are used to model automobile bodies, ship hulls, appliance housings, and turbine blades.

Many VR free-form surface manipulation programs try to provide interaction capabilities which behave similar to manipulating deformable objects in the physical world. Common paradigms include sculpting stone or clay. Gaylean and Hughes [5] created an early 3D-input sculpting program that used a voxel represen-

Contributed by the Design Automation Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received Nov. 2000. Associate Editor: G. M. Fadel.

tation. A voxel array is a very fine 3D volume of points, each of which can only be on or off. The piece of 'virtual clay' is a polygonal model generated by meshing the boundary between the on and off voxels. Attached to a 6 DOF tracker are software-based 'sculpting' tools that add or remove material (turn voxel points on or off), sand the surface (average the points in an area to smooth the surface), or melt the surface (slowly remove material from an area). The surface responds in real time to the tools, so the tools can be dragged across the surface in ways very similar to real clay sculpting. This method provides a very good simulation of clay and stone sculpting, but its usefulness for engineering applications is restricted because of the lack of surface representation in the voxel model that transfers easily into a CAD program.

3-Draw [6] uses a 6 DOF input device to create and modify 3D splines. A pair of 6 DOF trackers are used, one in each of the user's hands. One traces out splines in 3D space and the other controls the orientation of the modeling world. Software tools are provided for trimming and joining splines. The goal of the application is to generate splines that could eventually be meshed into surfaces and then used in the early stages of conceptual design. The use of a 6 DOF interface means that the 3D shapes can be created without the 3D to 2D mapping required when using a desktop mouse.

Furlong [7] created a Bezier free-form deformation-based sculpting program for use in a CAVE-type virtual reality environment. The workpiece is a polygonal surface embedded in a Bezier volume. Individual points are pushed or pulled on the surface and the Bezier volume provides spline-based continuity for the rest of the polygonal surface. The surface can also be painted using the 'paint brush' tool to give the model color. The program is designed for sketching free-form shapes during the conceptual design stage. A limitation of the program is that only single points on the surface can be directly controlled at a time. The blending between the edited point and the undisturbed portions of the surface is determined by the parameters of the Bezier volume.

Yeh and Vance [8] used a NURBS based free-form deformation volume to perform design shape changes on polygonal and finite element analysis (FEA) mesh models. The model geometry, along with the stress results from an initial FEA, are displayed in the virtual environment. The user defines and positions a NURBS volume around features of the geometry model which are allowed to change shape. The user then changes the shape of the NURBS volume in the virtual environment. When the NURBS volume is changed, the underlying location of each node in the finite element model is changed. New approximate stresses are calculated based on the nodal position change and are displayed on the new model geometry. The stress approximation calculations are performed using first order stress sensitivities which come from the FEA. The program also supports indirect and single point direct manipulation of the NURBS volume. This program allows the designer to experiment with the shape of an object while also examining the effect that shape changes have on the stresses generated in the object.

Direct Manipulation of NURBS Curves and Surfaces. The location of a point on a NURBS surface, S, at a parametric location (u,v), is calculated from a set of control points, P, weights, w, and basis functions, N.

$$S(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}} \quad (1)$$

Here, p and q are the degrees of the surface and $n+1$ and $m+1$ are the number of control points in the respective parametric directions. The values of u and v vary between 0 and 1.

Manipulation methods for NURBS surfaces are categorized by the way a designer interacts with the surface. The most basic method is to edit the individual control points. Since a NURBS surface does not pass through its control points, this is called indirect manipulation. Direct manipulation involves manipulating the surface directly and using a mathematical algorithm to determine the changes that are needed to the control point mesh. Direct manipulation can further be divided into single point direct manipulation and multiple point direct manipulation. Single point manipulation lets the designer only edit a single point on the surface at a time and uses the properties of NURBS to determine how the rest of the surface responds. Multiple point direct manipulation lets the user define a new shape for an area of the surface and the existing surface is fit to the new shape. The user is still limited by the properties of the surface, but a higher level of control is available. Multiple point manipulation does require a longer setup time because multiple change points have to be specified on the surface.

