Haptics-Based Dynamic Implicit Solid Modeling

Jing Hua, *Student Member*, *IEEE*, and Hong Qin, *Member*, *IEEE*

Abstract—This paper systematically presents a novel, interactive solid modeling framework, Haptics-based Dynamic Implicit Solid Modeling, which is founded upon volumetric implicit functions and powerful physics-based modeling. In particular, we augment our modeling framework with a haptic mechanism in order to take advantage of additional realism associated with a 3D haptic interface. Our dynamic implicit solids are semi-algebraic sets of volumetric implicit functions and are governed by the principles of dynamics, hence responding to sculpting forces in a natural and predictable manner. In order to directly manipulate existing volumetric data sets as well as point clouds, we develop a hierarchical fitting algorithm to reconstruct and represent discrete data sets using our continuous implicit functions, which permit users to further design and edit those existing 3D models in real-time using a large variety of haptic and geometric toolkits, and visualize their interactive deformation at arbitrary resolution. The additional geometric and physical constraints afford more sophisticated control of the dynamic implicit solids. The versatility of our dynamic implicit modeling enables the user to easily modify both the geometry and the topology of modeled objects, while the inherent physical properties can offer an intuitive haptic interface for direct manipulation with force feedback.

Index Terms—Geometric modeling, physics-based modeling and sculpting, implicit functions, interaction techniques, haptic interface.

1 Introduction and Motivation

During the past two decades, solid geometry has been rapidly gaining popularity as an intuitive and natural paradigm for modeling, manipulating, and interacting with 3D objects in geometric design, graphics, visualization, virtual environments, etc. This is primarily because a solid model offers users a consistent and unambiguous shape representation of a physical entity [1]. In contrast to popular parametric geometry, implicit functions have a number of solid modeling advantages such as point classification, intersection computation, and unbounded geometry. Comparing with the discrete, voxel-based representations prevalently used in volumetric sculpting [2], [3], [4], [5], an implicit solid object is evaluated as level-sets of volumetric, continuous functions defined over a 3D working space, where arbitrary topology and complicated geometry are implicitly defined. Therefore, it is easy to handle topological change and collision detection. The continuous functions can be evaluated anywhere to produce a mesh at the desirable resolution. Gradients and higher-order derivatives are determined analytically. Multiresolution editing and direct rendering are also easy to achieve.

Despite their representation potential, existing modeling techniques associated with implicit functions have certain severe shortcomings which hinder the widespread penetration of implicit functions into geometric modeling, interactive graphics, and virtual environments. Current techniques [6], [7], [8] mainly depend on Boolean operations, blending, convolution, or interpolation of simple primitives to generate implicit objects. Users essentially must interact with solid geometry through tedious and laborious operations on a large number of control coefficients. When the goals are to interactively sculpt implicit objects in real-time, directly manipulate the implicit solid geometry in a free-form manner, and conduct kinematic and dynamic analysis of implicit objects with haptics, the current state-of-the-art in implicit modeling falls short in offering designers a unified framework and an array of flexible and powerful modeling and sculpting tools. In general, flexible and direct free-form modeling techniques for implicit solids remain underexplored.

In this paper, we integrate implicit solid geometry, powerful physics-based modeling, and a haptic mechanism into a unique, graphical modeling framework, *Haptics-based Dynamic Implicit Solid Modeling*, and systematically present a modeling environment that can provide users with a wide spectrum of haptic, geometric, and physical tools to facilitate the creation and editing of dynamic solids. These tools are both transparent to and independent of the underlying representations.

Our solid geometry combines the benefits of conventional implicit functions with those of popular spline-based parametric representations [9]. The volumetric implicit functions are defined by a 3D hierarchical and/or CSG-based organization of the underlying constituent scalar B-splines. Through knot insertion, our framework takes advantage of both uniform and nonuniform B-spline functions so that the knot distribution will control and influence the local shape. Furthermore, by using a new hierarchical fitting algorithm, our environment provides a flexible mechanism for users to import different representations, such as point clouds and volumetric data sets.

Physics-based modeling attempts to overcome some of the shortcomings of geometric modeling through the integration of material attributes and physical behaviors with geometric modeling techniques. For implicit surfaces or solids, however, since their geometry is generated indirectly from the zero-set of their function evaluation, we cannot directly associate physics with the underlying

• *The authors are with the Center for Visual Computing and the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400. E-mail: {jinghua, qin}@cs.sunysb.edu.*

Manuscript received 24 Mar. 2003; revised 14 Nov. 2003; accepted 17 Dec. 2003.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0017-0303.

zero-set because it is implicitly defined and the zero-set geometry has the time-varying nature. Note that this is perhaps the greatest technical challenge that prevents the integration of physics-based modeling and implicit representations. We develop a feasible and powerful technique to overcome this difficulty. Physical attributes and material attributes are assigned inside the working space. The dynamic behavior of our solid objects is controlled by a novel mass-spring system and by differential equations of Lagrangian mechanics. Our free-form solids respond dynamically to applied forces in an extremely intuitive and natural fashion. Therefore, instead of manually modifying the coefficients associated with the volumetric implicit function as exhibited in [7], our sculpting tools facilitate direct editing of implicit functions' scalar values. Geometric parameters of the volumetric implicit functions can be hidden from users through the use of natural, force-based interfaces that facilitate directly manipulated free-form deformation of implicit solid objects. Comparing with [10], which employed haptic toolkits to explore dynamic subdivision solids, no special efforts need to be taken to change the topology of modeled objects in our system. However, in [10], the topological modification cannot be directly performed with haptic tools during the physical simulation. Our system synchronizes the geometric and physical representations of modeled objects. The inherent control coefficients of the implicit functions dictate the shape geometry, while the physical attributes support direct manipulation and dynamic behavior simulation. Our unified formulation permits users to deform a solid object quickly and easily in a physically plausible fashion. More importantly, the dynamic modeling method makes it much easier to define objects with inhomogeneous materials.

Haptics provides additional sensory cues to designers which allow designers to gain a richer understanding of the 3D nature of virtual solids. With haptics, users can have a hand-based mechanism for intuitive interactions. In our environment, since both haptics and dynamic models depend on real-world physical laws to govern the interaction of dynamic objects and their realistic simulation, a haptic interface is very valuable and intuitive for users to fully interact with our models. We develop a haptic interface and provide a suite of haptic sculpting tools to further enhance physics-based modeling techniques. Haptic feedback can be computed directly from geometric and material properties. With a standard haptic device, our approach permits users to interactively sculpt virtual materials and feel the physically realistic presence with force feedback throughout the design process. Fig. 1 shows the haptics-based user interface of our modeling environment, where the user is using a PHANToM device to manipulate a dynamic implicit solid.

This paper extends our previous work on haptic sculpting [9] and dynamic modeling [11] of implicit objects. The first paper [9] presented a sculpting mechanism on volumetric implicit models with several haptic tools. In later work [11], we derived dynamic implicit models with a simple, explicit numerical solver and a few simple sculpting tools. This paper broadens the accessibility and further expands the capabilities of the sculpting system with a more comprehensive and natural set of intuitive tools. It also derives a more sophisticated, implicit solver, provides a complete mathematical derivation of our physical and geometric models, and describes in greater detail the functionality and implementation of our haptics-based modeling environment. We generalize the existing virtual sculpting tools to handle real-world objects and develop new capabilities for haptic interaction. Our modeling methodology and the new environment aim to incorporate physics in general and elasticity in particular into implicit functions and to advance the state of the knowledge in the effective integration of implicit functions, physics-based modeling, and haptic sculpting.

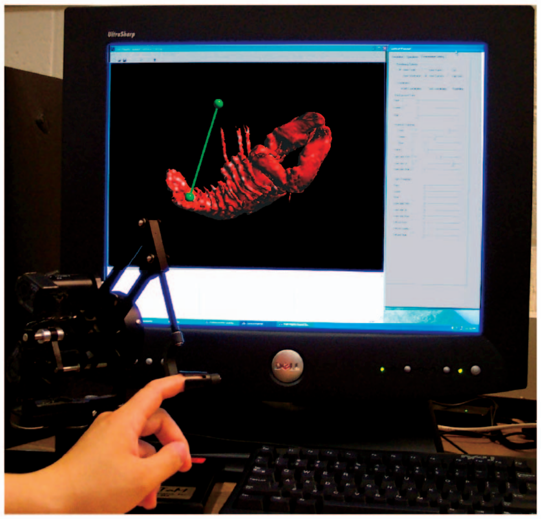


Fig. 1. Haptics-based user interface. The user is performing a bending operation on the red lobster, reconstructed from a CT-scanned data set.

2 Related Work

2.1 Implicit Modeling

Blinn [12] demonstrated that implicit functions are well-suited for both scientific visualization and modeling tasks in graphics. In order to interactively create implicit surfaces and gain more control over them, Bloomenthal [6] and Bloomenthal and Wyvill [13] used skeleton methods to construct implicit surfaces. Blobby models [12], also known as soft objects [14], [15], are another popular technique for designing implicit surfaces. Convolution surfaces [16], [17] represent another method for designing implicit surfaces through primitives. Hart et al. [18] proposed a method of finding the critical points of implicit surfaces. Witkin and Heckbert [19] used a physically-based particle approach to sample and control implicit surfaces. Turk and O'Brien [8] introduced new techniques for interpolating implicit surfaces. Implicit functions can also be used to represent a volume. Raviv and Elber [7] presented an interactive sculpting paradigm that employed a set of uniform trivariate B-spline functions as the underlying representation. Martin and Cohen [20] presented a trivariate spline-based mathematical framework to represent and extract volumetric attributes. Schmitt et al. [21] presented an approach for constructive modeling of FRep solids.

In other related work, the representation of sculpted volumetric objects is primarily through discretization (e.g., voxels). