Haptic Sculpting of Dynamic Surfaces

Frank Dachille IX, Hong Qin, Arie Kaufman, and Jihad El-Sana[†]
Center for Visual Computing (CVC)[‡]
and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

Conventional free-form surface design usually require tedious control-point manipulation and/or painstaking constraint specification via unnatural mouse-based interfaces. This paper presents a novel haptic approach for the direct manipulation of physics-based B-spline surfaces. Our method permits users to interactively sculpt virtual *yet real* material with a standard haptic device, and feel the physically realistic presence of virtual B-spline objects with force feedback throughout the design process. We aim to develop various haptic sculpting tools to expedite the direct manipulation of B-spline surfaces with haptic feedback and constraints. One significant contribution of this paper is that point, normal, and curvature constraints can be specified interactively and modified naturally using forces. We propose and formulate a dual representation for B-spline surfaces in both physical and mathematical space. This mass-spring model is mathematically constrained by the B-spline surface throughout the sculpting session. The equations of motion controlling the physical behavior of the B-spline surface are solved using a tractable numerical solver in real-time. The integration of haptics with traditional geometric modeling will increase the bandwidth of human-computer interaction, and thus shorten the time-consuming design cycle. We envision that this integrated approach promises a much greater potential in computer-integrated design and manufacturing, haptic interface, interactive graphics, medical applications, and virtual environments.

CR Categories: I.3.5 [Computer Graphics]: Physics-based modeling; I.3.3 [Computer Graphics]: Modeling packages; I.3.6 [Computer Graphics]: Interaction techniques; I.3.7 [Computer Graphics]: Virtual reality; I.3.7 [Computer Graphics]: Animation;

1 INTRODUCTION

Free-form surface modeling is critical to a wide range of areas including real-time interactive graphics, computer-aided geometric design (CAGD), scientific visualization, medical imaging, and virtual environments. During the past three decades, numerous free-form surface representations have been proposed for the modeling, design and manufacturing of aircraft, ship hulls, automobile bodies, various industrial parts, and consumer products. Although the theoretical foundations and mathematical properties of free-form surfaces have been extensively researched, better and more efficient modeling techniques using free-form surfaces have been evolved rather slowly. In a nutshell, traditional free-form surface modeling is usually associated with the tedious and indirect shape manipulation through time-consuming operations on a large number of (oftentimes irregular) control vertices. Despite the advent of various modern 3D graphics interaction tools, these indirect geometric techniques remain inherently laborious. In contrast, physics-based modeling offers a superior approach to free-form surface design that can overcome most of the limitations associated with conventional geometric modeling techniques. Within the physics-based framework, free-form surface 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 extremely intuitive and natural manner (refer to [13, 14] for the detailed advantages of the physics-based modeling methodology).

In spite of the recent research advances in physics-based modeling, it is *not yet* possible to achieve the full modeling potential intrinsically associated with the physics-based framework. This is because commonly-used interactive graphics systems often depend upon mouse-based 2D interfaces for *3D* interactions. The *direct* and *physical* operations on *real* objects via a 2D mouse are both unnatural and counter-intuitive. This paper presents a novel haptic approach for the intuitive and natural design of free-form surfaces within the physics-based framework. Haptics provides a means for intuitive manual interaction between users and virtual environments in which tactile exploration/manipulation of various objects can be obtained. We aim at the interactive and intuitive shape sculpting of full-scale physical objects in virtual environments. The integration of haptics with virtual environments will significantly contribute to various virtual reality (VR) applications such as health care (e.g., surgery simulation for medical training), entertainment (e.g., video games), education (e.g., the feeling of nano, macro, astronomical phenomena), and industry (e.g., the future haptics-based CAD systems).

The remainder of the paper is organized as follows. Sec. 2 motivates the work and details the novel contributions. Sec. 3 reviews prior research work. Sec. 4 presents the algorithm, including the mathematical formulations, the sculpting tools, constraint specification, and the calculation of force feedback. Sec. 5 describes the details of our system implementation. Sec. 6 presents the application examples. Sec. 7 concludes the paper and points out the future research directions.