Shape operators [9] is a basic method for performing multiple point direct manipulation of NURBS curves and surfaces. First, a new shape is defined, separate from the NURBS. Then, knot refinement is performed on the area until there are enough control points to represent the new shape. The new shape is then applied to the curve using control point interpolation. Shape operators allow any shape to be applied to a NURBS curve or surface. It is, however, computationally intensive and not suitable for real-time manipulation because the NURBS is constantly undergoing knot insertion or knot removal.

Constraint-based manipulation is the process of manipulating a NURBS curve, surface, or volume by specifying a set of constraints. A constraint consists of a change to the NURBS's position or derivative at an arbitrary point. To find the resulting changes to the control point mesh, each constraint is converted into a linear equation. The set of constraints is then solved simultaneously. Since this set of linear equations can be over- or underdetermined, an appropriate solver is needed.

The term 'constraint-based curve manipulation' was coined by Fowler and Bartels [10] when they published the equations for manipulation of the B-spline curve as well as the first and second derivatives of the curve. Hsu et al. [11] proposed the manipulation of B-spline volumes with this method in a paper on direct manipulation of free-form deformation volumes. Constraint-based manipulation is a powerful method for shaping NURBS because it allows the re-interpolation of a local area without requiring the re-interpolation of the entire NURBS. Obtaining the correct degree or control point spacing in the area being edited is not as critical as when using shape operators.

Overview of the Application

DN-Edit is a VR-based surface manipulation program designed to provide a higher level of control than what is currently available in free-form surface editing programs. The application provides the user with shaping tools that can deform a surface into a given shape. This provides a higher level of surface control when compared to single-point control provided in previous applications. A shaping tool provides for control over the surface normals, giving direct control over a NURBS surface's first derivative.

DN-Edit is implemented on an SGI Octane computer equipped with an Octane Channel Option which is used to generate two display channels to feed a stereo image to the Virtual Research V8 head mounted display (640×480 resolution). One Ascension Flock of Birds magnetic tracker is attached to the helmet to track the user's head position. The user holds another tracker in his/her dominant hand and holds an untracked four-button wand in the other hand (Fig. 1).

The position tracker for the wand is normally mounted inside the wand so that the position of the wand and manipulation of the buttons can be accomplished with one hand. It was found that the size and shape of the wand did not allow the user to achieve all possible wand orientations in this application, therefore, the posi-

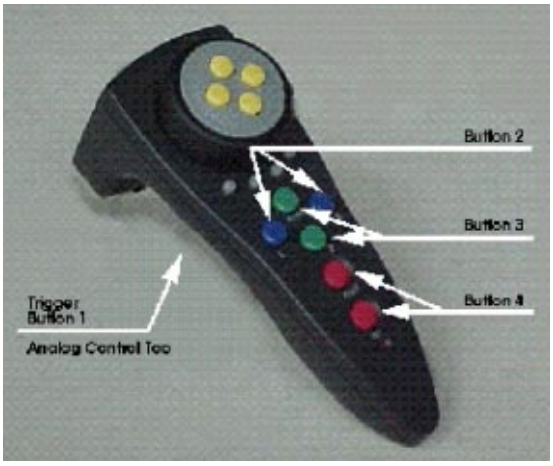

Fig. 1 The wand

tion input and button input were separated into two devices. The user holds the position tracker in one hand which controls the location of the virtual cursor in the display system. The wand is held in the other hand and is only used for button input to the application. The C2 software libraries are used to manage the viewing display and the input devices, and OpenGL is used for rendering. The C2 libraries provide functions to manage the viewing and interaction in the C2 virtual reality projection room at Iowa State University (Fig. 2). In additional to managing C2-type systems, the C2 libraries support helmet display and tracking. NURBS surfaces of any size and degree can be manipulated with DN-Edit. Surfaces are loaded from a file consisting of a list of control point locations and weights.

In DN-Edit, shaping tools match the position and orientation of the tracker held in the user's hand. When a tool intersects the surface, the surface responds to the tool in real time and deforms based on the tool's type. Most tools mimic the real-world interaction between a deformable object and a hard tool. This gives the user a familiar interface to work with. DN-Edit currently provides three different types of tools.

• Point tool: The point tool is the most basic tool, allowing a single point on the surface to be pushed in or pulled out (Fig. 3). The action of this tool most closely resembles the effect of a sharp stylus, which is poked into or pulled across a surface.

