ELSEVIER

Computer-Aided Design 33 (2001) 403–420

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

A novel haptics-based interface and sculpting system for physics-based geometric design

F. Dachille IX*, H. Qin, A. Kaufman

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA

Abstract

Conventional geometric design techniques based on B-splines and NURBS often require tedious control-point manipulation and/or painstaking constraint specification via unnatural mouse-based computer interfaces. In this paper, we propose a novel and natural haptic interface and present a physics-based geometric modeling approach that facilitates interactive sculpting of spline-based virtual material. Using the PHANToM haptic device, modelers can feel the physically realistic presence of virtual spline objects and interactively deform virtual materials with force feedback throughout the design process. We develop various haptic sculpting tools to expedite the deformation of B-spline surfaces with haptic feedback and constraints. The most significant contribution of this paper is that point, normal, and curvature constraints can be specified interactively and modified naturally using forces. To achieve the real-time sculpting performance, we devise a novel dual representation for B-spline surfaces in both physical and mathematical space: the physics-based mass-spring model is mathematically constrained by the B-spline surface throughout the sculpting session. We envision that the integration of haptics with traditional computer-aided design makes it possible to realize all the potential offered by both haptic sculpting and physics-based modeling in CAD/CAM, virtual prototyping, human–computer interface, and medical training and simulation. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Geometric design; Physics-based modeling; Haptic sculpting; Dynamics; Interaction techniques

1. Introduction

During the past several decades, standard free-form splines such as B-splines and NURBS are frequently utilized to satisfy various design and manufacturing needs within CAD/CAM systems. They have been employed for the rapid modeling, efficient design and manufacturing of aircraft, ship hulls, automobile bodies, various industrial parts, consumer products, and natural objects. Moreover, free-form splines are of paramount significance to a much wider range of fields such as real-time interactive graphics, scientific visualization, medical imaging, and virtual environments.

Although the theoretical foundations and mathematical properties of free-form splines have been extensively researched, better and more efficient modeling techniques using these splines have evolved rather slowly. Traditional free-form spline modeling is often associated with the tedious and indirect manipulation via a large number of (often irregular) control vertices. In spite of the advent of modern 3D graphics interaction tools, these indirect geometric techniques remain inherently laborious. In contrast, physics-based modeling offers a superior approach to free-form geometric design such that it augments (rather than supercedes) matured geometric modeling techniques, offering users extra advantages. Within the physics-based frame-work, free-form spline models are equipped with mass distributions, internal deformation energies, and other material properties. The models, governed by physical laws, respond dynamically to applied forces in an intuitive and natural manner (see Refs. [19,20] for the details of the physics-based modeling methodology). Refer to Fig. 1 for an example of a variety of shapes each modeled in a matter of seconds by a non-artist.

Despite recent research advances in physics-based modeling, it is not yet possible to achieve the full modeling potential associated with the physics-based design frame-work. This is due to the fact that existing modeling systems often rely upon 2D mouse-based interfaces for 3D interaction. Direct *physical* operations on virtual objects with a mouse are not as natural and intuitive as interaction using a 3D interface. To ameliorate, we present a novel haptic

* Corresponding author. Tel.: +1-631-632-8470; fax: +1-631-632-8334.
E-mail addresses: dachille@cs.sunysb.edu (F. Dachille IX), qin@cs.sunysb.edu (H. Qin), ari@cs.sunysb.edu (A. Kaufman).

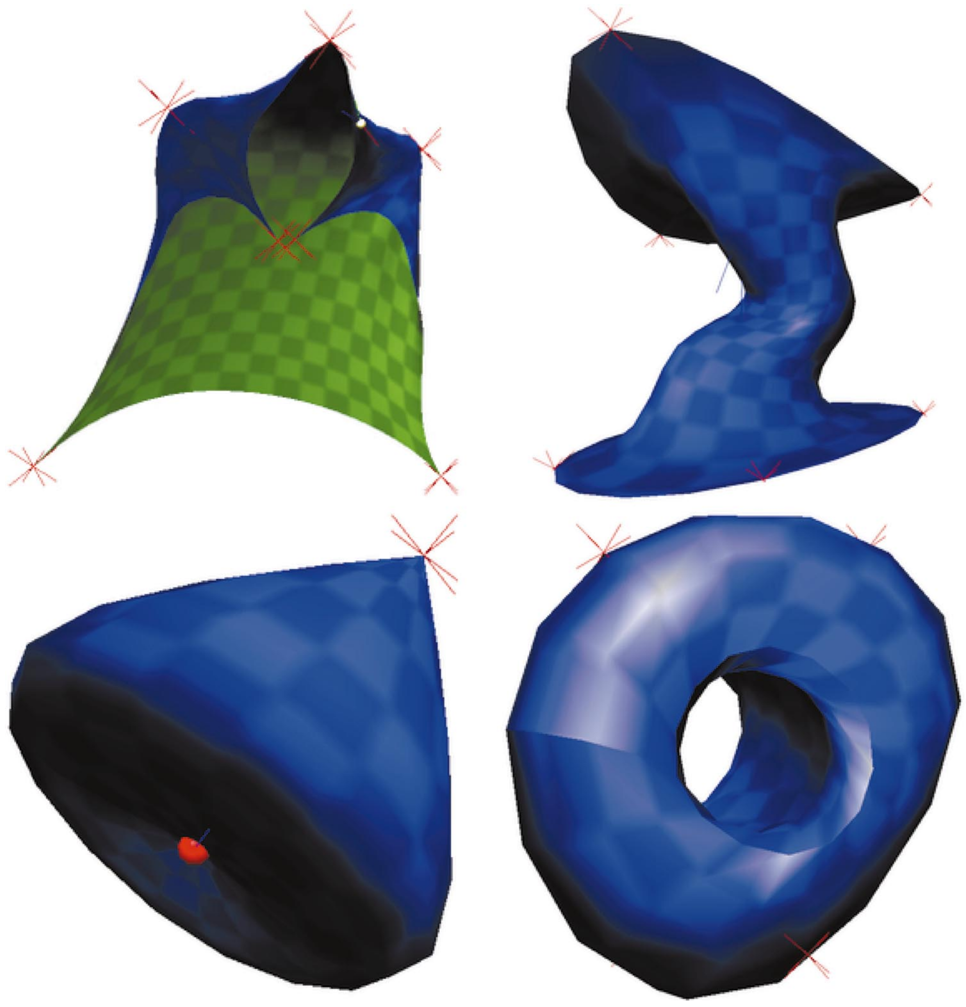

Fig. 1. A variety of shapes sculpted with the system each in a matter of seconds: (a) The surface formed into a “cape,”; (b) rolled into a cylinder and deformed; (c) the ends of the cylinder pinched to form a closed surface; and (d) the ends of the cylinder stitched together to form a torus.

approach for the intuitive and natural design of free-form spline models and integrate this easy-to-use interface with physics-based design algorithms (Fig. 2).

Haptics provides users a hand-based mechanism for intuitive, manual interactions with virtual environments towards realistic tactile exploration and manipulation. Haptics-based human–computer interaction has emerged as a critical metaphor in the fields of medicine, education, industry, entertainment, and computer arts. This is primarily because using force-feedback controls, designers, artists, as well as non-expert users can feel the model representation and modify the object directly, thus enhancing the understanding of object properties and the overall design. To date, practical devices for tactile interaction are commercially viable. A typical haptic device provides between two and six degrees of freedom (DOF) and its appearance ranges from a simple joystick to a complex robotic arm. We currently use the six DOF input and three DOF output PHANToM haptic device manufactured by SensAble Technologies. This device includes a pen-like stylus with a clickable button for interaction.

1.1. Motivation and contribution

Throughout a large variety of interactive graphics methods, few computer-based modeling techniques have come close to enabling modelers to design various shapes *directly* with their hands. Our goal is to allow users to reach toward an object, feel the physical presence of its shape, grab the object, manipulate it (with or without deformation), and release it in the desired location. Using a standard haptic device, our hand-based approach permits users to interactively sculpt virtual materials having realistic properties and feel the physically realistic presence with force feedback throughout the design process.

One potential advantage of this research is the integration of haptics with the computer-integrated design and manufacturing cycle. Using haptics in a virtual design environment, designers are able to feel and deform real objects in a natural 3D setting, rather than being restricted to mere 2D projections for input and output. Force feedback provides additional sensory cues to designers. This tactile exploration can afford designers to gain a richer understanding of the 3D

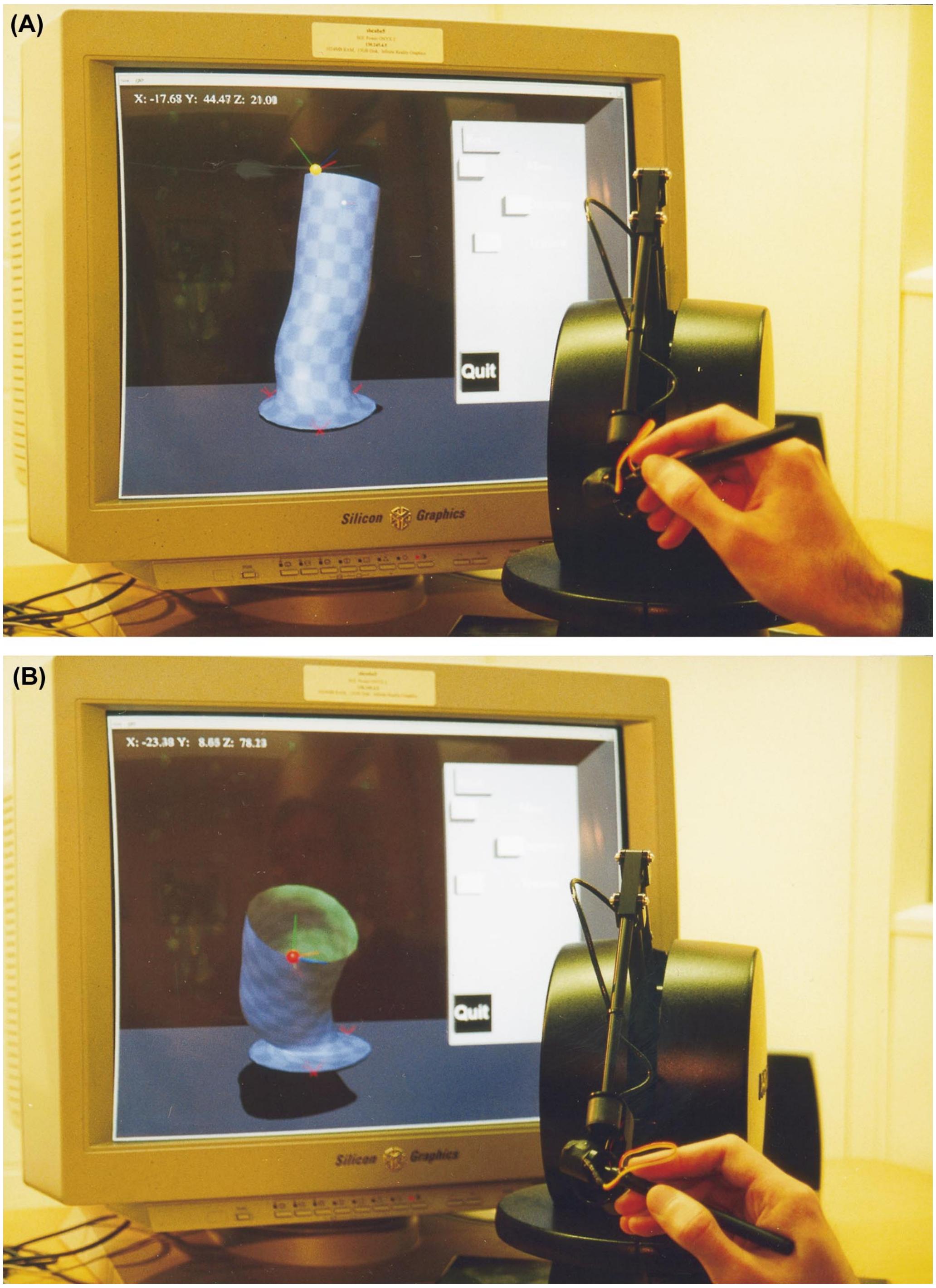

Fig. 2. A demonstration of the haptic sculpting system in action. A deformable cylinder anchored to the ground plane is pulled and twisted.