2 MOTIVATION AND CONTRIBUTION

Throughout a large variety of interactive graphics methods, few computer-based modeling techniques have come close to enabling

[†] {dachille|qin|ari|jihad}@cs.sunysb.edu
[‡] http://www.cvc.sunysb.edu

modelers to design various shapes directly with their best *toolkit* — human being's hands. The ideal and intuitive modeling scenario is to allow users to reach toward the object, touch the object, feel the physical presence of the shape, grab the object, manipulate the object (with or without deformation), and move the object to the desired location. Our goal is to simulate as realistically as possible this interactive hand-based design paradigm. With a standard haptic device, our approach permits users to interactively sculpt virtual *yet real* material and feel the physically realistic presence of virtual objects with force feedback throughout the design process.

One key contribution of this research is the integration of haptics with the computer-integrated design and manufacturing cycle. Conventional 3D shape manipulation can be achieved through (1) the perspective projection of 3D objects onto a typical computer screen, and (2) shape modification and refinement via 2D mouse-based interaction. Using haptics in a virtual design environment, designers are faced with real objects, they will be able to feel and deform the object in a much more natural 3D setting. The tactile exploration can make designers gain a richer understanding of the 3D nature using the ideal tool, i.e., human being's hand, for spatial and temporal interaction. A number of researchers have utilized haptics to simulate the "feeling" in a virtual environment and to transform rigid objects in a synthetic scene. We aim to integrate haptics with physics-based modeling and interactive graphics in order to sculpt physics-based deformable models in the virtual environment using force-based tools as if we are creating and modifying real objects. The use of haptics in a virtual design environment increases the bandwidth of information between designers and the synthetic modeling world. Furthermore, the use of haptics in design, analysis, and manufacturing processes definitely shortens the product development cycle, enhancing the effectiveness of the design and analysis process for industry.

Prior research results primarily focuses on haptic rendering (i.e., the feeling of rigid surfaces/solids). Our haptic modeling system allows modelers to interactively deform a 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 universally familiar behaviors of physical, real-world objects. The dynamic behavior 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 [14] object with fixed weights. Our on-going research endeavor is to further extend the haptic system to sculpt D-NURBS objects.

We develop numerous high-level haptic sculpting tools to expedite the intuitive modification of surface objects in the most natural, intuitive means. The haptic device keeps track of both 3D position and orientation and updates these geometric quantities in real-time throughout the sculpting session. This information offers designers complete spatial and temporal control over various toolkits. Users can directly manipulate the free-form surface with haptic feedback and constraints. The toolkits 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 toolkits 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 mesh, control points and their associated basis functions become transparent to modelers in our haptic design environment, only the object of interest remains.

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. Furthermore, in mathematical space, this mass-spring model is constrained 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 using a tractable numerical solver in real-time. Note that, the polygonal representation can be approximated to any user-specified error tolerance, making it useful for simultaneous graphics rendering as well as haptic rendering.

3 RELATED WORK

Various methods have been developed to generate fair surfaces which satisfy multiple constraints and optimize an energy-based objective function [2, 11, 27]. It is also possible to construct dynamic surfaces with natural behavior governed by physical laws [12, 24]. Terzopoulos and Fleischer [23] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [2] developed an interesting prototype system for interactive design based on the finite-element optimization of energy functionals. Bloor and Wilson [1] used similar energies optimized through numerical methods for B-splines. Celniker and Welch [3] investigated deformable B-splines with linear constraints. Welch and Witkin [27] proposed variational surface modeling method. Thingvold and Cohen [25] proposed to use elasto-plastic mass-spring-hinge models on the B-spline control points. Moreton and Sequin [11] interpolated a minimum energy curve network with quintic Bezier patches by minimizing the variation of curvature. Stewart and Beier [18] demonstrated a direct curve manipulation technique which allows the direct control of position, normal, and curvature. Halstead *et al.* [6] implemented smooth interpolation with Catmull-Clark surfaces using a thin-plate energy functional. Other relevant work of physical models include the realistic animation of volumetric objects and molecular structure [12, 19, 20, 24], and the direct manipulation of spline models [4, 7]. Our method dynamically sculpts 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 [5] proposed a framework for curve editing. Multiple representations of a curve must be maintained in order to speed up the curve modification in the most appropriate domain. In particular, a particle representation allow 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.).