• Block tool: The block tool controls an area of the surface. This tool is a cube with one side designated as an active face. Any part of the surface that falls within the tool is pushed to the active face (Fig. 4). This produces the effect of molding clay with a solid block.

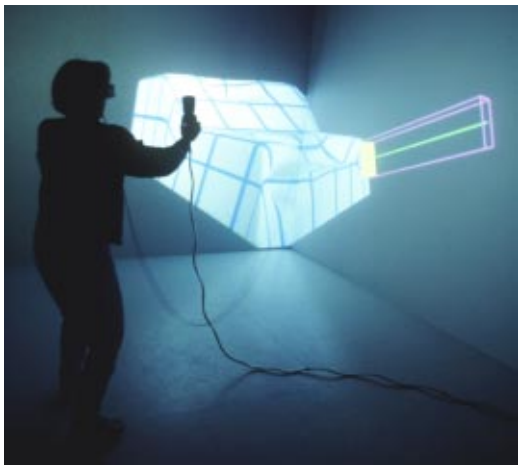

Fig. 2 DN-edit inside the C2 virtual environment

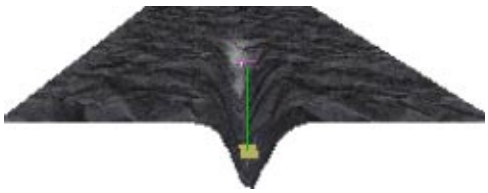

Fig. 3 The point tool

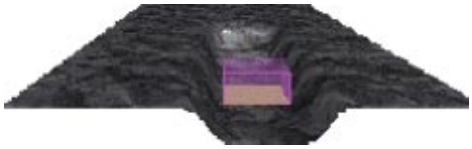

Fig. 4 The block tool

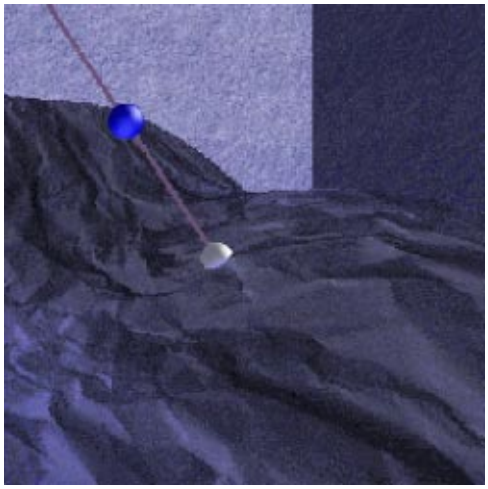

Fig. 5 The normal tool

• Normal tool: The normal tool controls a surface's derivatives to alter its normal vector at a point (Fig. 5). The tool is represented by a marker sphere that is placed on the surface. The surface normal at that point is then displayed and the user can grab and drag the normal to a new angle in 3D space. In real-time the surface updates its shape in the area around the tool to match the new normal vector.

In DN-Edit, pressing the second button on the wand cycles through the tools. Several different copies of the block tool of different sizes and proportions are pre-built. Pressing the trigger of the wand activates the currently selected tool. The surface is updated with respect to the effect of the tool in real-time. Because the tools have an instantaneous effect on the surface, the surface is edited by making sweeping motions across the surface. Only when very accurate changes are need is the tool first positioned on the surface and then activated.

Surface-Tool Interaction

The surface deformation algorithm used in DN-Edit is divided into two stages. First, the interaction between the surface and the tool is quantified. Then the program solves for the changes needed in the control point mesh in order for the surface to conform to the tool.

For the application to understand the effect the user wishes to have on the surface, the interaction between the tool and the sur-

face must be determined. This process results in the creation of a set of constraints. A constraint is a point on a NURBS surface and a vector representing the change in location of that point due to the application of the shaping tool. How these constraints are created depends on which tool is being used.

Point Tool. The point tool affects only a single point on the surface at a time; therefore only a single constraint is created each time-step. The tool consists of a line in space and a point on that line. Any location on the surface that the line intersects is moved along the line to the point. A cube is drawn around the tool's line to help the user visualize the location and limits of the tool.

Block Tool. To affect an area of the surface, the block tool must generate multiple constraints each time step. The distance between these constraints on the surface must be large enough so that the surface is not over-determined and yet small enough so that shape of the tool can be accurately represented. The distance between constraints, ΔS, along a constant u or v line should be no less than the length of the tool, L_{tool}, divided by the number of active control points, or control points that have non-zero basis functions in the area where the tool contacts the surface. The number of active control points along a line of constant u or v is one less than the degree of the surface, p, plus the number of knot spans the tool contacts, k_{tool}. The distance equation is given here:

$$\Delta S \geqslant \frac{L_{\text{tool}}}{p + k_{\text{tool}} - 1} \tag{2}$$

Two algorithms were tried for automatically creating the constraints. The first was a logical extension of the point tool and consisted of a block containing a grid of point tools. A difficulty encountered with implementing this method is that calculating the intersection between a line and a NURBS surface is computationally expensive and performing this calculation multiple times each time step reduced the frame rate below acceptable limits. In addition, the constraint spacing was difficult to control since it was a function of both the spacing of the lines in the block and the orientation of the tool with respect to the surface.

The solution to this problem was to create a fixed number of pre-computed points on the NURBS surface. These points, called trigger points, are created by dividing up the surface into a grid with constant u and v spacing. When the trigger-block tool intersects the surface, any trigger point that falls within the tool turns into a constraint. The trigger point spacing then controls the constraint spacing and Eq. (2) can be used to space the trigger points. Since the (x, y, z) location of each trigger point is pre-computed, surface intersection tests are reduced to a simple check to see which trigger points fall inside the tool volume.

Normal Tool. The normal tool controls the surface's shape based on equations for a surface's derivative instead of position. The surface normal, N, is the cross product of the partial derivatives of the NURBS surface in the u and the v directions (Eq. (3)).

$$N = \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v} \tag{3}$$

A surface normal that matches the user's requested normal is created on the surface by solving for the changes needed in $\partial S/\partial u$ and $\partial S/\partial v$. The program currently keeps one derivative constant while solving for the other. The derivative that is kept constant is switched each time-step. This is non-optimal, but is sufficient at this time as a proof of concept for interactive editing of surface derivatives.

In the following example, a new $\partial S/\partial v$ value is solved for when based on a new N and the current $\partial S/\partial u$. The first step is to separate Eq. (3) into three linear equations:

$$\begin{aligned} N_x &= \frac{\partial S_y}{\partial u} \cdot \frac{\partial S_z}{\partial v} - \frac{\partial S_z}{\partial u} \cdot \frac{\partial S_y}{\partial v} \\ N_y &= \frac{\partial S_z}{\partial u} \cdot \frac{\partial S_x}{\partial v} - \frac{\partial S_x}{\partial u} \cdot \frac{\partial S_z}{\partial v} \\ N_z &= \frac{\partial S_x}{\partial u} \cdot \frac{\partial S_y}{\partial v} - \frac{\partial S_y}{\partial u} \cdot \frac{\partial S_x}{\partial v} \end{aligned} \tag{4}$$

These equations are then arranged in the matrix form: $b = Ax$.

$$\begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\partial S_z}{\partial u} & \frac{\partial S_y}{\partial u} \\ \frac{\partial S_z}{\partial u} & 0 & -\frac{\partial S_x}{\partial u} \\ -\frac{\partial S_y}{\partial u} & \frac{\partial S_x}{\partial u} & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial S_x}{\partial v} \\ \frac{\partial S_y}{\partial v} \\ \frac{\partial S_z}{\partial v} \end{bmatrix} \tag{5}$$

The A matrix is singular, so traditional methods such as Gauss elimination can not be used to solve for $\partial S/\partial v$. The 3D-interaction method used to control the surface normal provides the extra information needed to solve this problem. Since the user grabs and drags the surface normal in real time, the normal, and therefore $\partial S/\partial v$, will only change by a small amount each frame. Since a current $\partial S/\partial v$ is known, it can be used as a starting point to find the new $\partial S/\partial v$. The equation for the normal can be rewritten to find the change between the old $\partial S/\partial v$ and the new $\partial S/\partial v$.

