Virtual Clay: Haptics-based Deformable Solids of Arbitrary Topology

Kevin T. McDonnell and **Hong Qin**

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794–4400
{ktm|qin}@cs.sunysb.edu

Abstract. This paper presents Virtual Clay as a novel, interactive, dynamic, haptics-based deformable solid of arbitrary topology. Our Virtual Clay methodology is a unique, powerful visual modeling paradigm which is founded upon the integration of (1) deformable models, (2) free-form, spline-based solids, (3) procedural subdivision solids of arbitrary topology, and (4) dynamic objects governed by physical laws. Solid geometry exhibits much greater modeling potential and superior advantages to popular surface-based techniques in visual computing. This is primarily because a CAD-based solid representation of a real-world physical object is both geometrically accurate and topologically unambiguous. We first introduce the concept of Virtual Clay based on dynamic subdivision solids. Then, we formulate the mathematics of Virtual Clay through the integration of the geometry of subdivision solids with the principle of physics-based CAGD. Our Virtual Clay models respond to applied forces in a natural and predictive manner and offer the user the illusion of manipulating semi-elastic clay in the real world. We showcase example sculptures created with our Virtual Clay sculpting environment, which is equipped with a large variety of real-time, intuitive sculpting toolkits. The versatility of our Virtual Clay techniques allows users to modify the topology of sculpted objects easily, while the inherent physical properties are exploited to provide a natural interface for direct, force-based deformation. More importantly, our sculpting system supports natural haptic interaction to provide the user with a realistic sculpting experience. It is our hope that our Virtual Clay graphics system can become a powerful tool in graphics, computer vision, animation, computer art, interactive techniques, and virtual environments.

1 Introduction and Rationale

Existing solid modeling techniques are oftentimes based on one or several of the following geometric foundations: implicit surfaces, including Constructive Solid Geometry (CSG) and blobby models; Boundary representations (B-reps), such as subdivision surfaces and general polygonal meshes; spline-based solids, including Bézier and B-spline solids; and cell decompositions, such as voxel-based models. Although these approaches are ideal for certain applications, each tends

to fall short in offering a flexible and unified representation that can be used to manipulate deformable solid objects of inhomogeneous material distributions. In this work we attempt to overcome some of the difficulties associated with existing modeling approaches by developing a physics-based, deformable, volumetric model founded upon *subdivision solids*. Our new model serves as the basis for a virtual sculpting system that features haptic interaction and a suite of intuitive sculpting tools.

Solid modeling approaches are inherently complex due to the great number of degrees of freedom that are required to represent real-world objects. In principle, such freedom is necessary in order to design complex mechanical parts and to model more organic or natural-looking objects. When interacting with spline-based solids, certain geometric quantities, such as the knot vectors in the formulation of Non-Uniform Rational B-splines (NURBS), have little intuitive or physical meaning. Users are typically forced to manipulate a large number of geometric parameters in order to make small changes to solid objects. This problem is especially severe when the modeled object consists of both complex interior structure and boundary information. The inclusion of interior information for solid modeling inevitably increases the number of degrees of freedom by at least one order of magnitude. Furthermore, most geometry-based interfaces permit only *indirect* interaction through control point manipulation, which is usually tedious and very time-consuming.

Physics-based modeling techniques can alleviate many of these problems by augmenting geometric objects with physical attributes such as mass, damping and stiffness distributions. Geometric parameters can be hidden from the user by providing natural, force-based interfaces that facilitate *direct* manipulation of solid objects through virtual sculpting tools. The system synchronizes the geometric and physical representations of objects in order to maintain the underlying geometric structures of sculpted solids. Such dynamic and interactive approaches can provide inexperienced users with a natural, force-based interface, while at the same time provide expert users with the familiar geometric interface if desired.

In this paper we use a physics-based approach to conceal from users the inherent topological and geometric complexity of subdivision solids. Physical attributes are assigned both on the boundary and in the interior of deformable subdivision solid objects. Meanwhile, we propose and develop a number of haptics-based tools that assist users during 3D interaction with force-feedback. In this manner users can use haptic exploration to gain a better understanding of the physical attributes of virtual sculptures. Haptics can enhance the functionality of physics-based models in a natural and intuitive way. Note that both physics-based modeling and haptic interaction are founded upon real-world physical laws. Therefore, their integration offers numerous new modeling, manipulation, and visualization advantages. For example, the haptic interface fundamentally improves understanding by permitting direct interaction with virtual objects and by enhancing the sense of realism through computer communication. Our deformable model is very general and should find application not only in vir-

tual sculpting, but also in education, surgical simulation, virtual reality, and entertainment.

2 Contributions

Our novel deformable model is based on a new type of subdivision model pioneered by MacCracken and Joy [10]. Whereas, their algorithm was conceived only as a new mechanism for free-form deformation (FFD), we employ their subdivision rules to systematically formulate free-form subdivision solids for physics-based volumetric sculpting. It may be noted that the new subdivision solids naturally include free-form volumetric splines such as B-spline solids as their special cases [12].

In comparison with existing modeling techniques associated with subdivision surfaces, subdivision solid formulations transcend surface-based approaches by defining geometry and topology both in the interior and on the boundary of solid objects. In our approach, we augment subdivision solids with physical properties that permit users to manipulate solids directly through a haptic interface and through physical forces. The inherent geometric parameters of the solid maintain the subdivision structure, while the physical attributes support direct manipulation. The geometric and topological complexities of subdivision solids are concealed by both our physics-based formulation and by an intuitive user interface. Our sculpting system provides users with many virtual tools that let them directly manipulate solid objects. The user can quickly and easily make topological, geometric and physical changes to sculpted objects without the need to understand the complicated mathematics of subdivision solids.

Our system offers both an explicit and an implicit solver. The former is efficient and is easy to implement, while the latter guarantees numerical stability. In addition, our physical model incorporates diverse types of elastic behavior that can produce structurally stable objects that exhibit great shape variation. Furthermore, we have devised and implemented a local, adaptive subdivision algorithm that can be employed to create refined features in any region(s) of interest. Additionally, our novel sculpting system provides the user with a suite of powerful, haptics-based tools that interact directly with our deformable modeling framework.