Haptics and its associated techniques are also well researched in recent years. A good review of haptics literature can be found in [17]. Minsky *et al.* [10] investigated the conditions required to sustain the illusion in a haptic system. Thompson *et al.* [26] investigated haptic rendering of NURBS surfaces. Haptic rendering requires (1) sensing the position of user's finger, (2) locating the nearest point of the surface, and (3) appropriately generating a force to be applied to the finger. One prime difficulty of realistic haptic modeling is speed. Convincing haptic feedback requires updates on the order of 1000 Hz, this is far greater than the necessary threshold required for real-time visual display (up to 60 Hz). This hard constraint is due to the fact that touch is sensitive to frequencies up to 1000 Hz, a less update rate results in tactile vibrations which are perceived as roughness. To tackle this problem, Jacobs *et al.* [8] discussed critical issues of synchronizing sensors in virtual reality environments. We apply their methods to synchronize multiple simulation loops, (i.e., graphics, haptics, dynamics simulation). Alternatively, Randolph *et al.* [9] 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 discovered by Salisbury and Tarr [16]. Recently, Tarr [22] and Swarup [21] explored physically-based haptic interaction. Moreover, Ruspini *et al.* [15] introduced the notion of a haptic proxy as a massless sphere that moves along the objects in the environment.

4 ALGORITHM

Our haptic system executes in a tight loop. 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 position and rotation of the haptic device. The commands impart sculpting forces to the physical model, and the internal force are calculated. The forces are applied, and the system estimates the position of dynamic model at the next time step. Note that, throughout the design sculpting, 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 multithread the haptics, graphics, and dynamics with weak synchronization. This technique lead to the parallel processing of haptic sculpting. We now detail major components of the haptic interaction.

4.1 Dynamic Surface

We represent a continuous B-spline surface $\mathbf{s}(u, v)$ 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:

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

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. There parametric domain is determined by two sets of nondecreasing knot sequences, respectively. In the interest of the space here, we refer readers to [14] 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 $\mathbf{p}_i = [x, y, z]^\top$:

$$\mathbf{p} = [\mathbf{p}_{0,0}^\top, \mathbf{p}_{0,1}^\top, ..., \mathbf{p}_{n,m}^\top]^\top,$$

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

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

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

The discretized dynamic model has material quantities such as mass, damping, and stiffness distribution. These values are distributed over the surface defined functions $\mu(u, v), \gamma(u, v)$, and $\rho(u, v)$, respectively, which often can be considered to be constant. As demonstrated later in Sec. 5, our system allows the user to *paint* these properties interactively and directly on 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. Alternatively, the dynamic surface can be approximated using finite element method based on $\mathbf{d}$. The finite elements derived from $\mathbf{d}$ are mathematically equivalent to our current implementation. We use a mass-spring model instead because of its simplicity. 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:

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

The force at every mass-point in the mesh is the sum of all possible external forces: $\mathbf{f}_{\mathbf{d}} = \sum \mathbf{f}_{ext}$. The internal forces are generated by the connecting springs, where each spring is modeled with force $\mathbf{f} = k\mathbf{l}$. 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, we shall formulate the motion equation of physical behavior for all the control points:

$$\mathbf{A}^\top\mathbf{M}\mathbf{A}\ddot{\mathbf{p}} + \mathbf{A}^\top\mathbf{D}\mathbf{A}\dot{\mathbf{p}} + \mathbf{A}^\top\mathbf{K}\mathbf{A}\mathbf{p} = \mathbf{A}^\top\mathbf{f}_{\mathbf{d}}. \tag{4}$$