The magnitude of $\partial S/\partial u$ and $\partial S/\partial v$ controls the spacing of the control points. To maintain the current control point spacing, $\partial S/\partial v$ must change as little as possible while still meeting the new surface normal requirements. The pseudoinverse is a solution matrix that can be found for singular matrixes and contains the smallest distance solution. It is perfectly suited for finding $\Delta \partial S/\partial v$. The pseudoinverse is discussed later in this paper.

To solve for the change in $\partial S/\partial v$, in Eq. (3) is rewritten to include a new variable, $\partial S/\partial v$, and the new normal vector, N_{new}.

$$N_{new} = \frac{\partial S(u,v)}{\partial u} \times \left(\frac{\partial S(u,v)}{\partial v} + \Delta \frac{\partial S(u,v)}{\partial v} \right) \tag{6}$$

Following the steps listed previously, the new set of equations is written in matrix form:

$$\begin{bmatrix} N_x - \frac{\partial S_y}{\partial u}\frac{\partial S_z}{\partial v} + \frac{\partial S_z}{\partial u}\frac{\partial S_y}{\partial v} \\ N_y - \frac{\partial S_z}{\partial u}\frac{\partial S_x}{\partial v} + \frac{\partial S_x}{\partial u}\frac{\partial S_z}{\partial v} \\ N_z - \frac{\partial S_x}{\partial u}\frac{\partial S_y}{\partial v} + \frac{\partial S_y}{\partial u}\frac{\partial S_x}{\partial v} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\partial S_z}{\partial u} & \frac{\partial S_y}{\partial u} \\ \frac{\partial S_z}{\partial u} & 0 & -\frac{\partial S_x}{\partial u} \\ -\frac{\partial S_u}{\partial u} & \frac{\partial S_x}{\partial u} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta \frac{\partial S_x}{\partial v} \\ \Delta \frac{\partial S_y}{\partial v} \\ \Delta \frac{\partial S_z}{\partial v} \end{bmatrix} \tag{7}$$

Equation (7) can be solved for $\Delta \partial S/\partial v$ and a single constraint is created that contains the point (u,v) and the change in $\partial S/\partial v$. The set of equations in terms of $\Delta \partial S/\partial u$ (Eq. 8) is developed similarly:

$$\begin{bmatrix} N_x - \frac{\partial S_z}{\partial u}\frac{\partial S_y}{\partial v} + \frac{\partial S_y}{\partial u}\frac{\partial S_z}{\partial v} \\ N_y - \frac{\partial S_x}{\partial u}\frac{\partial S_z}{\partial v} + \frac{\partial S_z}{\partial u}\frac{\partial S_x}{\partial v} \\ N_z - \frac{\partial S_y}{\partial u}\frac{\partial S_x}{\partial v} + \frac{\partial S_x}{\partial u}\frac{\partial S_y}{\partial v} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\partial S_z}{\partial v} & \frac{\partial S_y}{\partial v} \\ \frac{\partial S_z}{\partial v} & 0 & -\frac{\partial S_x}{\partial v} \\ -\frac{\partial S_u}{\partial v} & \frac{\partial S_x}{\partial v} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta\frac{\partial S_x}{\partial u} \\ \Delta\frac{\partial S_y}{\partial u} \\ \Delta\frac{\partial S_z}{\partial u} \end{bmatrix} \tag{8}$$

Control Point Mesh Changes

Constraint-based manipulation is the method used to find the changes necessary to have the surface conform to the constraints generated by the tools. Each constraint is converted into a linear equation. The resulting set of linear equations is then solved simultaneously for the new control point positions to satisfy all the constraints. Since the set of equations may be under- or over-determined, a singular value decomposition solver is used.

For a constant u, the vector equation for a position of a point on a NURBS surface (Eq. (1)) is a linear equation that can be solved for the control point locations, P. This is valid because the weights, w, the basis functions, N, and the control points, P, are constant for a particular u.