3 Background Review

3.1 Deformable Models

Our deformable subdivision solid model is based on well-established techniques and algorithms from deformable and physics-based modeling. Physically-based solid models were introduced to the graphics and vision communities by Terzopoulos and colleagues [14, 22–24]. In essence, the geometry of their models is discretized from continuous surfaces and solids and is attached to mass-spring

meshes. Others have used modal dynamics [17] and other finite element approaches [5] in order to improve the stability and accuracy of dynamic models. Baraff, Witkin, Kass and others [2, 3, 27] have developed techniques for animating and constraining non-penetrating dynamic surfaces. Qin and colleagues [11, 19, 20] derived dynamic models for direct manipulation of spline- and subdivision-based surfaces. James and Pai [9] developed a dynamic surface model based on the Boundary Element Method (BEM).

3.2 Subdivision Solids

Subdivision solids [10] have recently emerged as a generalization of tri-cubic B-spline solids to free-form solid models of arbitrary topologies. In general, B-spline solids and other spline-based solid representations are cumbersome to use in sculpting applications since many separate solids must be carefully patched together to create simple features such as handles and holes. Subdivision solids, in contrast, can represent a topologically complex, deformable object with a single *control lattice*. In contrast to the *control mesh* that defines a typical subdivision surface, which consists of points, edges and polygons, the control lattice of a subdivision solid consists of points, edges, polygons and closed polyhedra (which we call "cells"). Note that the use of cells results in the complete subdivision of the volumetric space occupied by the control lattice. In this manner it allows users to represent a volumetric object without any geometric ambiguity or topological inconsistency. The geometric construction of our deformable solid model is based on [10]. Since the original subdivision rules do not define the surface structure, we use the Catmull-Clark subdivision surfaces rules [4] to obtain smooth boundaries.

Because of their unique advantages [12, 13], subdivision solids can provide a very general framework for representing volumetric objects. While subdivision surfaces are very convenient and general for creating models in which only the boundary is important, subdivision solids can offer users additional flexibility whenever necessary or appropriate. Like subdivision surfaces, subdivision solids are natural candidates for multiresolution analysis and level-of-detail (LOD) control since one can subdivide a given model until the desired amount of detail is achieved. Subdivision solids depend solely on the use of simple subdivision algorithms that are similar to those found in most subdivision surface approaches. Thus, the solid subdivision algorithm is relatively easy to implement and does not require very sophisticated data structures.

3.3 Haptic Interfaces

Our sculpting system offers a number of haptics-based sculpting tools that attempt to enhance the sense of realism experienced by the user. A good review of haptics literature can be found in [21]. Thompson *et al.* [25] derived efficient intersection techniques that permit direct haptic rendering of NURBS surfaces. Miller and Zeleznik [15] defined a set of robust haptic principles and techniques that can be applied to nearly any type of haptic interface. Dachille *et al.* [6]

developed a haptic interface that permits direct manipulation of dynamic surfaces. Balakrishnan *et al.* [1] developed *ShapeTape*, a curve and surface manipulation technique that can sense user-steering bending and twisting motions of the rubber tape. To date, haptics has been an active area of research that has demonstrated great potential for the development of natural and intuitive human-computer interfaces.

4 Deformable Subdivision Solids

Our deformable subdivision solid model marries the geometric information and topological structure of subdivision solids with physical attributes and other relevant material quantities. Figure 1 shows a cubical control lattice after one level of subdivision. The complex cell structure in the interior of the solid is highlighted in Figure 1b. The difference in complexity between the subdivision solid wireframe (Figure 1c) and subdivision surface wireframe (Figure 1d) can also be easily observed. The second wireframe is that of a Catmull-Clark surface which coincides with the boundary of the solid. The first wireframe clearly demonstrates the topological and geometric complexity introduced by the cells of the solid. In order to manage the complex non-manifold topological nature of subdivision solids, we employ the radial-edge data structure invented by Weiler [26]. This data structure can be viewed as a generalization of the winged-edge data structure for polygonal surfaces to non-manifold objects. Note that, in the interest of clarity and convenience, we render only the boundary surfaces of solid sculptures for most of the examples in this paper. Nevertheless, a large amount of topological, geometric, physical and material information is employed for the interior description and is therefore hidden.

After a user-specified number of subdivisions of the control lattice, the resulting subdivision solid is assigned physical properties such as mass, damping, and stiffness. To avoid ambiguity, we use the term "subdivided lattice" to describe the control lattice after one or more applications of the subdivision algorithm. Material properties are assigned both in the inside and on the boundary of the subdivided solid. Note that a subdivision surface (or any other pure surface model) could not represent heterogeneous interior attributes because the interior of a subdivision surface is empty. The control lattice is retained but is not assigned any physical parameters. That is, material properties are associated only with the limit shape of subdivision solids and not with the initial control lattice. The control structure is required to maintain the geometric smoothness and continuity of the subdivided model.

Using our approach, each point in the subdivided solid is assigned an initial mass by the application. To avoid confusion with points in the control lattice, these points will henceforth be called "mass points." Vertices in the control lattice shall be called "control points." Each edge between mass points is assigned an initial spring stiffness. Such edges are termed "normal springs," while edges in the control lattice shall be called "control edges." Each face in the subdivided lattice is also assigned a set of "angular springs," which we describe in detail

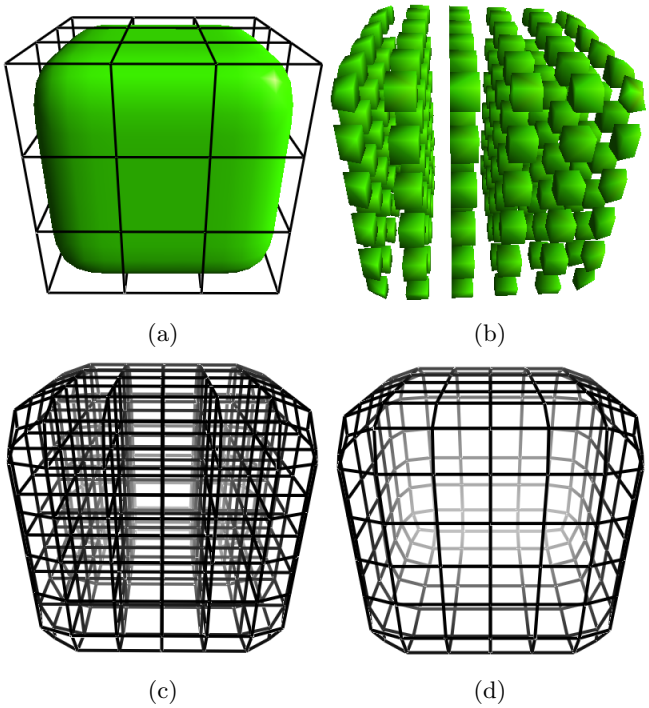


Fig. 1. (a) A subdivision solid after one level of subdivision along with its control lattice. (b) After scaling the cells, the complex interior structure becomes visible. (c) A wireframe rendering of the subdivided solid. (d) A wireframe of a Catmull-Clark subdivision surface that coincides with the boundary. The difference in geometric and topological complexity of the solid and surface is significant.

later. These faces in the subdivided solid are called "subdivided faces," while faces in the control lattice are termed "control faces." Similarly, cells in the subdivided lattice are "subdivided cells," and cells in the control lattice are "control cells." In summary, the initial control lattice is subdivided and is then used to define a mass-spring lattice containing mass points and two types of springs.

Like virtually all procedural subdivision algorithms, the subdivision solid representation can be expressed as a global matrix multiplication:

$$\mathbf{d} = \mathbf{A}\mathbf{p}. \tag{1}$$

The $\mathbf{p}$ is a column vector of the control point positions; the matrix $\mathbf{A}$ is a sparse, rectangular matrix that contains weights given by the subdivision rules; and the column vector $\mathbf{d}$ gives the positions of the mass points. Note that matrix $\mathbf{A}$ is a global subdivision matrix and that local changes to the subdivision rules can be expressed by changing a few rows of the matrix. Due to the sparsity of $\mathbf{A}$,

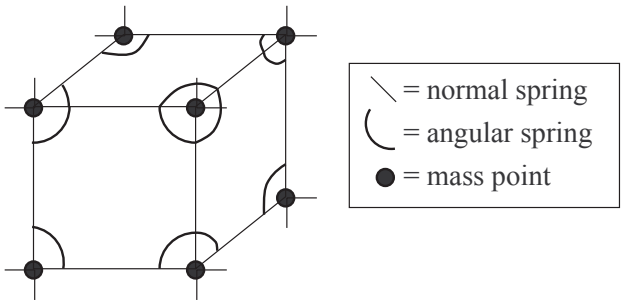


Fig. 2. A typical cell in the subdivided lattice. Normal springs (straight lines) resist stretching forces, while angular springs (arcs) resist shearing forces. Each subdivided edge is assigned a normal spring, and each vertex in each face is assigned an angular spring.

we employ a sparse matrix storage scheme to improve time performance and to save memory.

Figure 2 shows a typical hexahedral cell from a subdivided solid along with its mass points, normal springs, and angular springs. Normal springs are traditional linear springs found in many other deformable models and in introductory physics textbooks. Their end-points are mass points that appear in the subdivided lattice. The force exerted on each end-point is given by Hooke's law: $\mathbf{f} = -k(\mathbf{x} - \mathbf{x}^0)$, where k is the spring's stiffness, $\mathbf{x}$ is the current position of the end-point, and $\mathbf{x}^0$ is the rest position of the end-point. In our solid model, each normal spring can have a different stiffness and rest length, which permits the definition of a wide variety of physical behaviors.

Traditional mass-spring lattices often suffer from structural instability since they do not attempt to maintain internal angles (*i.e.*, to avoid shearing). Some other deformable models rely on diagonal springs to avoid these problems, but these types of springs can exert forces even when shearing is not occurring (see Figure 3). In order to better characterize an object's resistance to shearing forces, we employ angular springs instead. The system assigns one angular spring per mass point per subdivided face. An angular spring exerts forces on points only when its associated angle deviates from its initial value. The angular spring pushes its "end-points" away from (or towards) each other in order to regain the original angle. In this way the model attempts to enforce a soft curvature constraint on faces.

Figure 4 shows the arrangement of angular springs in a subdivided face. For each mass point in each subdivided face, we compute the initial rest angle of the two normal springs that coincide at that mass point. When the face undergoes forces as a result of shearing deformation, the angular springs respond by exerting forces on the two end-points to bring the angle back to its initial value (see Figure 5). The initial angle (θ^0) and current angle (θ) can be computed directly using the dot product of the two incident edges ($\mathbf{s}_1$ and $\mathbf{s}_2$): $\theta = \cos^{-1}\left(\frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{|\mathbf{s}_1||\mathbf{s}_2|}\right)$.

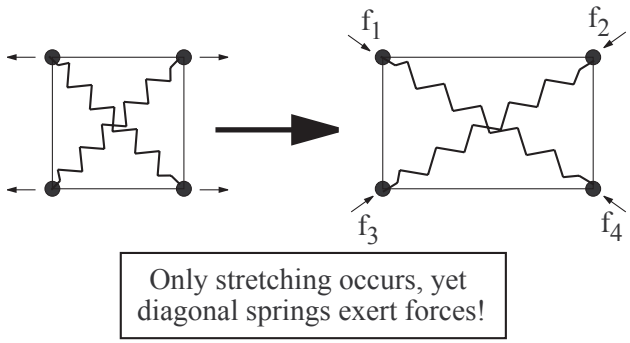


Fig. 3. Diagonal springs exhibit incorrect behavior when stretching occurs but not shearing, as this figure demonstrates. In this special case, diagonal springs erroneously exert forces to resist stretching. Angular springs (Figures 4 and 5) should be used instead in order to correctly characterize an object's resistance to shearing forces.

To compute the forces that should be applied to each end-point of $\mathbf{s}_1$ and $\mathbf{s}_2$ (indicated by $\mathbf{e}_1$ and $\mathbf{e}_2$ in Figure 5), we calculate where the end-points would be if the current angle equaled the rest angle (*i.e.*, $\theta = \theta^0$). This requires computing two rotations around the mass point. The axis of rotation is computed using the cross product of $\mathbf{s}_1$ and $\mathbf{s}_2$: $\mathbf{s}_1 \times \mathbf{s}_2$. The rotations provide two new *virtual* points, $\mathbf{e'}_1$ and $\mathbf{e'}_2$, from which we compute displacements $\mathbf{d_1} = \mathbf{e'}_1 - \mathbf{e}_1$ and $\mathbf{d_2} = \mathbf{e'}_2 - \mathbf{e_2}$. Using Hooke's law, linear spring forces are applied to $\mathbf{e_1}$ and $\mathbf{e_2}$ in the directions of $\mathbf{d_1}$ and $\mathbf{d_2}$ to help bring the angle back to its rest value.

It should be noted that our unique formulation of deformable subdivision solids and its aforementioned stiffness distribution (especially the structure of its angular springs) are founded upon concepts and their rigorous analysis from differential geometry [22]. Consider a 3×3 metric tensor function $\mathbf{G}(\mathbf{s})$:

$$\mathbf{G} = \begin{bmatrix} \frac{\partial \mathbf{s}}{\partial u} \cdot \frac{\partial \mathbf{s}}{\partial u} & \frac{\partial \mathbf{s}}{\partial u} \cdot \frac{\partial \mathbf{s}}{\partial v} & \frac{\partial \mathbf{s}}{\partial u} \cdot \frac{\partial \mathbf{s}}{\partial w} \\ \frac{\partial \mathbf{s}}{\partial v} \cdot \frac{\partial \mathbf{s}}{\partial u} & \frac{\partial \mathbf{s}}{\partial v} \cdot \frac{\partial \mathbf{s}}{\partial v} & \frac{\partial \mathbf{s}}{\partial v} \cdot \frac{\partial \mathbf{s}}{\partial w} \\ \frac{\partial \mathbf{s}}{\partial w} \cdot \frac{\partial \mathbf{s}}{\partial u} & \frac{\partial \mathbf{s}}{\partial w} \cdot \frac{\partial \mathbf{s}}{\partial v} & \frac{\partial \mathbf{s}}{\partial w} \cdot \frac{\partial \mathbf{s}}{\partial w} \end{bmatrix}, \tag{2}$$

where (u, v, w) is a local parameterization for $\mathbf{s}(\mathbf{x})$. One important result is that the above matrix tensor remains unchanged if there is no solid deformation for any solid object $\mathbf{s}(\mathbf{x})$ (*i.e.*, rigid-body motion does not modify the tensor function $\mathbf{G}(\mathbf{s})$). The amount of deformation of a given object is therefore defined by a function of $G^0_{ij} - G^n_{ij}$, which is the difference between the metric tensors of the initial and current states of the object, where the superscript denotes time, and the subscripts indicate matrix entries. Therefore, in the context of deformable subdivision solids, normal springs attempt to minimize the change for the diagonal terms of $\mathbf{G}(\mathbf{s})$, while the angular springs attempt to minimize the variation of the off-diagonal terms.

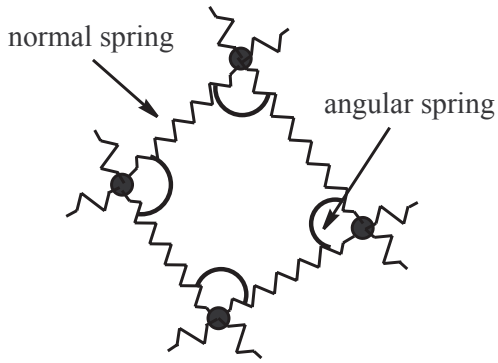


Fig. 4. To help maintain the structural stability of a dynamic subdivision solid, the system employs angular springs to help maintain internal angles and to counteract shearing forces. Traditional springs, indicated by jagged lines, are used to resist stretching forces.

5 Numerical Algorithms for Physics-Based Simulation

We have implemented both an explicit numerical solver for time integration and an implicit one. When extensive user interaction is required, the explicit solver is used to provide real-time update rates. The implicit solver can be invoked when numerical stability is more of a concern or when one is performing offline simulations.

5.1 Explicit Time Integration

Our dynamic model is founded upon the energy-based Lagrangian equation of motion, which has the following continuous and discrete forms, respectively:

$$\mu\ddot{\mathbf{s}} + \gamma\dot{\mathbf{s}} + \frac{\partial E(\mathbf{s})}{\partial \mathbf{s}} = \mathbf{f}, \tag{3}$$

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}_{\mathbf{x}}. \tag{4}$$

The $\mathbf{M}$, $\mathbf{D}$ and $\mathbf{K}$ matrices represent the mass, damping and internal energy distributions of an object; the $\mathbf{x}, \dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ represent the discrete position, velocity and acceleration of an object; and $\mathbf{f}_{\mathbf{x}}$ contains the total external forces acting an object.

We augment the discrete Lagrangian equation of motion with geometric and topological quantities derived from the subdivision solid algorithm. Recall that $\mathbf{p}$ is a vector containing the positions of the control points, and $\mathbf{d}$ stores the positions of the mass points. The $\mathbf{f}_{\mathbf{d}}$ collects the external forces acting on the mass points. The rectangular matrix $\mathbf{A}$ stores the subdivision weights that define the positions of the mass points as a function of the control points. Subject to the constraints defined by Equation 1, we augment the Lagrangian equation of

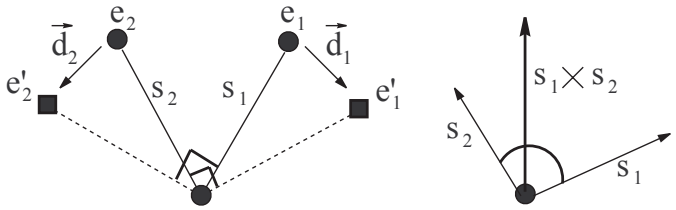


Fig. 5. To compute the forces exerted by an angular spring, we begin by computing where end-points $\mathbf{e_1}$ and $\mathbf{e_2}$ would be if there were no shearing (given by $\mathbf{e'}_1$ and $\mathbf{e'}_2$). The points are rotated about the mass point using the axis of rotation (given by the cross product $\mathbf{s_1} \times \mathbf{s_2}$). Using these virtual positions we compute a pair of displacements ($\mathbf{d_1}$ and $\mathbf{d_2}$). The displacements are used to provide the spring forces we apply to the end-points to help bring the angle back to its rest configuration. In this example we assume the rest angle is 90 degrees.

motion as follows. First we substitute $\mathbf{d}$ for the generic $\mathbf{x}$ and multiply each side by $\mathbf{A}^\top$:

$$\mathbf{A}^\top \mathbf{M}\ddot{\mathbf{d}} + \mathbf{A}^\top \mathbf{D}\dot{\mathbf{d}} + \mathbf{A}^\top \mathbf{K}\mathbf{d} = \mathbf{A}^\top \mathbf{f_d}.$$

By applying Equation 1 and rearranging terms we obtain:

$$\mathbf{A}^\top \mathbf{M}\mathbf{A}\ddot{\mathbf{p}} + \mathbf{A}^\top \mathbf{D}\mathbf{A}\dot{\mathbf{p}} + \mathbf{A}^\top \mathbf{K}\mathbf{A}\mathbf{p} = \mathbf{A}^\top \mathbf{f_d},$$

$$\mathbf{A}^\top \mathbf{M}\mathbf{A}\ddot{\mathbf{p}} = \mathbf{A}^\top \mathbf{f_d} - \mathbf{A}^\top \mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^\top \mathbf{K}\mathbf{d}.$$

Then we solve for the acceleration of the control points using a least-squares approach:

$$\ddot{\mathbf{p}} = (\mathbf{A}^\top \mathbf{M}\mathbf{A})^{-1}(\mathbf{A}^\top \mathbf{f_d} - \mathbf{A}^\top \mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^\top \mathbf{K}\mathbf{d}). \tag{5}$$

Note that Equation 5 essentially states that the external forces acting on the mass points are mapped to the control points (which are purely geometric objects) using the subdivision matrix. Once the accelerations of the control points are computed, the model's position and velocity are computed using a forward Euler method to drive the simulation:

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

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

5.2 Implicit Time Integration

While the explicit numerical solver derived above is easy to implement and is very efficient, it tends to suffer from numerical instability when large time-steps are taken or when spring stiffnesses are very high. To ameliorate this problem we derived and implemented an implicit solver based on backward Euler integration. Discrete derivatives are computed using backward differences:

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

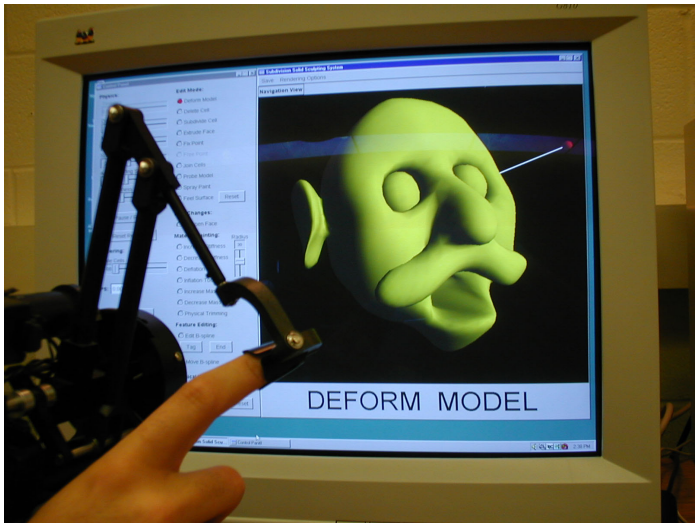


Fig. 6. The user interface consists of a PHANToM haptic device, a standard 2D mouse, and on-screen GUI controls.

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

We obtain the time integration formula

$$\begin{aligned} \left(2\mathbf{M_p} + \Delta t\mathbf{D_p} + 2\Delta t^2\mathbf{K_p}\right)\mathbf{p}_{i+1} = \\ 2\Delta t^2\mathbf{f_p} + 4\mathbf{M_p}\mathbf{p}_i - (2\mathbf{M_p} - \Delta t\mathbf{D_p})\mathbf{p}_{i-1}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} \mathbf{M_p} &= \mathbf{A}^\top\mathbf{M}\mathbf{A}, \\ \mathbf{D_p} &= \mathbf{A}^\top\mathbf{D}\mathbf{A}, \\ \mathbf{K_p} &= \mathbf{A}^\top\mathbf{K}\mathbf{A}, \end{aligned}$$

and the subscripts denote evaluation of the quantities at the indicated time-steps. It is straightforward to employ the conjugate gradient method [18] to obtain an iterative solution for $\mathbf{p}_{i+1}$. To achieve interactive simulation rates, we limit the number of conjugate gradient iterations per time-step to 10. We have observed that two iterations typically suffice to converge the system to a residual error of less than 10^{-4}. More than two iterations tend to be necessary when the physical parameters are changed dramatically during interactive sculpting.

6 Sculpting System and Toolkits

6.1 Architecture and Interface

The user interface (see Figure 6) of our sculpting system consists of a Sensable Technologies PHANToM 1.0 3D haptic input/output device, a standard 2D

mouse, and on-screen GUI controls. The PHANToM features a thimble for the user's index finger and can exert a real-world force in any 3D direction. The mouse is used in conjunction with the PHANToM to activate or enable over a dozen sculpting tools. Forces are exerted on the user's finger only when a haptics-based sculpting tool is active. The entire system runs on a generic Microsoft Windows NT PC with an Intel Pentium III 550 Mhz CPU and 512 MB RAM. Experimental results and system performance data are provided towards the end of the paper.

An overview of the physical simulation and design process is shown in Figure 7. After an initialization step in which data structures are assembled, the system runs in an infinite loop that continuously updates the physical state of a sculpture. The system traverses all springs and computes the total internal forces acting on the mass points. External forces are queried from the input devices and added to the system. The aggregate forces acting on the mass points are then mapped to the control vertices (see Section 5). The acceleration and velocity of the control lattice are then computed in order to move the control lattice to its new position. The subdivision matrix is applied to the control vertices to obtain the new positions of the mass points. This step is required to maintain the geometric and subdivision structure of the subdivided lattice. At any time during the simulation, the user may pause the execution and invoke a tool that causes a change in topology (for instance, to create a protrusion). This type of action requires re-subdivision of the geometry and the re-assembly of certain data structures.

A second thread is required to control the haptic interaction and to accept 3D input from the user. When any haptic tool is in use, this thread computes the haptic force to be felt by the user. Without a second thread of execution, the haptic device is not able to send real-time forces to the user's finger. This causes the PHANToM to buzz and vibrate, which is very distracting during sculpting.

Regarding data structures, most geometric quantities and matrices are stored in one- and two-dimensional arrays. The sophisticated topological structure of subdivision solids is represented using a simplified version of Weiler's radial-edge data structure [16, 26]. By storing adjacency information, this data structure allows our system to efficiently query and update topological information. Fast accesses are critical to make topological changes as efficiently as possible.

The remainder of this section briefly covers the sculpting functionality of our system. The interested reader is directed to [12, 13] for a full description of each sculpting tool.

6.2 Haptic Tools

Through our non-compressive "rope" tool (Figure 8a) the user can select any of the mass points, both on the boundary and in the interior, and exert a linear spring force on the point. This tool is employed to deform a shape much in the same way that an artist might mold a piece of clay with his fingers. The PHANToM exerts a real force on the user's hand that is a linear function of the mass of the attached point, the distance of the cursor to the point, and the

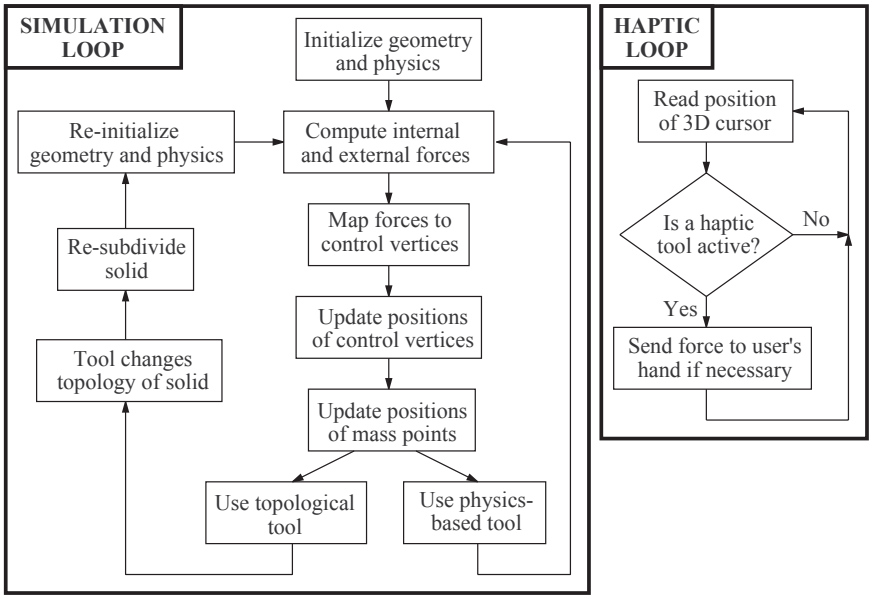
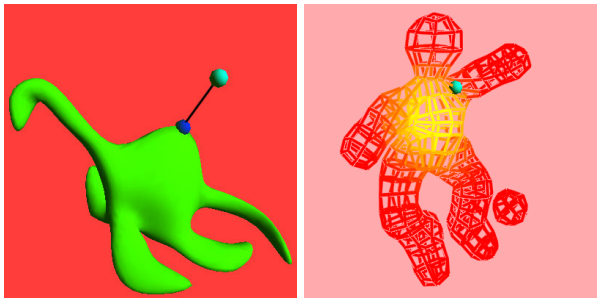


Fig. 7. An overview of the physical simulation and design process featured in our sculpting system. The application requires two separate execution threads in order to generate real-time haptic feedback.

strength of the rope tool (whose stiffness can be changed in the GUI). The user also has the ability to study an object's stiffness both graphically and haptically through our probing tool (Figure 8b). When active, the probing tool exerts a force on the user's hand proportional to the local stiffnesses of those springs within a given radius of the tool. In addition, the user can run the cursor over the surface of a solid to examine the boundary tactilely. In this way a designer can feel how smooth the surface is at the current subdivision level and also look for any surface anomalies that might be hard to detect visually.

6.3 Geometric and Topological Tools

Our geometric and topological tools operate directly on the control lattice. For instance, using the 3D cursor provided by the PHANToM, the user can carve material from a sculpture (Figure 9a). As with all of the tools that modify the topology of the sculpted object, the radial-edge data structure and certain matrices need to be re-computed to effect the change. Second, the extrusion tool (Figure 9b) can be employed to extrude new material from the face on the surface nearest the 3D cursor. When the user activates the tool, a new control cell is created and attached to the face. It "grows" in the direction of the surface normal, and its size is computed based on the size of the control cell to which it is attached. Third, two cells near the surface of a sculpture can be joined together



(a) (b)

Fig. 8. Haptics-based sculpting tools. (a) Rope tool. (b) Haptics-based probing.

by using our joining tool (Figure 9c). This tool is very useful for making loops, twisted cylindrical structures, ring-like features, and handles. In reality this kind of action is somewhat difficult to achieve even with actual clay.

Our local subdivision algorithm allows users to make very fine-level details on their models (Figure 9d). Much like the carving tool, the user selects a cell to subdivide with the 3D cursor. The system replaces the highlighted cell with several smaller cells whose structure is based on the normal subdivision solid rules. Faces and edges in adjacent cells are also subdivided in order to avoid T-shapes and other topological anomalies (such as a vertex in the middle of a face). This process is illustrated in Figure 10. Although this procedure complicates the local topological structure, the resulting control lattice is still valid and can be treated and manipulated like any other subdivision control lattice.

Sharp features (Figure 9e) such as corners and creases can be created by tagging faces, edges, or points as "sharp". In our implementation, sharp features are created by using different subdivision rules for tagged geometric elements. Note that since we use a Catmull-Clark surface to represent the boundary, one can use other special rules, such as those found in [7, 8, 10].

6.4 Deformation Tools

Our sculpting system features a number of tools that use physics to interact with a model. The use of physical tools frees the user from the need to deal with the multitude of control vertices inherent in subdivision solid objects. In addition, the user can interact with deformable subdivision solids directly, rather than through indirect and often non-intuitive control point manipulation.

For example, our B-spline manipulation tool (Figure 11a) lets the user select mass points from a sculpture, build a cubic B-spline curve that interpolates the points, and then manipulate the spline using its control points. In this way the user can use a B-spline to quickly design curved regions as well as make large, well-controlled deformations. It is straightforward to generalize this tool to B-spline surfaces and other types of curve and surface representations.

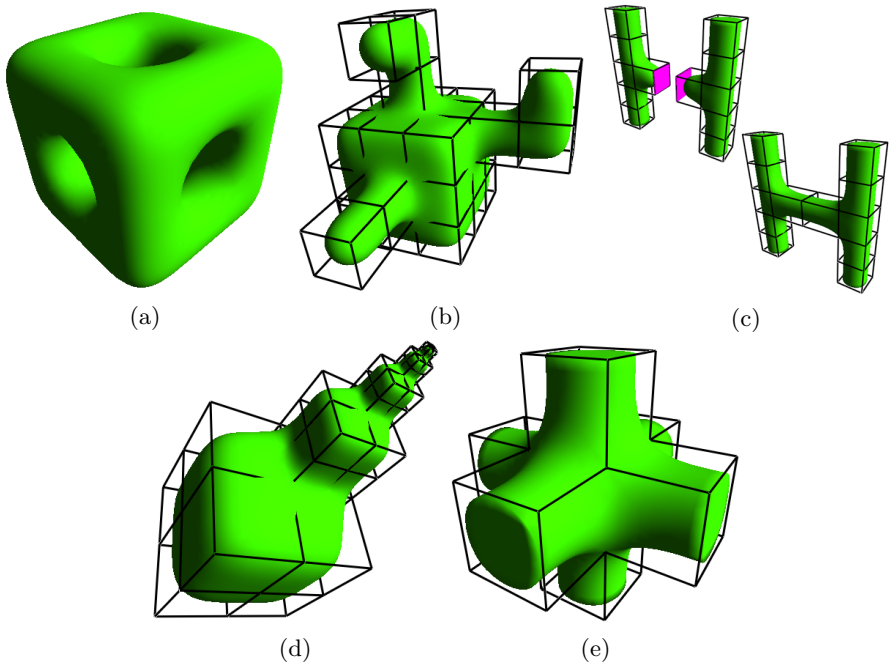


Fig. 9. Geometry-based and topology-based sculpting tools. (a) Carving. (b) Extrusion. (c) Joining. (d) Detail editing. (e) Sharp feature creation.