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

$$\mathbf{A}^\top\mathbf{M}\mathbf{A}\ddot{\mathbf{p}} + \mathbf{A}^\top\mathbf{D}\dot{\mathbf{d}} + \mathbf{A}^\top\mathbf{K}\mathbf{d} = \mathbf{A}^\top\mathbf{f}_{\mathbf{d}} \tag{5}$$

$$\mathbf{A}^\top\mathbf{M}\mathbf{A}\ddot{\mathbf{p}} = \mathbf{A}^\top\mathbf{f}_{\mathbf{d}} - \mathbf{A}^\top\mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^\top\mathbf{K}\mathbf{d} \tag{6}$$

$$\ddot{\mathbf{p}} = (\mathbf{A}^\top\mathbf{M}\mathbf{A})^{-1}(\mathbf{A}^\top\mathbf{f}_d - \mathbf{A}^\top\mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^\top\mathbf{K}\mathbf{d}). \tag{7}$$

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 certain user-defined proportion. Non-linear damping reduces the velocity of each part model point by 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 make use of the evaluation of relatively inexpensive function to the inner loop of stiffness and damping equations.

Error control is an integral part of the algorithm. Errors are possibly introduced into the design system by either coarse timesteps or a low-resolution discretization. The timestep is normally very small since the haptic device requires a high update rate. If the model is very complex, the timestep may become large and errors creep into the simulation unless an adaptive timestep 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 modelers.

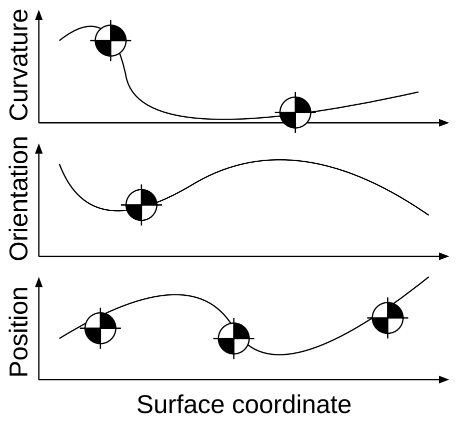

Figure 1: Surface constraints are specified in multiple domains, and the dynamic model evolves to optimize the shape subject to those constraints.

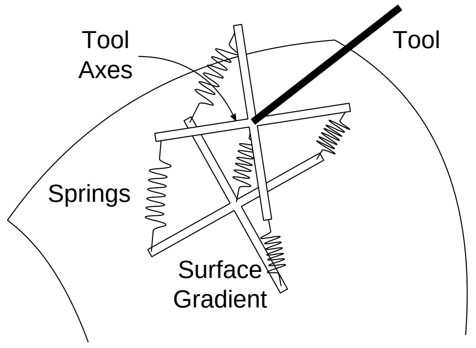

Figure 2: The set of five springs is used to locate and orient the surface relative to the cursor position.

The surface dynamically optimizes its energy functional, imparting realism during the modeling session. The modeler has the ability to specify constraints in several domains simultaneously (see Fig. 1). Constraints are supplied in physical space, tangent space, and curvature space, corresponding to location, first derivative, and second derivative, similar to [18].

4.2 Sculpting Tools

External forces are generated by modelers' input and constraints. The user's cursor is attached to the surface by an incompressible spring — *rope*. The user can not push on 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 length of the rope decreases over time in such a manner that within one second, the length becomes zero and the user is connected directly to the object. 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.

The rope tool always grabs the nearest mass-point on the surface from the cursor. An alternative magnet tool allows the user to grab a nearby subset of the mass-points simultaneously. The force is then distributed among nearby points using user-defined function $\theta(x, y, z)$, which can be a constant, spherical, cylindrical, conical, or any other distributions.

If the user wishes to not only place, but also to orient the 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 (see Fig. 2). 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.