The following example performs constraint-based manipulation on a NURBS curve for multiple constraints. A point, C, on a NURBS curve is defined as:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{9}$$

Let the denominator be defined as $D(u)$.

$$D(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i \tag{10}$$

Since the basis functions and the weights are known, $D(u)$ is a constant for any given u. Equation (9) can be rearranged to form:

$$C(u)D(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i P_i \tag{11}$$

Each constraint contains a change, ΔC, to a specific point on the curve. To avoid the work of completely reinterpolating the curve, only the changes to the control points, ΔP, are solved for.

$$C_{new}(u)D(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i P_{i,new} \tag{12}$$

C_{new} and P_{new} are defined as:

$$C_{new}(u) = C(u) + \Delta C(u) \tag{13}$$

$$P_{new}(u) = P(u) + \Delta P(u) \tag{14}$$

Combining Eqs. (11–14) leads to:

$$\Delta C(u)D(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i \Delta P_i \tag{15}$$

To include the effect of all the constraints for a given time step, the set of linear equations for all the constraints is solved simultaneously. This collection of linear equations can be written in matrix form, $b = Ax$, as:

$$\begin{bmatrix} \Delta C_1 D_2 \\ \Delta C_2 D_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} N_0(u_1)w_0 & N_1(u_1)w_1 & N_2(u_1)w_2 & N_3(u_1)w_3 & 0 \\ 0 & N_1(u_2)w_1 & N_2(u_2)w_2 & N_3(u_2)w_3 & N_4(u_2)w_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} \Delta P_0 \\ \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \\ \Delta P_4 \end{bmatrix} \tag{16}$$

Any control point that does not have any non-zero basis function for any of the constraint's u values cannot be moved to affect the edited area of the curve and can be dropped from the equation. This creates a smaller matrix to solve even for very large NURBS surface meshes.

Equation (16) can be extended for use with NURBS surfaces. When performing constraint-based manipulation on a NURBS surface, the 2D matrix of control points is mapped into a $(n+1)\cdot(m+1)$ column vector. The ordering of the control points is not important as long as similar ordering is used for the A matrix. A trivial example on a 2×2 surface with 3 constraints is presented here:

$$\begin{bmatrix} \Delta S_1 D_1 \\ \Delta S_2 D_2 \\ \Delta S_3 D_3 \end{bmatrix} = \begin{bmatrix} N_{00}(u_1)N_{00}(v_1)w_{00} & N_{10}(u_1)N_{10}(v_1)w_{10} & N_{01}(u_1)N_{01}(v_1)w_{01} & N_{11}(u_1)N_{11}(v_1)w_{11} \\ N_{00}(u_2)N_{00}(v_2)w_{00} & N_{10}(u_2)N_{10}(v_2)w_{10} & N_{01}(u_2)N_{01}(v_2)w_{01} & N_{11}(u_2)N_{11}(v_2)w_{11} \\ N_{00}(u_3)N_{00}(v_3)w_{00} & N_{10}(u_3)N_{10}(v_3)w_{10} & N_{01}(u_3)N_{01}(v_3)w_{01} & N_{11}(u_3)N_{11}(v_3)w_{11} \end{bmatrix} \cdot \begin{bmatrix} \Delta P_{00} \\ \Delta P_{10} \\ \Delta P_{01} \\ \Delta P_{11} \end{bmatrix} \tag{17}$$

The constraints created by the normal tool specify changes in a surface's derivative. Constraint-based surface manipulation can allow editing of surface derivatives by replacing Eq. (1) with Eq. (18) or Eq. (19) which are the equations for the partial derivatives of a surface. The equations for directly determining the partial derivative of a NURBS surface with respect to u or v can be rearranged so that only the basis functions in the numerator are replaced with their derivatives.

$$\frac{\partial S(u,v)}{\partial u} = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m}\frac{\partial N_{i,p}(u)}{\partial u}N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{18}$$

$$\frac{\partial S(u,v)}{\partial v} = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)\frac{\partial N_{j,q}(v)}{\partial v}w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{19}$$

The editing of either the positions or derivatives results in the creation of a matrix of linear equations. Depending on the number of constraints, the set of linear equations can be over-determined, under-determined, or sufficient. A solution method that can handle such a variety of systems is the pseudoinverse. The pseudoinverse is a method for finding the solution to a set of linear equations, $b=Ax$, even if the system is over-determined, under-determined, or singular. The pseudoinverse, A^+, is a matrix where $x_0=A^+b$. If the matrix is over-determined, the solution, x_0 will be a least-squares approximation. If the matrix is under-determined, the solution will be a smallest distance solution. The pseudoinverse is found using a singular value decomposition (SVD) solver. The code for the SVD solver used in this application is from *Numerical Recipes in C* [12].