nature using the human hand for spatial and temporal interaction. The use of haptics in a virtual design environment promises to increase the bandwidth of information between designers and the synthetic modeling world. Furthermore, the use of haptics in design, analysis, and manufacturing processes can potentially shorten the product development cycle, enhancing the effectiveness of the design and analysis process for industry.

Prior research primarily focused on haptic rendering (i.e. the feeling of rigid surfaces/solids). In contrast, our haptic modeling system allows modelers to interactively deform a non-rigid free-form surface (e.g. a B-spline object) in real-time. The B-spline surface sculpted by our haptic device is a dynamic physics-based model, which inherits all the intrinsic behaviors of physical, real-world objects. The dynamic behaviors of our free-form surfaces result from a set of differential equations and produce intuitive shape variations. From an optimization point of view, our haptic sculpting dynamically optimizes an array of geometric and physical constraints enforced upon an arbitrary set of geometric degrees of freedom (i.e. control vertices). The B-spline surface currently available in our haptic design system is a specific case of a more general D-NURBS [20] object with fixed weights. Our on-going research endeavor is to further extend the haptic system to sculpt more powerful D-NURBS objects.

We develop several high-level haptic sculpting tools to expedite the intuitive modification of spline objects in a natural way. The tools developed in this system allow direct interactive modification of position, tangent, normal, and curvature constraints via forces. One key advantage for introducing these high-level tools into haptic design is that non-expert users are able to concentrate on visual shape variation without necessarily comprehending the underlying (rather complicated) mathematics of object representation. In particular, the B-spline control points and their associated basis functions become transparent to modelers in our haptic design environment, only the objects of interest remain visible from the users' point of view.

We develop a dual representation for physics-based geometric design. In physical space, a physics-based B-spline surface is discretized into a mass-spring model equipped with material and elastic properties to provide dynamic realism. The physical model provides an efficient, intuitive approach to specify curvature, normal, tangent, and other constraints. This mass-spring model is constrained in mathematical space by the B-spline surface throughout the sculpting session. Its behavior evolves in response to the Lagrangian equations of motion subject to various geometric constraints. The equations of motion are solved in real-time using a tractable numerical solver. Note that the polyhedral representation can be approximated to any user-specified error tolerance making it useful for simultaneous graphics rendering and haptic rendering.

It may be noted that the integration of a haptic interface and physics-based modeling should be of interest to broader communities. This novel approach also presents other value-added potential:

- The haptics-based system is more user-friendly, more intuitive, and easier-to-use from the viewpoint of both professional designers and non-expert users such as artists; it should appeal to the general public.
- The haptic interface should be more attractive to computer professionals from diverse application domains. For example, collaborative design, involving a group of artists, computer programmers, and engineering designers, can be readily accomplished.

The remainder of the paper is organized as follows. Section 2 reviews research concerning physics-based modeling. Section 3 presents the mathematical formulations. Section 4 describes the current state of haptics and addresses technical challenges relevant to our haptic sculpting system. Section 5 describes the detailed components of our sculpting system and implementation issues. Section 6 presents the components of the application and addresses the issue of system assessment. Section 7 concludes the paper and outlines future research directions.

2. Physics-based modeling

Various techniques have been developed to generate fair surfaces that satisfy multiple constraints and optimize an energy-based objective functional [3,15,33]. It is also possible to construct dynamic surfaces with natural behavior governed by physical laws [17,30]. The benefit of physics-based behavior during interactive design is that the development of the surface follows intuitive physical paths and the surfaces react to external manipulation in a predictable way. For example, pulling on a ribbon will tend to bend rather than twist and stretch it. Thus, the incorporation of physics with sculpting aids the design task, especially with the reality-based feedback obtained from the haptic device.

Terzopoulos and Fleischer [29] provided a basis for physics-based design by combining the two techniques for simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [3] developed a prototype system for interactive design based on the finite-element optimization of energy functionals. Bloor and Wilson [2] used similar energies optimized through numerical methods for B-splines. Moreton and Séquin [15] interpolated a minimum energy curve network with quintic Bezier patches by minimizing the variation of curvature. Celniker and Welch [4] investigated deformable B-splines with linear constraints. Welch and Witkin [33] proposed a variational surface modeling method which automatically generates a surface model that attempts to satisfy a set of desired properties. Our method similarly develops a surface in response to the users' desired properties. Thingvold and Cohen [31] proposed to use elasto-plastic mass-spring-hinge models on the B-spline control points. Our approach assigns the masses directly on the surface and hides the control points from the user. Stewart and Beier [24] demonstrated a direct curve manipulation technique which allows the direct control of position, normal, and curvature. Our approach facilitates direct manipulation of point, normal, and curvature constraints using physically based techniques and direct feedback via haptics. Halstead et al. [7] implemented smooth interpolation with Catmull–Clark surfaces using a thin-plate energy functional. We can use a large variety of functionals in our system (e.g. stretching and bending) to limit the degrees of freedom and impart realism. For example, it could be helpful to define a functional which resists self-penetration by simulating an electrostatically charged surface that avoids itself.

Other relevant work on physical models include the realistic animation of volumetric objects and molecular structure [17,25,26,30], and the direct manipulation of spline models [5,8]. Our method dynamically sculpts with haptics spline objects controlled by physical laws subject to various constraints. The haptic device directly operates on position, tangent, normal, and curvature with forces. Recently, Grimm and Ayers [6] proposed a framework for curve editing. Multiple representations of a curve are maintained in order to facilitate curve modification in the most appropriate domain. In particular, a particle representation allows the user to push the particles around using a *spatula* tool for the purpose of local and/or global curve deformation. In our haptic system, we formulate our spline models with two synchronized representations that permit surfaces to conform to B-splines in mathematical domain, while exhibiting physical behavior and satisfying material properties subject to intrinsic geometric constraints (e.g. curvature, etc.).

3. Dynamics formulation

We represent a continuous B-spline surface $s(u,v)$ as the combination of a set of basis functions $B_{i,k}$ and $B_{j,l}$ with $(n+1)\times(m+1)$ control points $\mathbf{p}(t)$, where t denotes time

$$s(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{p}_{i,j}B_{i,k}(u)B_{j,l}(v) \tag{1}$$

Note that, $B_{i,k}$ and $B_{j,l}$ are piecewise polynomials of order k and l, respectively. Both u and v are parametric variables. Their parametric domain is determined by two sets of non-decreasing knot sequences, respectively. In the interest of the space here, we refer readers to Ref. [20] for the B-spline details. Without loss of generality, we assume that u and v belong to [0,1]. The control point vector, $\mathbf{p}$, is the concatenation of all 3D control points $\mathbf{p}_i=[x,y,z]^{\mathrm{T}}$

$$\mathbf{p}=[\mathbf{p}_{0,0}^{\mathrm{T}},\mathbf{p}_{0,1}^{\mathrm{T}},\ldots,\mathbf{p}_{n,m}^{\mathrm{T}}]^{\mathrm{T}},$$

where T denotes matrix transposition. We now discretize the dynamic surface into a set of parametrically uniform $g\times h$ points d, which form $(g-1)\times(h-1)$ quadrilateral finite elements. Our dynamic surface has dual representations in the mathematical domain ($\mathbf{p}$, A) and the physical space (d). The two formulations are connected by

$$\mathbf{d}=\mathbf{A}\mathbf{p} \tag{2}$$

where A is a transformation matrix whose entries are basis functions evaluated at various parametric values. For B-spline surfaces, A is uniquely determined by the parametric values of $\mathbf{d}$, thus it is constant when all parametric values of the discretization are constant. In this case, the matrix and this pseudo-inverse can be pre-computed.

The discretized dynamic model has material quantities such as mass $\mu(u,v)$, damping $\gamma(u,v)$, and stiffness $\rho(u,v)$ defined as functions over the surface. Often, these functions are constant over the surface, yielding uniform, homogenous surface characteristics. However, as demonstrated later in Section 6.1, our system allows the user to *paint* these properties interactively and directly onto the model in real-time. The discretized surface is modeled as point masses connected by a network of springs across nearest neighbors and along both diagonals forming a polyhedral representation useful for haptic interaction (see Fig. 3). This representation lumps all of the surface mass at discrete locations ("mass points") on the surface with zero-weight spring connections. This representation is computationally simple and can be simulated using only basic physical equations. Alternatively, the dynamic surface can be approximated using finite element methods based on **d**. The finite elements derived from **d** are mathematically equivalent to our current implementation. We use a mass-spring model instead because of its simplicity and the critical needs of real-time haptic sculpting. Note that, although one diagonal spring is mathematically necessary to prevent skew in the rectangular mesh, it makes the stiffness anisotropic.

We formulate the motion equation of all mass-points using a discrete simulation of Lagrangian dynamics

$$\mathbf{M}\ddot{\mathbf{d}}+\mathbf{D}\dot{\mathbf{d}}+\mathbf{K}\mathbf{d}=f_{\mathrm{d}}. \tag{3}$$

The force at every mass-point in the mesh is the sum of all possible external forces: $f_{\mathrm{d}}=\Sigma f_{\mathrm{ext}}$. The internal forces are generated by the connecting 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 non-linear phenomena are desired.