Since our model defines geometric and physical properties on both the boundary and interior, the user has the option of modifying the mass and stiffness distributions both in the inside and on the boundary (Figure 11b). Our virtual sculpting material is more general than real clay since it can actually have different physical properties throughout the sculpture. This offers extra flexibility and advantages to the user. For instance, the user can use such functionality to localize the effects of certain physics-based tools, especially the rope tool.

The inflation and deflation tools (Figures 11c and 11d) allow the user to grow and shrink local portions of the model, respectively. The rest lengths of springs are increased or decreased, depending on whether inflation or deflation is taking place. The inflation tool is typically used to create bumps and rounded features, while the deflation tool is useful for shrinking and tapering parts of sculptures.

The physical trimming tool (Figure 11e) allows users to effectively "turn off" cells in the subdivided lattice. Unlike the cell-deletion tool, which modifies the topology of a sculpture, this tool directly operates on the physical representation. It enables the user to prevent portions of a model from participating in the physical simulation.

Sometimes the user may wish to deform only a local region of a sculpture. This action can be facilitated by using our "physical window" feature (Figure 11f) that constrains the physical simulation to occur in a local region of

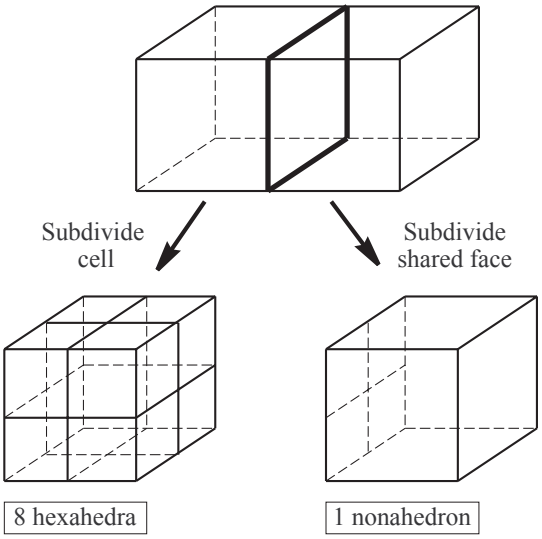


Fig. 10. The cell on the left is subdivided locally. The face indicated by heavy lines is shared between the two cells. This face must be subdivided to maintain topological consistency.

interest. In addition to localizing the effect of deformation, the physical window speeds up the simulation and improves frame rates by reducing the amount of data that needs to be processed by the system. For instance, the subdivision matrix $\mathbf{A}$ as well as the vectors $\mathbf{d}$ and $\mathbf{p}$ can be partitioned into "free" $(\mathbf{A}_1, \mathbf{p}_1, \mathbf{d}_1)$ and "fixed" $(\mathbf{A}_2, \mathbf{p}_2)$ matrices:

$$\mathbf{d} = \mathbf{A}\mathbf{p} \qquad \rightarrow \qquad \mathbf{d}_1 = \mathbf{A}_1\mathbf{p}_1 + \mathbf{A}_2\mathbf{p_2}.$$

The fixed portions can be computed once and stored for look-up later. Other matrices such as $\mathbf{M}$, $\mathbf{D}$ and $\mathbf{K}$ can be modified in a similar manner.

6.5 Additional GUI Controls

In addition to the sculpting tools, our GUI contains several widgets that allow the user to change the global state of the model and various simulation parameters, such as the time-step, damping coefficient, spring stiffness values, etc. The user can also change the stiffness (or strength) of the rope tool using a slider in the GUI. High values of this stiffness allow the user to made large, sweeping deformations in a short amount of time. Also, the user can press the "Reset Rest Shape" button in order to redefine the rest shape of a sculpture to its current shape. Without this functionality a sculpted object would always revert to its initial physical state due to the elasticity of the springs.

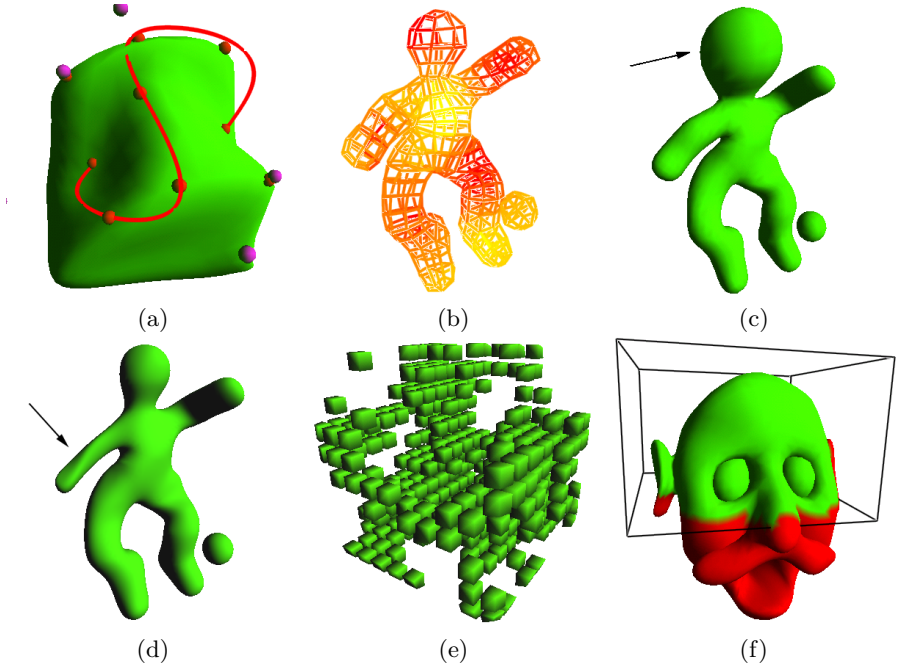


Fig. 11. Physics-based sculpting tools. (a) B-spline curve manipulation. (b) Stiffness painting. (c) Deflation. (d) Inflation. (e) Physical trimming. (f) Physical window.

7 Experimental Results and Time Performance

Table 1 details the data sizes and run-time statistics for the sculptures featured in this paper. This table contains the running times only for the simulation loop. Note that in order to generate final images, we subdivided the models several additional times to produce smooth boundary surfaces seen in the figures.

Table 1. Data-set sizes and simulation timing information for some of the models described in this paper. All statistics are for one level of subdivision, except for the soccer player, which was subdivided twice.

Model	**Control Cells**	**Subdivided Cells**	**Update Time (ms)**
Man's head	123	1069	114.7
Soccer player	24	1536	85.0
Chair with holes	76	744	82.5
Chair without holes	62	552	50.6
Mug	102	816	80.9

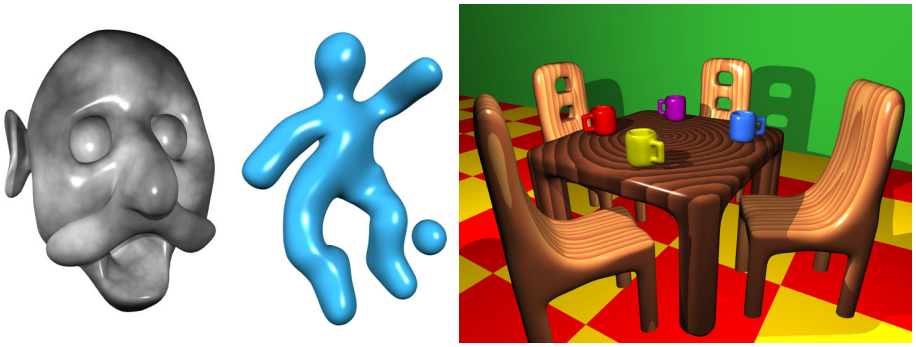


Fig. 12. Several sculptures created entirely with our system and then composed and rendered with POV-Ray (www.povray.org).

7.1 Example Sculpture: Human Head

We shall use the sculpted cartoon head in Figure 12 to describe how our sculpting tools can be used in practice. The sculpting of the head required the use of almost every design tool, and so it serves as a good example of how to use our sculpting system. We began with a $5 \times 5 \times 5$ block containing 125 control cells. Cells on the underside and back were carved out to give the head its general shape. The eye sockets were created by removing one cell each. The eyes were then inserted by extruding a single cell from each hole. The nose was extruded from the front of the face and then shaped with the rope tool. The mustache was extruded from the nose, bent, and slightly twisted downward. The mouth was created first by deleting a cell, by locally subdividing the lower jaw, and then by stretching it to create the thin lower lip and protruding chin. The ears were sculpted by extruding a few cells on either side of the head, locally subdividing them, and then stretching and denting them to make them thin and oval. The deformation and inflation tools were employed to give the top of the head a more rounded appearance. Then the deflation tool was used to smooth out bumps and pits. For certain features the physical window was activated to constrain the influence of the rope tool on the model. Finally, the surface of the solid was extracted and ray-traced offline.

8 Conclusions

We have developed a novel deformable solid modeling framework along with its intuitive, natural haptic interface based on the unique integration of subdivision solids and physics-based techniques. Our system for shape design and sculpting permits the user to create real-world, complicated models in real-time using an extensive suite of geometry-based, topology-based, physics-based, and haptics-based tools. Within our modeling environment, the virtual clay responds to

direct, user-applied forces in a predictable and intuitive manner while the haptic feedback can significantly enhance the sense of realism. Our physical formulation of topologically complex subdivision solids frees users from the need to deal with abstract geometric quantities and esoteric mathematical structures that often hinder the extensive use of sophisticated solid models on a large scale. To verify the applicability of our sculpting system on virtual clay and its powerful toolkits, we have conducted many sculpting sessions which have resulted in numerous interesting sculptures. The gallery of our virtual sculptures in Figure 12 (see [12, 13] for more examples) should appeal to a wide spectrum of users including researchers, designers, animators, artists and even non-professionals.

Acknowledgments

This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, the NSF ITR grant IIS-0082035, the NSF grant IIS-0097646, Alfred P. Sloan Fellowship, and Honda Initiation Award.

References

1. R. Balakrishnan, G. Fitzmaurice, G. Kurtenbach, and K. Singh. Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. In *Proceedings of the 1999 ACM Symposium on Interactive 3D Graphics*, pages 111–118, 1999.
2. D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, volume 26, pages 303–308, July 1992.
3. D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 43–54, July 1998.
4. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, Sept. 1978.
5. G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. In *Computer Graphics (SIGGRAPH 91 Proceedings)*, pages 257–266, July 1991.
6. F. Dachille IX, H. Qin, A. Kaufman, and J. El-Sana. Haptic sculpting of dynamic surfaces. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 103–110, 1999.
7. T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 85–94, July 1998.
8. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Computer Graphics (SIGGRAPH 94 Proceedings)*, pages 295–302, July 1994.
9. D. L. James and D. K. Pai. ARTDEFO: Accurate Real Time Deformable Objects. In *Computer Graphics (SIGGRAPH 99 Proceedings)*, pages 65–72, Aug. 1999.
10. R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, pages 181–188, August 1996.

11. C. Mandal, H. Qin, and B. C. Vemuri. A novel FEM-based dynamic framework for subdivision surfaces. In *Proceedings of Solid Modeling '99*, pages 191–202, 1999.
12. K. T. McDonnell and H. Qin. Dynamic sculpting and animation of free-form subdivision solids. *The Visual Computer*, 18(2):81–96, 2002.
13. K. T. McDonnell, H. Qin, and R. A. Wlodarczyk. Virtual Clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*, pages 179–190, March 2001.
14. D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, pages 309–312, July 1992.
15. T. Miller and R. C. Zeleznik. The design of 3D haptic widgets. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 97–102, 1999.
16. M. J. Muuss and L. A. Butler. Combinatorial solid geometry, boundary representations, and n-manifold geometry. In D. F. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics: Visualization and Modeling*, pages 185–223. Springer-Verlag, 1991.
17. A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (SIGGRAPH 89 Proceedings)*, pages 215–222, July 1989.
18. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition, 1992.
19. 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.
20. 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.
21. M. A. Srinivasan and C. Basdogan. Haptics in virtual environments: taxonomy, research status, and challenges. *Computers & Graphics*, 21(4):393–404, July 1997.
22. D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, Dec. 1988.
23. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, pages 269–278, August 1988.
24. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Computer Graphics (SIGGRAPH 87 Proceedings)*, pages 205–214, July 1987.
25. T. V. Thompson, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 167–176, 1997.
26. K. J. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, August 1986.
27. A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 159–168, Aug. 1988.