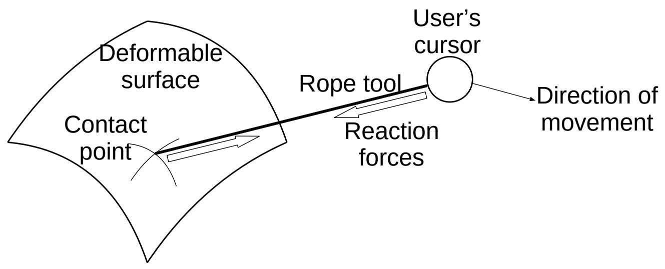

Figure 3: To compute the haptic reaction force, Newton's third law is applied. By pulling on the *rope* tool, tension is generated in the rope pulling in opposite directions for the surface and the user.

4.3 Force Feedback

By adding external input force 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. 3). When the rope tool pulls on a mass-point by adding a force along 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 special haptic device that we use only senses rotation; therefore, it is not possible to transmit torque to the user. If torque motors are available, the quality of the simulation will 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 sphere, force is always transmitted to apply on relevant mass-points which reflect forces back to the user along the opposite direction during each timestep. A simple vector calculation (e.g., addition and minus) leads to the exact reaction force in all cases.

4.4 Constraints

Many methods have been used to implement constraints. Hsu *et al.* [7] 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 [27] utilized Lagrange multipliers to enforce a least-squared solution to a constraint matrix. Moreton and Sequin [11] used a minimum-energy network to optimize a system of linear and non-linear constraints. Terzopoulos [24] used the penalty method to drive a dynamic deformation for animation. Qin and Terzopoulos [14] used linear constraint techniques to deform physical models for design purposes. Platt and Barr [12] discussed various constraint methods for deformable models including the penalty method, reaction constraints, Lagrange constraints, and augmented Lagrange constraints. Among various techniques to handle constraints, penalty methods exhibit the property of simplicity, but suffer from inexact solutions and the need for small 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 equation that are not suitable for a 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, in general they require a much higher computational effort. Tremendous computation costs 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 Fig. 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 the adaptive timestep, i.e., if the acceleration exceeds user-defined threshold, the timestep 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, it is physically plausible, and other methods fail to solve non-linear systems which are oftentimes 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. We use the penalty method since this constraint is non-linear. When a tangent plane constraint is enabled, the current discrete $\frac{\Delta \mathbf{s}}{\Delta u}$ and $\frac{\Delta \mathbf{s}}{\Delta v}$ 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 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 $\mathbf{v}_d$. 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 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. 4). 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 inversely proportional to the desired curvature at that point. The desired curvature is variable, and it can be set independently for u and v by sliders. The actual curvature could be either positive or negative. This method does not force the sign of the curvature, only the 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.

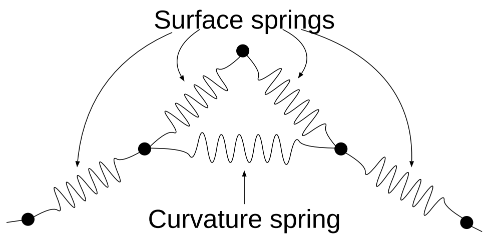

Figure 4: A curvature spring is added to the surface to cause bending to a desired curvature

4.5 Numerical Integration

At each timestep of the simulation, the state of the system is advanced based on the previous equations. 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:

$$\mathbf{f}_\mathrm{p} = \mathbf{A}^\top \mathbf{f}_\mathrm{d}$$

The velocity $\frac{\Delta \mathbf{p}}{\Delta t}$ of the control points is updated according to the applied forces and 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 $\mathbf{d}_{i+1} = \mathbf{A}\mathbf{p}_{i+1}$.

5 IMPLEMENTATION

The haptic system runs on both Windows NT and IRIX machines with a Phantom haptic device from Sensable Technologies. The B-spline surface is shaded with two colors and deformed into an odd shape. The control points, not shown here, seem to be arranged randomly for this shape (see Fig. 5). 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.

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 an oriented blue line, and a curvature constraint 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. The curvature constraint is specified in the two parametric directions, and 2D map allows interactive modification of the amount of curvature once it has been constrained.

The system is designed to run multithreaded, preferably on multiple processors. There are three processes that need to run simultaneously at different rates (see Fig. 6). First, the simulation runs at the maximum possible speed so that the surface movement is as

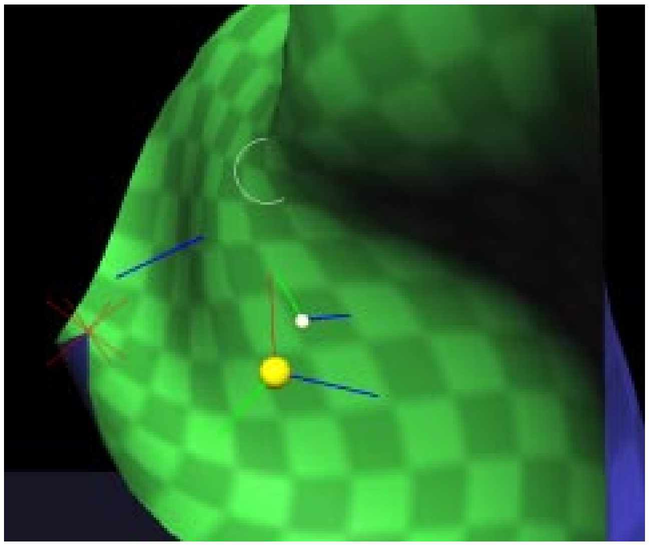

Figure 5: Screen shot showing a shaded B-spline surface, the user's cursor as a large sphere and colored axes, the nearest point on the surface as a white sphere and colored axes, point constraints as red jacks, a normal constraint as a blue vector, and a curvature constraint as a white circle. (See also in color section.)

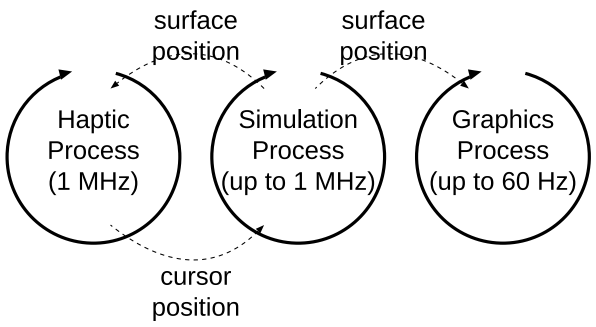

Figure 6: The system is composed of three loosely coupled, independent processes which control the input, output and physical simulation.

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 the computation could not always be completed in the 1ms allotted. Therefore, we used three processes and decoupled the interfaces using the techniques presented in [8].

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 nonlinear 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 by [9, 16].

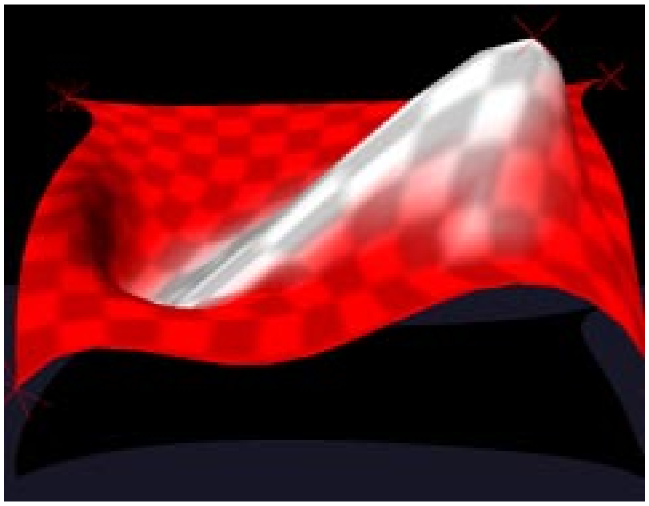

Figure 7: Stiffness quantities are projected onto the B-spline surface, then are interactive changed by "airbrushing" with the haptic device. In this example, the white area was brushed with a lower stiffness, thus allowing it to stretch outward more than the rest of the surface. (See also in color section.)