Because all discretized points and springs are constrained by the dynamic B-spline surface (Eq. (2)), we shall formulate the motion equation of physical behavior for all the control points

$$\mathbf{A}^{\mathrm{T}}\mathbf{M}\mathbf{A}\ddot{\mathbf{p}}+\mathbf{A}^{\mathrm{T}}\mathbf{D}\mathbf{A}\dot{\mathbf{p}}+\mathbf{A}^{\mathrm{T}}\mathbf{K}\mathbf{A}\mathbf{p}=\mathbf{A}^{\mathrm{T}}f_{\mathrm{d}}. \tag{4}$$

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

$$A^{\mathrm{T}}MA\ddot{\mathbf{p}}+A^{\mathrm{T}}D\dot{d}+A^{\mathrm{T}}Kd=A^{\mathrm{T}}f_{\mathrm{d}} \tag{5}$$

$$A^{\mathrm{T}}MA\ddot{\mathbf{p}}=A^{\mathrm{T}}f_{\mathrm{d}}-A^{T}D\dot{d}-A^{\mathrm{T}}Kd \tag{6}$$

$$\ddot{\mathbf{p}}=(A^{\mathrm{T}}MA)^{-1}(A^{\mathrm{T}}f_{d}-A^{\mathrm{T}}D\dot{d}-A^{\mathrm{T}}Kd). \tag{7}$$

Note that, if a finite element model is used in our system instead, then non-linear effects would be rather difficult to model. However, with our finite difference approach, non-linear damping and stiffness effects can be enforced in a straightforward fashion. Linear damping is implemented by reducing the velocity of each mass-point by a certain user-defined proportion. Non-linear damping reduces the velocity of each model point by a certain proportion that can be characterized by a function of time and spring-force magnitude. Similarly, non-linear stiffness is achieved by applying forces proportional to a function of the spring-magnitude. These techniques need to evaluate only

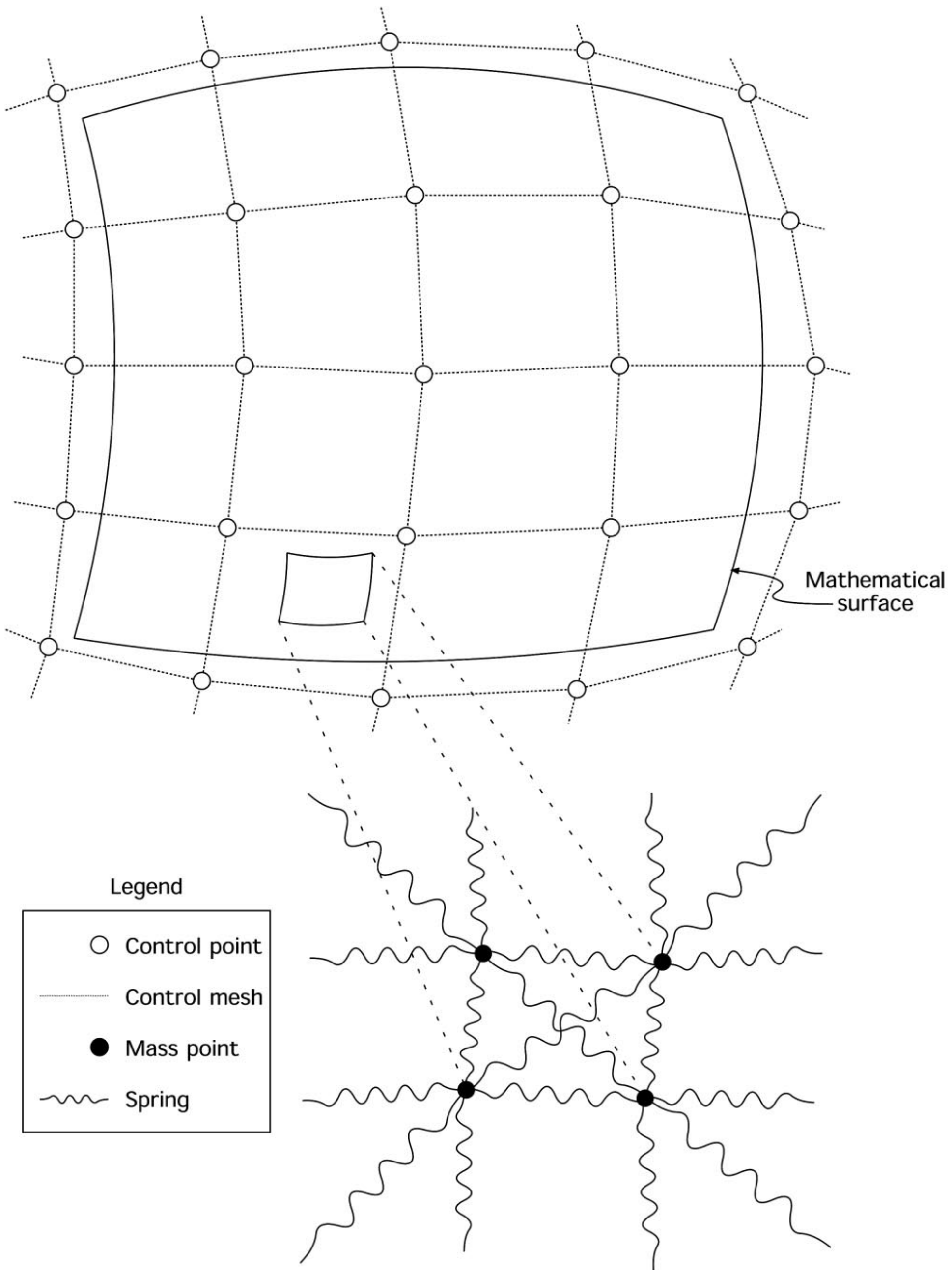

Fig. 3. A depiction of the control mesh and its discretized mass-spring polyhedral representation.

relatively inexpensive functions in the inner loops of the stiffness and damping equations.

Error control is an integral part of the algorithm. Errors are possibly introduced into the design system by either coarse time steps or a low-resolution discretization. The time step is normally very small since the haptic device requires a high update rate. If the model is very complex, the time step may become large and errors creep into the simulation unless an adaptive time step is used. However, these errors do not necessarily deteriorate the surface design task since the system is continually evolving towards an equilibrium of energy minimization. Temporary inconsistencies in dynamics appear not to have a negative effect in our system toward the final stable shape. Coarse discretization also leads to potential errors, so an accurate bound is necessary for discretization. Fortunately, the discretization density is user-specified, thus the error can be controlled by the users.

The surface dynamically optimizes its energy functional, constantly seeking a lower total energy state. Because the energy functional is physically based, the dynamic optimization imparts realism during the modeling session. The modeler has the ability to specify constraints in several domains simultaneously (Fig. 4). Constraints are supplied in physical space, tangent space, and curvature space, corresponding to location, first derivative, and second derivative, similar to Ref. [24].

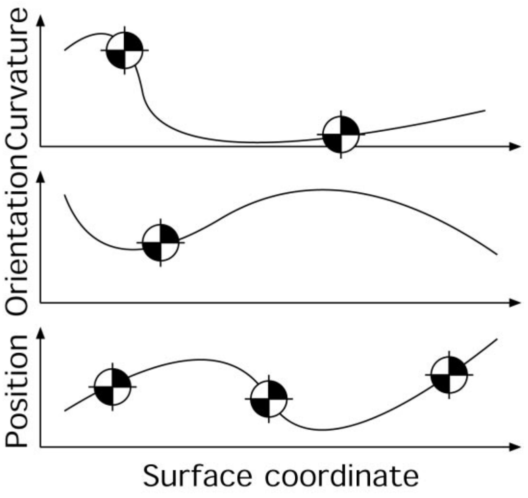

Fig. 4. Surface constraints are specified in multiple domains, and the dynamic model evolves to optimize the shape subject to those constraints.

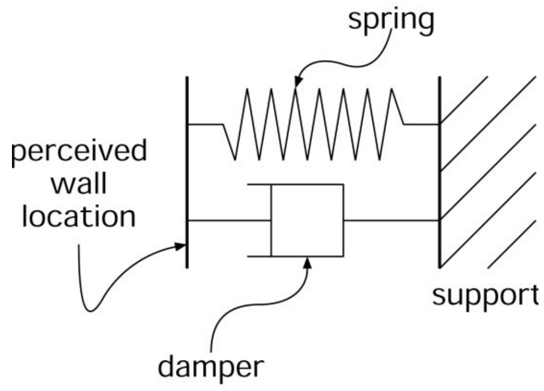

Fig. 5. Spring-damper simulation of a rigid wall.

4. Haptics techniques

Haptics and its associated techniques have been well researched in recent years. A good review of haptics literature can be found in Ref. [23], where the terminology and fundamentals of haptics simulation, including cognitive studies and mechanical requirements, are also detailed. Minsky et al. [14] investigated the conditions required to sustain the illusion of reality in a haptic system. Thompson et al. [32] investigated haptic rendering of NURBS surfaces. The difficulty in rendering NURBS surfaces is the complexity of the intersection task. Special techniques were invented in order to improve the computation. Haptic rendering requires: (1) sensing the position of the user's finger; (2) locating the nearest point to the surface; and (3) appropriately generating a force to be applied to the finger. One prime difficulty of realistic haptic modeling and rendering is speed. A rather high force update rate, on the order of 1000 Hz, is necessary to achieve convincing haptic feedback. This is because, although the nervous system of human being can only perform tactile tasks at the order of 10–20 Hz, humans can sense small movements up to 1000 Hz. Hence, any attempts to emulate physical movement with update rate less than 1 kHz will result in tactile vibrations, leading to the uncomfortable perception of friction, ridges, and general roughness depending on the frequency range. Note that, this is far greater than the necessary threshold required for real-time visual display (up to 60 Hz).

Because haptics entails a great amount of time-critical processing, a high haptic update rate is often achieved either using a dedicated processor or using a multiprocessor machine. Jacobs et al. [9] discussed the critical issues of synchronizing sensors in virtual reality environments. Inspired by Ref. [9], we also exploit the similar idea to synchronize multiple simulation loops, (i.e. graphics, haptics, dynamics simulation). It may be noted that providing a high force update rate is currently feasible for simple objects with simple linear even non-linear characteristics. For example, a solid wall and its physical behavior in a virtual environments can be approximated using a high-stiffness spring and a high-viscosity damper (see Fig. 5). The spring is only activated when the user penetrates the plane of the wall; and it then repels the user to the edge of the wall. The strength of the spring and the viscosity of the damper can be pre-determined to match the mass and velocity of a typical user's hand. It has been observed in Ref. [23] that a wall should resist a force of about 5–15 N in order to be perceived as a solid obstacle. The spring reaction force is computed using Hooke's law $f = kx$, where k is the spring stiffness. The direction of force application can be one of the following: (1) opposite to the velocity; (2) normal to the surface; (3) or any direction in between. Note that, if the direction always is kept normal to the surface, the wall will be perceived like frictionless ice. Varying quantities of surface friction can be simulated by applying extra force opposite to the velocity along the tangential direction. Additional non-linear effects, such as static and dynamic friction are also possible, depending on the magnitude of the tangential velocity opposing force. Note that all of these techniques can be easily implemented through simple vector calculations, which can be reasonably completed in less than the allotted 1 ms.