Future Work

The mathematical method behind this surface manipulation program has much untapped potential. The block tool algorithm can be expanded to create new tools by changing its shape. The active face of the tool could be stored as a NURBS surface, letting the user shape the shaping face of the tool. The shape of the volume of the tool could be changed to create a stamping tool.

The normal tool is only an example of what can be done with shaping tools that edit a surface's derivatives. A tool that edited the second derivative of a surface would control the surface curvature. Tools could be created that affect the different smoothness ratings of a surface, such as Gauss or reflection lines. The block tools could be modified to force the surface to be tangent to the active face at a tool's edges. This would create a smoother transition between the edited area of the surface and the rest of the model. The normal tool currently affects only a point on the surface, with the area of effect determined by the degree and control point spacing. It would be possible to control a larger area by placing more normal constraints on the surface. A point or line could be defined as the point/line of rotation by also applying a constraint to keep the surface fixed to the point/line.

The addition of haptic force feedback would also greatly enhance this application by providing the designer with resistance based on the rate the surface is being deformed. The haptic feedback would reflect how the tools are intersecting and deforming the surface, information that is already generated by the application.

Conclusion

This paper has presented DN-Edit, a method for editing NURBS surfaces within a virtual environment. Virtual shaping tools have been created that are able to edit NURBS-based virtual prototypes. These tools enable control over multiple points on a surface for a higher level of control over the surface shape. A method has also been presented where a surface normal can be controlled through a direct interface.

DN-Edit provides designers with a tool to change the shape of NURBS volumes in a virtual environment using virtual shape tools. The ability to change the shape of a virtual prototype while interacting in the virtual environment will facilitate collaborative design reviews by allowing participants to view the results of their shape change recommendations immediately.

Acknowledgments

This research was conduct at the Virtual Reality Applications Center at Iowa State University with funding provided by the National Science Foundation grant #DMI-9625601.

References

[1] Durlach, N. I., and Mavor, A. S., 1995, *Virtual Reality, Scientific and Technological Challenges*, National Academy Press, Washington, D.C.
[2] Liang, J., and Green, M., 1994, "JDCAD: A Highly Interactive 3D Modeling System," Computers and Graphics, **18**(4), pp. 499–506.
[3] Dani, T. H., Chu, C. P., and Gadh, R., 1997, "COVIRDS: Shape Modeling in a Virtual Reality Environment," 1997 ASME Design Engineering Technical Conferences, Sacramento, California, September, 14–17, DETC97/CIE4302.
[4] Sederberg, T. W., 1986, "Free-Form Deformation of Solid Geometric Models," Computer Graphics, **20**(4), August, pp. 151–160.
[5] Gaylean, T. A., and Hughes, J. F., 1991, "Sculpting: An Interactive Volumetric Modeling Technique," Computer Graphics, **25**(4), pp. 267–274.
[6] Sachs, E., Roberts, A., and Stoops, D., 1991, "3-Draw: A Tool for Designing 3D Shapes," IEEE Comput. Graphics Appl., **11**(6), pp. 18–26.
[7] Furlong, T. J., 1997, "Virtual Reality Sculpture Using Free-Form Surface Deformation," 1997 ASME Design Engineering Technical Conferences, Sacramento, California, September 14–17, DETC97/DFM4511.
[8] Yeh, T.-P., and Vance, J. M., 1998, "Applying Virtual Reality Techniques to Sensitivity-Based Structural Shape Design," ASME J. Mech. Des., **120**(4), pp. 612–619.
[9] Piegl, L., and Tiller, W., 1997, *The NURBS Book*, 2nd Edition, Springer, Berlin.
[10] Fowler, B., and Batels, R., 1993, "Constraint-Based Curve Manipulation," IEEE Comput. Graphics Appl., **13**(5), pp. 43–49.
[11] Hsu, W. M., Hughes, J. F., and Kaufman, H., 1992, "Direct Manipulation of Free-Form Deformations," Computer Graphics, **26**(2), July, pp. 177–184.
[12] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992, *Numerical Recipes in C, The Art of Scientific Computing*, Cambridge University Press.