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. 7).

The curvature may be evaluated at every point on the surface and visualized in real-time (see Fig. 8). The curvature is then visualized by mapping the u curvature to the green channel of the surface color and v the blue. This type of visualization *and interactive fairing* has been extensively employed in the automotive industry.

6 RESULTS

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 (see Fig. 9). We have examined the timings achieved for the physical simulation using various configurations of the control points and the discretized mesh (see 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

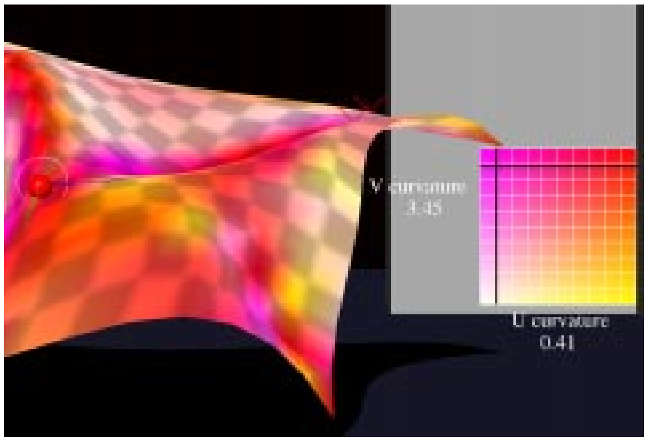

Figure 8: The B-spline curvature is evaluated over the surface and displayed in pseudo-color, then the user is able to interactively modify constraints and immediately visualize the results. In this case, the u curvature is shown using the green color channel and v, the blue. (See also in color section.)

Table 1: Physical simulation timings using both an explicit and an implicit solver versus the variation of surface complexity.

Control point resolution	Discretized mesh resolution	Explicit timing	Implicit timing
4×4	10×10	.5ms	10ms
4×4	20×20	2ms	11ms
4×4	40×40	10ms	19ms
8×8	20×20	5ms	100ms
8×8	40×40	25ms	120ms
12×12	25×25	42ms	780ms

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 width 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 is used with success.

The penalty method used to optimize multiple constraints suffers from the limitation that exact constraints are not always met. 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

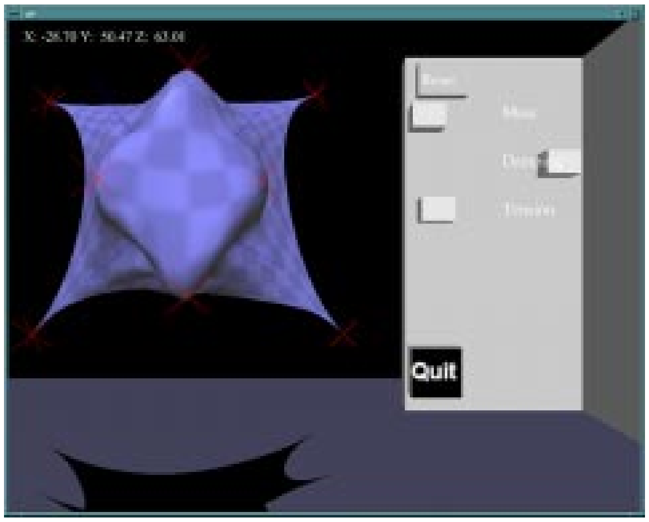

Figure 9: The application screen showing the user interface and a complex dynamic surface is generated from a 12×12 B-spline control mesh. It is much more intuitive to manipulate the surface directly rather than to determine the positions of the complex control mesh. (See also in color section.)

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.

7 CONCLUSION

We have presented a novel haptic approach for the direct sculpting of dynamics surfaces based on B-spline formulation. The haptics is more intuitive and natural than conventional mouse-based interfaces. We have demonstrated a haptic system which offers users to a set of interaction toolkits, supporting point, normal, tangent, and curvature manipulation via haptic feedback devices. Haptic sculpting capability, incorporated into traditional CAD/CAM industry, can (1) expedite the geometric process of conceptual design, (2) increase the bandwidth of human-computer interaction, and (3) reduce the cost of the entire product development cycle. 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 medical applications, and virtual environments.

It may be noted that the ever-increased amount of human-computer bandwidth inherent to various haptics devices provides the possibility for a significant increase of design productivity and effectiveness, thus we shall further our efforts towards the quantitative analysis of haptic bandwidth and its implication for human communication, knowledge and understanding as well as for the CAD/CAM industry. Our other future research agenda is to extend the functionalities of the haptic modeling system to cover numerous complicated objects and realistic interaction of multiple objects within a virtual design environment. First, our haptic approach should be generalized to allow multiple 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 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.

8 ACKNOWLEDGMENTS

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 for her review, ideas, and support.

References

[1] M. I. G. Bloor and M. J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.

[2] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 257–266, July 1991.

[3] G. Celniker and W. Welch. Linear constraints for deformable B-spline surfaces. In D. Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 165–170, Mar. 1992.

[4] M. Gleicher. Integrating constraints and direct manipulation. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 171–174, Mar. 1992.

[5] C. Grimm and M. Ayers. A framework for synchronized editing of multiple curve representations. In *Eurographics '93*, Barcelona, Spain, Sept. 1998.

[6] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 35–44, Aug. 1993.

[7] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 177–184, July 1992.

[8] M. C. Jacobs, M. A. Livingston, and A. State. Managing latency in complex augmented reality systems. In *Computer Graphics (1997 Symposium on Interactive 3D Graphics)*, pages 54–49, 1997.

[9] W. R. Mark, S. C. Randolph, M. Finch, J. M. V. Verth, and R. M. T. II. Adding force feedback to graphics systems: Issues and solutions. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 447 – 452. ACM SIGGRAPH, ACM Press, August 1996.

[10] M. Minsky, M. Ouh-young, O. Steele, F. P. Brooks, Jr., and M. Behensky. Feeling and seeing: Issues in force display. In R. Riesenfeld and C. Sequin, editors, *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24(2), pages 235–243, Mar. 1990.

[11] H. P. Moreton and C. H. Sequin. Functional optimization for fair surface design. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 167–176, July 1992.

[12] J. C. Platt and A. H. Barr. Constraint methods for flexible models. In J. Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 279–288, Aug. 1988.

[13] H. Qin, C. Mandal, and B. C. Vemuri. Dynamic catmull-clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, July 1998.

[14] H. Qin and D. Terzopoulos. D-NURBS: A physics-Based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, Mar. 1996.

[15] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 345–352, Aug. 1997.

[16] J. K. Salisbury and C. Tarr. Haptic rendering of surface defined by implicit functions. In *ASME Dynamic Systems and Control Division*, November 1997.

[17] M. A. Srinivasan and C. Basdogan. Haptics in virtual environments: taxonomy, research status, and challenges. *Computers & Graphics*, 21(4):393–404, July 1997.

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

[19] M. C. Surles. An algorithm with linear complexity for interactive, physically-based modeling of large proteins. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 221–230, July 1992.

[20] M. C. Surles. Interactive modeling enhanced with constraints and physics with applications in molecular modeling. In D. Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 175–182, Mar. 1992.

[21] N. Swarup. *Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation*. Ms thesis, Massachusetts Institute of Technology, 1995.

[22] C. M. Tarr. *Rigid, Plastic, and Visco-Elastic Haptic Surface Interaction*. Advanced undergraduate thesis, Massachusetts Institute of Technology, 1998.

[23] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, Dec. 1988.

[24] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 205–214, July 1987.

[25] J. A. Thingvold and E. Cohen. Physical modeling with B-spline surfaces for interactive design and animation. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24(2), pages 129–137, Mar. 1990.

[26] T. V. Thompson, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Computer Graphics (1997 Symposium on Interactive 3D Graphics)*, pages 167–176, 1997.

[27] W. Welch and A. Witkin. Variational surface modeling. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 157–166, July 1992.