In contrast, effectively computing the reaction force for complicated geometry is a much more difficult problem. In general, finding the intersection between the user's haptic interface point (HIP) and the surrounding environment characterized by the complex geometry is not trivial. The haptic interface point is generally a simple sphere, but it may actually not fit into thin crevices or it may simply pass through thin objects during the sculpting session. Computing the intersection of a sphere with many small triangles of a complex polyhedron can be highly time-consuming, in spite of the fact that the complexity analysis shows that the problem itself is virtually O(n). For the same reason, coherence and spatial sorting must be exploited to maximize performance. In addition, ambiguities can also arise, such as in the case of penetrating the corner of an object. It is uncertain which directions the normal force should point to: whether along the direction of the entrance or along the direction of the nearest surface point. To ameliorate this, force shading [16] has been devised, which blends together neighboring normals to avoid discontinuities between neighboring polygons. Please note that sometimes an opposite effect is more desirable, e.g. when the user is attempting to select an edge or a vertex. Instead of falling off the polygon when the edge is reached, an infinitesimally small tactile ridge can be added to the edge of all surfaces so that as long as the user contacts the surface, he/she can not fall off the edge of the surface [13]. The best treatment of edges is highly application dependent.

There have been several approaches to solve the intersection problem and make it feasible for generic environments. Massie and Salisbury [11] divided the virtual environments into many sub-regions within which the reaction force could be readily calculated, unfortunately, this technique has been proved to be rather problematic in general because of potentially overlapping regions. The force computation was based on the penalty method similar to the wall configuration described above. Later, Zilles and Salisbury [34] proposed to use a *god object* which would be constrained to the surface of the intersecting object. The haptic simulation would then continually attempt to minimize the distance between the god object and the haptic cursor. More recently, Ruspini et al. [21] further improved the *god object*, naming it a *virtual proxy* and making it sufficiently large in order to prevent it from penetrating any objects in the scene. In Ref. [21], constraint-based path planning was exploited to move the proxy toward the physical position, hence improving both force shading and surface texture rendering. A set of bounding spheres was used to perform efficient computation of intersection with large polygonal datasets. In our system, we do not utilize continuous sliding contact with the surface, thus sophisticated intersection routines are not required.

Alternatively, Randolph et al. [10] developed an approach for haptic rendering of a complex surface by using an intermediate representation. A locally planar approximation to the surface was computed at the collision point as frequently as possible, but not as fast as the haptic device requires to sustain illusion (about 1 kHz). The intermediate representation was used to generate the reaction forces for the high-speed haptic loop. A further improved method has been developed by Salisbury and Tarr [22]. Recently, Swarup [27] and Tarr [28] explored physically based haptic interaction with compliant objects. Visco-elastic virtual objects are modeled as a discrete network of masses connected by springs and dampers. Our work expands the notion of just touching compliant objects to directly manipulating fully deformable objects for the purpose of sculpting.

5. Sculpting system and implementation

The sculpting system allows the user to reach out toward the surface and click the stylus button to grab hold of the surface. In response, the surface reacts in a physically plausible manner and deforms according to the manipulation of the user. At any time, the user can *lock-in* changes and set a persistent constraint.

Our haptic system executes in a tight loop for direct manipulation. It constantly evolves the dynamic surface (governed by physical equations) in response to the user's sculpting forces and other inputs. At each time step, the system samples the user's commands to obtain the current six DOF position of the haptic device. The commands (keyboard hotkeys, mouse clicks, haptic stylus clicks, and haptic stylus movement) impart sculpting forces to the physical model, and the internal forces are calculated. The forces are applied, and the system estimates the position of the dynamic model at the next time step. Note that, throughout the entire sculpting process, the discrete finite elements are constrained to form a B-spline surface, whose control vertices are functions of time. The surface discretization is then updated using the new deformed surface configuration, and the shape is sent to the display device. To reduce the latency and maximize the throughput, we resort to multi-threading the haptics, graphics, and dynamics processes with weak synchronization. This technique leads to the parallel processing of haptic sculpting.

We now detail the major components of the haptic interaction, including grabbing the nearest point on the surface, interacting with arbitrary points on the surface, maintaining realism, implementing sculpting tools, computing force feedback, enforcing constraints, and numerical integration. The fundamental sculpting tool is the rope tool, which allows positioning and orienting the surface at a single point via a virtual rope. A magnet tool permits gross deformations of the surface. Other sculpting tools allow specifying persistent constraints (position, normal, and curvature) and interactively painting material properties.

5.1. Manipulating the surface via the rope tool

To achieve the ultimate goal of real-time haptic sculpting, first of all, we shall address the fundamental question of how to grab the surface location with the virtual haptic cursor and subsequently move it with force feedback to any desired six DOF position. The intrinsic physical properties of the dynamic surface will govern the motion and behavior of the deformable sheet in response to the user's input force.

Our system permits users to freely grab any position on the surface and move it accordingly. The performance of this function is enhanced by the accompanied polyhedral representation. The polyhedral representation makes it much easier to search for the nearest point on the surface, unlike the complicated NURBS surface intersection task [32]. The nearest point on the surface does not need to be updated too frequently, so the system can compute the distance from the cursor to all the vertices of the polyhedral representation during the graphics loop which operates at 60 Hz. When the user invokes the grab instruction by pressing the button on the haptic stylus, he/she locks onto the nearest point, obviating the need to re-compute the nearest point on the surface. Furthermore, when the surface is not being manipulated, the cursor is allowed to freely intersect the surface without force feedback, aiding in the selection of interior parts of the surface. It is also possible (although not implemented) to constantly intersect the surface and feel the existence of continuous surface geometry and discrete vertices using the techniques in Ref. [32].

The user's cursor is attached to the surface by an incompressible spring — a *rope*. The rope tool provides a way to "lasso" the surface and manipulate it by pulling on the rope. When the user first clicks the stylus button using the rope tool, a spring (a virtual rope) is generated between the tool and the surface. The user can not push on the surface via the rope, but if the length exceeds a certain threshold, a high spring force is generated so that it feels like a stiff rope. The rest length of the spring is initially set to be the original distance. The length of the spring decreases over time in such a manner that within about 1 s, the length becomes zero. This will attract the surface towards the tool (and of course, draw the human hand toward the surface geometry) so that the user will quickly feel that he/she is in fact holding the surface directly. Note that the tool is tethered to the surface by a high-stiffness spring, but the contact point is a massless proxy constrained to a fixed location on the surface for as long as the user grasps the stylus button. Gradually reeling in the surface toward the cursor prevents high derivative jerking forces which can potentially injure the user of the haptic device or damage the device.

Attaching the rope tool initially causes no change to the surface. Only after engaging the rope tool for a fraction of a second does any deformation result. Therefore, small deformations are possible using only short clicks of the stylus button. The extent of deformation can be controlled by changing the overall mass of the surface, allowing the surface to react either wildly or mildly to sculpting forces. An area for future work would be to associate a "temperature" with the surface parameterization which adjusts the malleability via the mass; a "hot" region would deform easily, while a "cold" region would be very stiff. In addition to the temperature always progressing toward ambient, virtual heat and cool could be brushed over the surface to vary the temperature.

5.2. Interaction with an arbitrary location

While it is straightforward to grab vertices of the polyhedral representation, the grabbing operation is easily extended to interact with any point (not necessarily a mass vertex) on the surface. Note that the point interaction between the exact location on the surface and the cursor

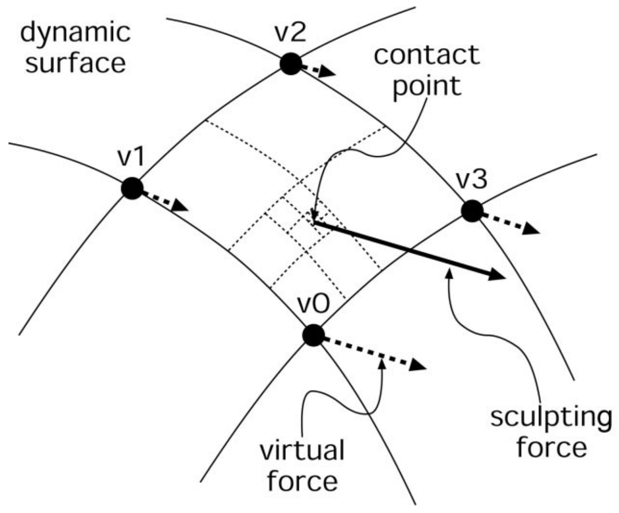

Fig. 6. Grabbing the dynamic surface at a non-vertex location distributes the virtual force to the nearest vertices (v0–v3).

in general is not solvable in a straightforward manner due to the non-linear nature of this computation. Instead, a simple iterative method is employed to compute an approximate location. Our algorithm works as follows. Once the nearest vertex is found, the four (or fewer) adjacent faces are tested for proximity to the cursor. The nearest face is then iteratively subdivided, and the nearest quadrant to the cursor is successively selected for subsequent operation of subdivision. We observe that about five levels of subdivision are sufficient to locate an approximate surface location within the satisfactory accuracy in the parametric domain.

When the user grabs an arbitrary point on the face defined by the four vertices as shown in Fig. 6, the sculpting force generated by the interaction must be transformed into an equivalent force field acting on all the adjacent control points **p**. Note that this would require to compute four basis functions corresponding to each of the control points, respectively. On the other hand, the virtual force could be transmitted to the four surrounding vertices using the idea that the polygonal quadrilateral element is temporarily a stiff non-deformed plate so that pushing in the vicinity of central regions transmits a force to the four vertices. The force for each of the vertices is simply computed as a weighted bilinear interpolation of the distance from each point to the contact point. Note that the latter approach performs the weighted interpolation without the use of the parametric domain, hence it is more efficient. The above iterative search procedure only needs to be invoked at the time of user selection, i.e. when the user grabs the surface by clicking the stylus button, therefore, it is only required to be as fast as 1/20th of a second, the limit of human ability for manual dexterity.

5.3. Sculpting tools

If the user wishes to not only place, but also to orient the

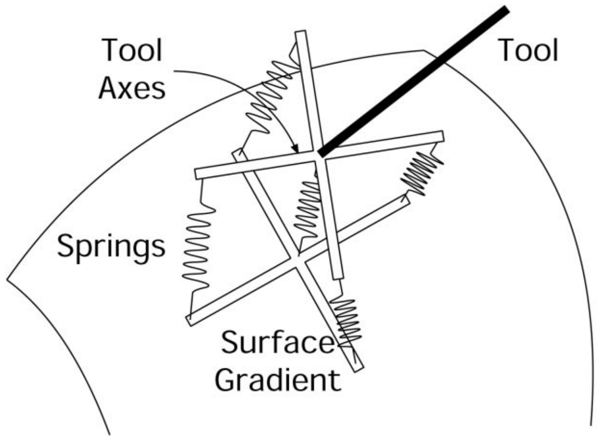

Fig. 7. The set of five springs is used to locate and orient the surface relative to the cursor position.

surface, then two additional springs are created. The goal is to align the u and v direction vectors with the user's cursor orientation. An orthogonal set of axes are generated at the cursor position, which is located at the end of the pen-like haptic device (Fig. 7). The axes are always fixed relative to the pen and the user's hand and the length of the vectors is set to match the rest length of the springs. Five springs are used to orient the surface relative to the cursor position.

The rope tool always grabs the nearest mass-point on the surface. An alternative magnet tool allows the user to manipulate a nearby subset of the mass-points simultaneously to allow gross deformations. The user first selects the magnet tool instead of the rope tool. Then, when the stylus button is depressed, all the mass-points within an effective radius are attracted to (or repelled from) the cursor position based on their distance. The force is distributed among nearby points using a user-defined function $\theta \in \mathbf{R}^3$, which can be constant across the region, inversely proportional to distance squared (like a real magnet), or any other distribution. Keyboard hotkeys allow selecting the various options such as rope or magnet tool, polarity of the magnet (either attractive or repulsive), and the magnet distribution function.

5.4. Force feedback

By adding external input forces based on the user's actions, the mesh deforms according to the physical properties of the model. Furthermore, Newton's third law must be held throughout the haptic sculpting: any force that the user applies to the model must be reflected back to the user via the haptic feedback mechanism (see Fig. 8). When the rope tool pulls on a mass-point by adding a force along a certain direction, then an equal and opposite force is generated at the other end of the rope (the cursor) along the direction of the rope. (Note that a rope can only transmit forces axially, not tangentially.) The force is calculated at 1000 Hz and transmitted to the haptic device where the force values are converted into motor torques leading to real forces at the cursor position.

The particular haptic device that we use only senses the orientation; therefore, it is not possible to transmit feedback torque to the user. If torque motors were available, the quality of the simulation would be increased by the improved haptic display. In that case, the net torque will be calculated by simply reflecting the torque applied to the surface. For more advanced tools such as the magnet, force is always transmitted to relevant mass-points which reflect forces back to the user along the opposite direction. Simple vector calculation (i.e. addition and negation) leads to the exact reaction force in all cases.

5.5. Constraints

Many methods have been used to implement constraints. Hsu et al. [8] 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 [33] utilized Lagrange multipliers to enforce a least-squared solution to a constraint matrix. Moreton and Séquin [15] used a minimum-energy network to optimize a system of linear and non-linear constraints. Terzopoulos [30] used the penalty method to drive a dynamic deformation for animation. Qin and Terzopoulos [20] used linear constraint techniques to deform physical models for design purposes. Platt and Barr [17] 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 requirement of

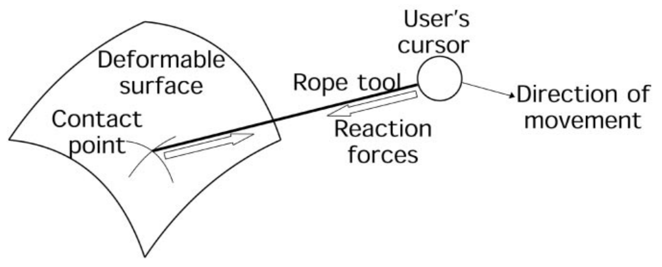

Fig. 8. To compute the haptic reaction force, Newton's third law is applied. By pulling on the virtual *rope*, tension is generated in the rope pulling in opposite directions for the surface and the user.

Table 1
Run times of physical simulations using both an explicit and an implicit solver versus surface complexity

Control point resolution	Discretized mesh resolution	Explicit solver run times (ms)	Implicit solver run times (ms)
4×4	10×10	0.5	10
4×4	20×20	2	11
4×4	40×40	10	19
8×8	20×20	5	100
8×8	40×40	25	120
12×12	25×25	42	780

smaller time steps. Reaction constraints improve the penalty method by fulfilling constraints exactly in the presence of outside forces. Lagrange constraints and augmented Lagrange constraints require complex differential equations that are not suitable for the real-time requirement of our haptic interaction.

Our system currently uses an explicit integration method for updating the physical system. Although in principle implicit solvers are capable of offering more stable and robust solutions to complex physical systems, they typically require a much higher computational effort, which may make it impossible to achieve the vital objective of real-time haptic interaction. Our experiments demonstrate that even a small number (3–5) of conjugate-gradient iterations requires roughly 20 times as much computation as the more cost-effective explicit approach (please refer to Table 1 for the detailed comparisons). Note that, stability and robustness are preserved in our prototype system with the explicit integration method. One technique we apply to our sculpting examples is an adaptive time step, i.e. if the acceleration exceeds a user-defined threshold, the time step is recursively halved.

Point constraints are implemented as high-stiffness springs between a mass-point and the specified location. We adopt the penalty method because our system is non-linear, and because it is physically plausible. Other methods fail to solve non-linear systems which are often over-constrained. At equilibrium, our method generates a least-squared energy fit to the specified constraints. This is because an ideal Hookean spring with energy $E = kl^2$ is minimized by the kinematic simulation, and the energy exactly corresponds to the squared error in the surface fit.

Tangent plane, or first derivative, constraints are implemented in the physical model by adding four springs to the system (similar to the four outer spring shown in Fig. 7). We again use the penalty method since this constraint also is non-linear. When a tangent plane constraint is enabled, the current discrete $(\Delta s/\Delta u)$ and $(\Delta s/\Delta v)$ are stored. At each subsequent timestep, the derivatives are added to the activation point to establish the desired position of the four neighboring points. Stiff springs are generated between the actual and the desired points, forcing the neighboring points into alignment with the tangent plane. Note that, this method fixes the rotation of the surface normal.

To avoid fixing the rotation, the tangent plane vectors can be established in an alternate way. The desired normal vector $\mathbf{n}_d$ is crossed with the actual u direction vector $\mathbf{u}_a$ to obtain the desired v direction vector $\mathbf{v}_d$ (see Fig. 9). Then the actual v direction vector $\mathbf{v}_a$ is crossed with the desired normal vector $\mathbf{n}_d$ to obtain the desired u direction vector $\mathbf{u}_d$. The two desired vectors are mirrored to compute four desired point locations. The vectors are then normalized to be the same length as the actual direction vectors so as not to introduce length distortions. Just as in the previous steps, stiff springs are generated between the four actual and the four desired points to force the neighboring points into alignment with the tangent plane. The vector sum of these four forces would result in an undesired translation of the mesh. Therefore, the vector sum is negated and added to the center point. This results in a force equilibrium and zero sum translation.

Curvature, or second derivative, constraints are implemented in the physical model by adding two springs to the system (see Fig. 10 for a 1D cross sectional example). Once again, we use the penalty method for unconstrained optimization since this constraint is non-linear. One spring spans the two neighboring points in the u direction and the other in the v direction. The length of each spring is determined as $2L \sin(\theta/2)$, where L is the average edge length and θ is the desired angle of curvature. The desired curvature is variable, and it can be set independently for u and v by selecting with the mouse a point on a 2D map of possible curvatures. The actual curvature could be either positive or negative; this method does not force the sign of the

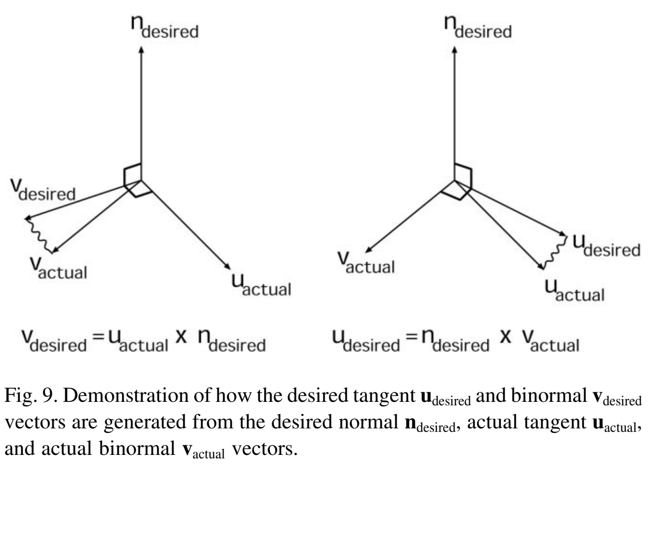

Fig. 9. Demonstration of how the desired tangent $\mathbf{u}_{desired}$ and binormal $\mathbf{v}_{desired}$ vectors are generated from the desired normal $\mathbf{n}_{desired}$, actual tangent $\mathbf{u}_{actual}$, and actual binormal $\mathbf{v}_{actual}$ vectors.

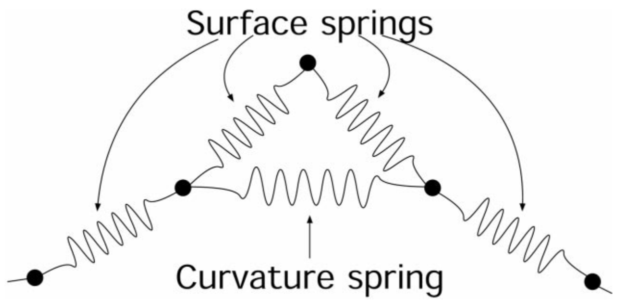

Fig. 10. A 1D example in which a *curvature spring* is added to the surface to cause bending to a desired curvature.

curvature, only its magnitude. It is up to the user to pull the surface in the proper direction. If more than two springs were used, the system would become over-constrained and increase the computational effort. Furthermore, this method does not force an absolute constraint, but in the absence of completing constraints and sculpting forces, the surface achieves the desired curvature.

Additional geometric constraints can be incorporated easily. For instance, control points can be constrained to be linear combinations of free control points using a special constraint matrix **B** to convert the control point vector **p** into a vector of free control points $\mathbf{p}'$ by

$$\mathbf{p} = \mathbf{B}\mathbf{p}'.$$

The constraint matrix **B** can be set up once at the beginning, incorporated into the basis function matrix **A** of Eq. (2), and the rest of the equations operate as before. **B** is initially the identity matrix, but as the number of free control points is reduced, each corresponding row of **B** is replaced by a linear combination of the remaining free control points $\mathbf{p}'$. As an example, we wrapped the edges of the control point lattice around on itself forming a cylinder (see Fig. 1b). By overlapping three rows of control points we achieved C^2 continuity across the seam. Note that in our system users specify high-level topological types without worrying about the four-level control point alignment developed in **B**. From the cylinder we formed a topologically closed surface by pinching the ends (see Fig. 1c). By further overlapping three rows of control points in the other parametric direction, we can develop a torus topology (see Fig. 1d).

5.6. Numerical integration

To sculpt spline surfaces in a haptic modeling system, it is vital to provide users with real-time feedback. Rather than using costly time integration methods that take the largest possible time steps, it is more important to provide a smoothly animated display by maintaining the continuity of the dynamics from one step to the next. Hence, less costly yet stable time integration methods that take modest time steps are desirable. In particular, we have developed simulation algorithms using both explicit and implicit integration methods in our system.

At each time-step of the explicit integration, the control point vector of the sculpt object at time t_i is computed based on the previous values at t_{i-1} and t_i. The summarized forces on the discretized mesh are applied, and the transformation matrix is used to determine the virtual force on the **B**-spline control vertices

$$f_\mathrm{p} = A^\mathrm{T} f_\mathrm{d}$$

The velocity ($\Delta\mathbf{p}/\Delta t$) of the control points is updated according to the applied forces and to the material quantities such as mass, damping, and stiffness. The control points are moved to a new position

$$\dot{\mathbf{p}}_{i+1} = \dot{\mathbf{p}}_i + \ddot{\mathbf{p}}_i \Delta t \tag{8}$$

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \dot{\mathbf{p}}_i \Delta t \tag{9}$$

The updated control points $\mathbf{p}_{i+1}$ are further used to update the discretized model defined by $d_{i+1} = Ap_{i+1}$.

A more robust and stable (yet less effective) simulation solver using implicit time integration is also available in our system. Analogous to the prior discussion, the state of the **B**-spline vertices at time $i + 1$ is integrated using prior states at time i and $i - 1$. Discrete derivatives of **p** are computed using backward differences

$$\ddot{\mathbf{p}}_{i+1} = (\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1})/(\Delta t^2),$$

and

$$\dot{\mathbf{p}}_{i+1} = (\mathbf{p}_{i+1} - \mathbf{p}_{i-1})/(2\Delta t).$$

We obtain the time integration formula

$$(2\mathbf{M}_p + \Delta t\mathbf{D}_p + 2\Delta t^2\mathbf{K}_p)\mathbf{p}_{i+1} = 2\Delta t^2 f_p + 4\mathbf{M}_p\mathbf{p}_i - (2\mathbf{M}_p - \Delta t\mathbf{D}_p)\mathbf{p}_{i-1}, \tag{10}$$

where

$$\mathbf{M}_p = \mathbf{A}^\mathrm{T}\mathbf{M}\mathbf{A},$$

$$\mathbf{D}_p = \mathbf{A}^\mathrm{T}\mathbf{D}\mathbf{A},$$

$$\mathbf{D}_p = \mathbf{A}^\mathrm{T}\mathbf{K}\mathbf{A}$$

and the subscripts denote evaluation of the quantities at the indicated timesteps. The matrices and forces are evaluated at time i. Instability due to large transient applied forces may be mitigated by adaptively reducing the size of the integration time step. It is straightforward to employ the conjugate gradient method to obtain an iterative solution for $\mathbf{p}_{i+1}$ [18]. The discretized mesh vector $\mathbf{d}_{i+1}$ can be updated in a similar way. The time performance of various solvers is detailed later in Section 6.

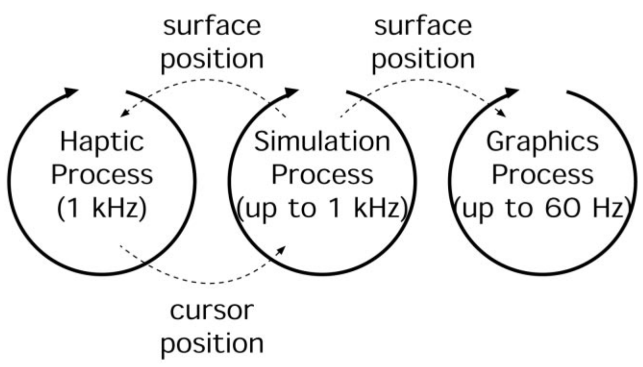

Fig. 11. The system is composed of three loosely coupled, independent processes which control the input, output, and physical simulation.

5.7. Multi-threading

The system is designed to run multi-threaded, preferably on multiple processors. There are three processes that need to run simultaneously at different rates (see Fig. 11). First, the simulation runs at the maximum possible speed so that the surface movement is as realistic as possible. Second, the graphics process runs at its maximum speed (up to 60 Hz), always using the most recent discretization from the simulation thread. Finally, the haptics process always runs at 1000 Hz or above and it supplies the simulation process with the most recent cursor position. The haptics process also computes output forces based on the finger position and the estimated surface position. A hardware mechanism in the haptics device ensures that the update rate is at least 1000 Hz or else it signals the process to exit.

For simple systems, the simulation process can be rolled into the haptic process. In our system, the simulation is very complex, and thus the computation could not always be completed in the 1 ms allotted. Therefore, we used three processes and decoupled the interfaces using the techniques presented in Ref. [9].

Normally, the simulation process takes longer than the haptic process, so the haptic process must estimate the resultant force several times in between updates. However, the force is highly non-linear and difficult to estimate due to discontinuities such as surface constraints. The position of the surface changes relatively slowly, while the haptic device updates very rapidly. Therefore, it is better to estimate the surface position and compute the force directly using the known cursor position, similar to the methods used in Refs. [10,22].

Consider computing the force using only a piecewise constant estimation of the surface position. This yields a step-like curve shown in Fig. 12. This way, the user would feel a snapping, as if pulling the surface over ridges. A better technique would use linear (or higher degree) extrapolation to predict the new surface position based on two (or multiple) previous states. Unfortunately, extrapolation amplifies small measurement errors, potentially leading to even greater discontinuities. Alternatively, we have implemented an exponentially weighted average which has the desirable characteristics of smoothing the result. The underlying mathematics can be simply formulated as

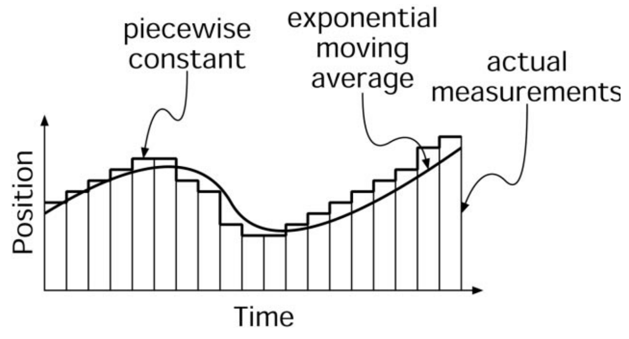

Fig. 12. A graphical depiction of two different position estimation schemes.

$$E_{i+1} = E_i\alpha + F_i(1-\alpha),$$

where E_i is the expected value at time i, $\alpha \in (0,1)$ is the smoothing factor, and F_i is the value of the sampled function at time i. We can see how this works by carrying out the iteration

$$E_{i+1} = F_i(1-\alpha) + F_{i-1}(1-\alpha)\alpha + F_{i-2}(1-\alpha)\alpha^2 + F_{i-3}(1-\alpha)\alpha^3 + \ldots + E_0\alpha$$

This shows that the jth most recent function sample is multiplied by the smoothing factor α raised to the jth power, leading to an exponential decay in influence. A more sophisticated method would first automatically determine the uncertainty (i.e. noise) in the system and subsequently improve the computation accuracy with the extra corrected term using filtered extrapolation. One typical example is the Kalman filter technique by Azuma and Bishop for their motion predictor in a virtual reality application Ref. [1].

6. Results

This section presents the graphic interface functionalities, details our experimental results, and discusses the system limitation.

6.1. Graphic interface

The **B**-spline surface is shaded with two colors and deformed into an odd shape (see Fig. 13). The control points, not shown here, seem to be arranged randomly for this shape. It would have been difficult and non-intuitive to specify this shape just by placing various control points in three dimensions with a mouse. The user's cursor is the sphere, which has been attracted to the surface. Not shown is a control panel that operates in 3D with sliders to control the overall mass, damping, and stiffness variables. Some constraints have been specified and are shown as small indicators on the surface.

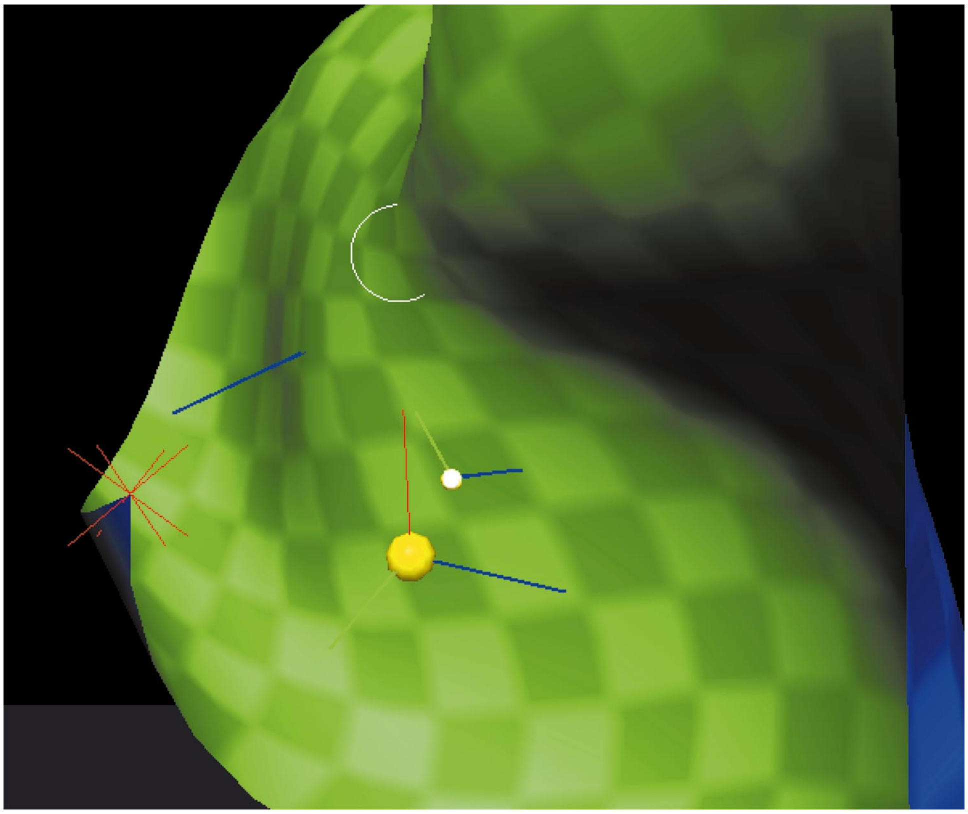

Fig. 13. Screen shot showing a shaded B-spline surface, the user's cursor as a large sphere and colored axes, the nearest point on the surface as a white sphere and colored axes, point constraints as red jacks, a normal constraint as a blue vector, and a curvature constraint as a white circle. (See also on the color page).

At any time during the simulation, pressing a key will toggle the existence of a point, normal, or curvature constraint at the point nearest to the cursor on the discretized mesh. A jack is drawn on the surface to indicate that a constraint has been added. The point constraint is drawn as a red cross, a normal constraint as an oriented blue line, and a curvature constraint as a white circle. The constraints are initially specified at their current quantities, i.e. the current position is used as the location for a point constraint, the current normal for a normal constraint, and the current curvature for the curvature constraint. Curvature constraints are specified separately in the two parametric directions, and a 2D map allows interactive modification of their values.

Using a finite difference approach with a mass-spring haptic system is computationally efficient; because only springs are used to model the physics, the force calculations can be optimized. Furthermore, mass, stiffness, and damping are quantities that are allowed to vary over the surface according to arbitrary distribution functions. In our system, the quantities vary according to a 2D map, like a texture map, which is evaluated using a bilinear resampling lookup. The haptic device enables intuitive modification of the map; the haptic device operates as an airbrush, literally painting mass, stiffness, and damping on the B-spline surface. Pressing a certain key turns off normal light shading and turns on map display, in which the 2D mass, stiffness, or damping map is projected onto the surface using pseudo-colors to represent value (see Fig. 14).

The curvature may be evaluated at every point on the surface and visualized in real-time (Fig. 15). The curvature is then visualized by mapping the u curvature to the magenta channel of the surface color and v the yellow. This style of visualization *and interactive fairing* has been extensively employed in the automotive industry.

6.2. Experiments and performance

We have presented a method capable of modeling dynamic free-form surfaces through an intuitive user interface. The user interface integrates precise 3D input with haptic feedback of a surface model. The method is capable of modeling B-spline surfaces by an innovative, physics-based formulation. By representing the surface in both physical and geometric domains, a useful B-spline representation can be generated to represent a dynamic surface with any user-specified physical properties. By using the exact physical properties of the surface, the designer can work with material and dynamics in virtual reality, gaining an intuitive understanding of its malleability.

The B-spline surface can be generated with a variable number of control points and with a variable number of discretized elements over the surface. We have examined the run times achieved for physical simulations using

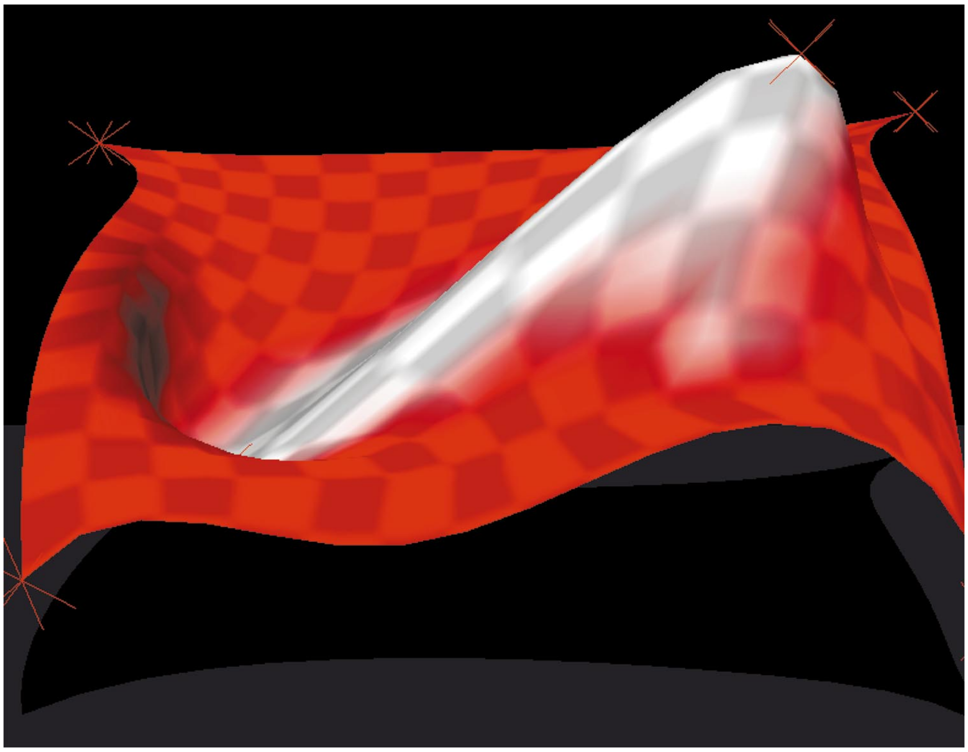

Fig. 14. Stiffness quantities are projected onto the B-spline surface, then are interactively changed by "airbrushing" with the haptic device. In this example, the white area was brushed with a lower stiffness, thus allowing it to stretch outward more than the rest of the surface. (See also on the color page).

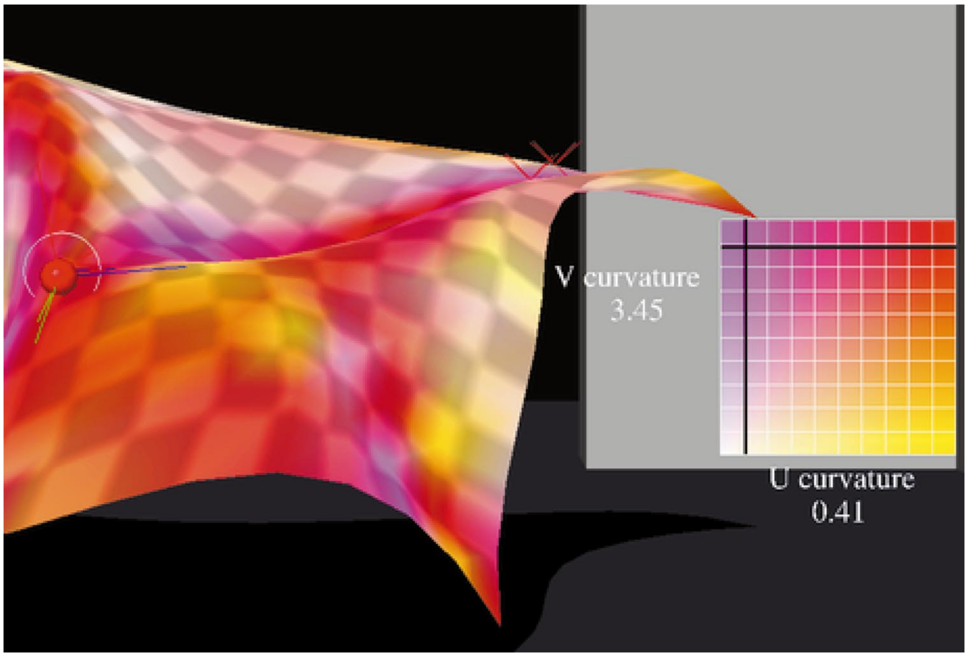

Fig. 15. The B-spline curvature is evaluated over the surface and displayed in pseudo-color. The user is able to interactively modify constraints and immediately visualize the results. In this case, the *u* curvature is shown using the magenta channel and *v*, the yellow. (See also on the color page).

various configurations of the control points and the discretized mesh (Table 1). Theoretically, the timing achieved is on the order of $O(s + m + c)$ where s is the number of springs, m is the number of mass points, and c is the number of control points. As the numbers indicate, the simulation update rate is inversely proportional to both the number of control points and the number of mass/spring elements.

The times for reasonably sized meshes are on the order of hundreds of updates per second, which provides a markedly realistic simulation of real surfaces. Subjectively, as the complexity of the surface increases, the surface reacts sluggishly since the propagation time with finite difference depends on the size of the mesh. This feature does not harm the ability to manipulate a surface; it only hampers the ability to make quick movements. If the timestep were indiscriminately increased to account for the time discrepancy for larger meshes, then stability problems would result; therefore, an adaptive timestep (as described in Section 5.6) is used with success.

6.3. Discussion

The penalty method used to optimize multiple constraints suffers from the limitation that constraints are not always met exactly. If conflicting constraints are specified, then the method globally minimizes the energy set by the penalties — considered the best solution by most metrics. Because it is possible to specify conflicting constraints, some constraints may not be precisely met. In the case of conflicting constraints, weight may be given to particular constraints by increasing their strength via the particular spring constant.

When the surface has homogeneous, isotropic spring constants, then the surface minimizes energy by attempting an equal surface area parameterization. Since the parameterization is fixed to equal intervals, it achieves this by moving the control points. Even without explicit point, normal, or curvature constraints, the surface is constrained by the surface area energy. Thus, as few as two point constraints may actually over-constrain the system, leading to a minimum energy configuration without meeting both point constraints exactly.

The described system has been demonstrated to a wide variety of audiences consisting of both experts and non-experts. The general assessment of the system based on users' feedback has been that it provides a powerful new tool for design because of the realism and spatial understanding that it imparts. Users claim to feel as if they are directly holding and manipulating the surface without appreciable lag and the spontaneous feedback force of the haptics device prevents users from "punching through" the surface. Although the surface visually reacts to the input rotation, users still *feel* as if they are connected to the surface by a ball joint without rotational feedback. The feel of the system would be improved if the currently used haptics device were capable of not only sensing rotation, but also imparting torque to the user. It might be helpful to selectively enable four different combinations of force and torque input and feedback at different sessions for precise sculpting. The density of discretization does not affect the quality of the simulation (the surface feels the same) as long as (1) the system is able to provide a high enough update rate and (2) the mass is properly distributed among the discrete mass points. Without the proper adjustment for the parameterization, the surface can feel artificially heavy and dampened.

While highly realistic, the system does not yet achieve the kind of performance and interactivity that is possible with the human hand. With only six degrees of freedom operating at 1 kHz, the haptic device cannot simulate the nearly infinite degrees of freedom represented by a real elasto-plastic material. The particular model chosen — a single rectangular B-spline surface — limits our ability to model complex shapes of arbitrary genus. As the mesh becomes distorted, the static parameterization may unequally distribute physical parameters (e.g. mass) leading to subtle non-realistic behavior. Our model does not simultaneously extend the surface with additional patches when deformations become large, as it would be difficult to maintain consistent dynamics during transition. However, this research provides a groundwork for future implementations of the method using alternate representations (e.g. general D-NURBS surfaces, subdivision surfaces, and even Catmull–Clark subdivision solids [12]).

7. Conclusion

We have presented a novel haptics-based interface and sculpting system that facilitates the direct manipulation of dynamic surfaces based on a B-spline formulation. The 3D haptics-based interface is more intuitive and natural than conventional 2D mouse-based interfaces. We have demonstrated a desktop haptic modeling system which is suitable for a spectrum of users ranging from highly trained engineering designers, computer professionals, artists, to even computer illiterates. Our system offers users a set of interaction toolkits, supporting point, normal, tangent, and curvature manipulation via haptic feedback devices. We have formulated a dual representation for dynamic surfaces satisfying various geometric, material, and elastic properties for the maximum dynamic realism. We anticipate that haptics offers great promise in interactive graphics, geometric modeling and design, medical training and simulation, and virtual environments.

It may be noted that the ever increasing amount of human–computer bandwidth inherent to various haptics devices provides the possibility for an increase of design productivity, thus we shall further our efforts towards the quantitative analysis of haptic sculpting effectiveness and its implications for the CAD/CAM industry. Our other future research agenda is to extend the functionalities of the haptic modeling system to handle multiple dynamic objects and their realistic interaction within a virtual design environment. First, our haptic approach should be generalized to allow multiple connected and disconnected patches. When gluing two patches together, for example, continuity requirements must be maintained. This is a challenging task to enforce smoothness criteria throughout physics-based haptic interaction. More powerful formulations, such as D-NURBS and dynamic subdivision-based models for arbitrary topology, will also be investigated. More advanced intuitive toolkits will be explored and developed towards the ultimate industrial practice.

Acknowledgements

This work was partially supported by NSF grant MIP9527694, NSF CAREER award CCR9896123, NSF grant DMI9896170, ONR grant N000149710402, Intel Corp, and Ford Motor Corp. The authors wish to thank Justine Dachille, Kevin McDonnell, John Woodwark, and several anonymous reviewers for their helpful comments.

References

[1] Azuma R, Bishop G. A frequency-domain analysis of head-motion prediction. SIGGRAPH 95 Conference Proceedings, Annual Conference Series, August 1995. p. 401–8.

[2] Bloor MIG, Wilson MJ. Representing PDE surfaces in terms of B-splines. Computer-Aided Design 1990;22(6):324–31.

[3] Celniker G, Gossard D. Deformable curve and surface finite elements for free-form shape design. Computer Graphics (SIGGRAPH'91 Proceedings), vol. 25, 1991. p. 257–66.

[4] Celniker G, Welch W, Linear constraints for deformable B-spline surfaces. In: Zeltzer D, editor. Computer Graphics (1992 Symposium on Interactive 3D Graphics), vol. 25(2), 1992. p. 165–70.

[5] Gleicher M. Integrating constraints and direct manipulation. Computer Graphics (1992 Symposium on Interactive 3D Graphics), vol. 25(2), 1992. p. 171–4.

[6] Grimm C, Ayers M. A framework for synchronized editing of multiple curve representations. Eurographics'93, Barcelona, Spain, September 1998.

[7] Halstead M, Kass M, DeRose T. Efficient, fair interpolation using Catmull–Clark surfaces. In: Kajiya JT, editor. Computer Graphics (SIGGRAPH'93 Proceedings), vol. 27, 1993. p. 35–44.

[8] Hsu WM, Hughes JF, Kaufman H. Direct manipulation of free-form deformations. Computer Graphics (SIGGRAPH'92 Proceedings), vol. 26, 1992. p. 177–84.

[9] Jacobs MC, Livingston MA, State A. Managing latency in complex augmented reality systems. Computer Graphics Symposium on Interactive 3D Graphics), 1997. p. 49–54.

[10] Mark W, Randolph S, Finch M, Van Verth J, Taylor II RM. Adding force feedback to graphics systems: issues and solutions. SIGGRAPH 96 Conference Proceedings, Annual Conference Series, August 1996. p. 447–52.

[11] Massie TM, Salisbury JK. The PHANToM haptic interface: a device for probing virtual objects. ASME haptic interfaces for virtual environment and teleoperator systems 1994, Dynamic Systems and Control 1994, vol. 1, 1994. p. 295–301.

[12] McDonnell K, Qin H. Dynamic modeling and sculpting of Catmull–Clark subdivision solids. Proceedings of Computer Animation 2000.

[13] Miller T, Zeleznik RC. The Design of 3D Haptic Widgets. Computer Graphics Symposium on Interactive 3D Graphics), 1999. p. 97–102.

[14] Minsky M, Ouh-Young M, Steele O, Brooks FP, Jr, Behensky M. Feeling and seeing: Issues in force display. Riesenfeld R, Sequin C, editors. Computer Graphics (1990 Symposium on Interactive 3D Graphics), vol. 24(2), 1990. p. 235–43.

[15] Moreton HP, Sequin CH. Functional optimization for fair surface design. Computer Graphics (SIGGRAPH'92 Proceedings), vol. 26, 1992. p. 167–76.

[16] Morgenbesser HB. Force shading for haptic shape perception in haptic virtual environments. M. eng thesis, Massachusetts Institute of Technology, 1995.

[17] Platt JC, Barr AH. Constraint methods for flexible models. In: Dill J, editor. Computer Graphics (SIGGRAPH'88 Proceedings), vol. 22, 1988. p. 279–88.

[18] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C: the art of scientific computing. 2nd ed. Cambridge: Cambridge University Press, 1992 (ISBN 0-521-43108-5).

[19] Qin H, Mandal C, Vemuri BC. Dynamic Catmull–Clark subdivision surfaces. IEEE Transactions on Visualization and Computer Graphics 1998;4(3):215–29.

[20] Qin H, Terzopoulos D. D-NURBS: a physics-based framework for geometric design. IEEE Transactions on Visualization and Computer Graphics 1996;2(1):85–96.

[21] Ruspini DC, Kolarov K, Khatib O. The haptic display of complex graphical environments. SIGGRAPH 97 Conference Proceedings, Annual Conference Series, August 1997. p. 345–52.

[22] Salisbury JK, Tarr C. Haptic rendering of surface defined by implicit functions. ASME Dynamic Systems and Control Division, November 1997.

[23] Srinivasan MA, Basdogan C. Haptics in virtual environments: taxonomy, research status, and challenges. Computer and Graphics 1997;21(4):393–404.

[24] Stewart PJ, Beier K-P. Direct manipulation of free-form curves with generalized parametric basis functions. Technical report, Personal Communication, 1998.

[25] Surles MC. An algorithm with linear complexity for interactive, physically-based modeling of large proteins. In: Catmull EE, editor. Computer Graphics (SIGGRAPH '92 Proceedings), vol. 26, July 1992. p. 221–30.

[26] Surles MC. Interactive modeling enhanced with constraints and physics with applications in molecular modeling. In: Zeltzer D, editor. Computer Graphics (1992 Symposium on Interactive 3D Graphics), vol. 25(2), March 1992. p. 175–82.

[27] Swarup N. Haptic Interaction with deformable objects using real-time dynamic simulation. MS thesis, Massachusetts Institute of Technology, 1995.

[28] Tarr CM. Rigid, plastic, and visco-elastic haptic surface interaction. Advanced undergraduate thesis, Massachusetts Institute of Technology, 1998.

[29] Terzopoulos D, Fleischer K. Deformable models. The Visual Computer 1988;4(6):306–31.

[30] Terzopoulos D, Platt J, Barr A, Fleischer K. Elastically deformable models. In: Stone MC, editor. Computer Graphics (SIGGRAPH'87 Proceedings), vol. 21, July 1987, p. 205–14.

[31] Thingvold JA, Cohen E. Physical modeling with B-spline surfaces for interactive design and animation. Computer Graphics (1990 Symposium on Interactive 3D Graphics), vol. 24(2), March 1990. p. 129–37.

[32] Thompson TV, Johnson DE, Cohen E. Direct haptic rendering of sculptured models. Computer Graphics (1997 Symposium on Interactive 3D Graphics), 1997. p. 167–76.

[33] Welch W, Witkin A. Variational surface modeling. Computer Graphics (SIGGRAPH'92 Proceedings), vol. 26, July 1992, 157–66.

[34] Zilles CB, Salisbury JK. A constraint-based god-object method for haptic display. ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, Dynamic Systems and Control 1994, vol. 1, November 1994. p. 146–50.

Frank Dachille received a BS in Naval Architecture and Marine Engineering from Webb Institute of Naval Architecture in 1994 on a full-tuition scholarship. He worked for two years developing collaborative virtual reality environments at Concurrent Technologies Corporation. He is currently a PhD candidate in Computer Science at the State University of New York at Stony Brook, where he is a research assistant in the Center for Visual Computing (CVC) headed by Dr Arie Kaufman. His current research interests include global illumination, volume visualization, volume rendering architectures, physics-based modeling, and virtual reality. For more information, see http://www.cs.sunysb.edu/~dachille.

Hong Qin is an Assistant Professor of Computer Science at State University of New York at Stony Brook, where he is also a member of the Center for Visual Computing (CVC). He received his BS (1986) degree and his MS degree (1989) in Computer Science from Peking University in Beijing, People's Republic of China. He received his PhD (1995) degree in Computer Science from the University of Toronto. During 1989–1990, he was research scientist at North-China Institute of Computing Technologies. During 1990–1991, he was a PhD candidate in Computer Science at the University of North Carolina at Chapel Hill. During 1996–1997, he was an Assistant Professor of Computer and Information Science and Engineering at the University of Florida. He received the Honor Student Award from 1983 to 1985 and the Best Graduate Award in 1996 from Peking University. During his years at the University of Toronto, he received a University of Toronto Open Doctoral Fellowship. In 1997, Dr Qin was awarded NSF CAREER Award from the National Science Foundation (NSF). He is a member of ACM, IEEE and SIAM.

Arie E. Kaufman is the Director of the Center for Visual Computing (CVC), a Leading Professor and Chair of Computer Science, and Leading Professor of Radiology at the State University of New York at Stony Brook. He was the founding Editor-in-Chief of the IEEE Transaction on Visualization and Computer Graphics (TVCG), 1995–1998. Kaufman has been the co-Chair for the multiple Eurographics/Siggraph Graphics Hardware Workshops, the Papers or Program co-Chair for the IEEE Visualization '90–'94 and ACM Volume Visualization Symposium '92, '94, '98, and the co-founder and member of the steering committee of the IEEE Visualization conference series. He has previously chaired and is currently a director of the IEEE Computer Society Technical Committee on Computer Graphics. He is the recipient of a 1995 IEEE Outstanding Contribution Award, the 1996 IEEE Computer Society's Golden Core Member, 1998 IEEE Fellow, 1998 ACM Service Award, and 1999 IEEE Computer Society's Meritorious Service Award. Kaufman has conducted research and consulted for about 30 years specializing in volume visualization; graphics architectures, algorithms, and languages; virtual reality; user interfaces; and multimedia. He received a BS in Mathematics and Physics from the Hebrew University of Jerusalem in 1969, an MS in Computer Science from the Weizmann Institute of Science, Rehovot, in 1973, and a PhD in Computer Science from the Ben-Gurion University, Israel, in 1977. For more information see http://www.cs.sunysb.edu/~ari.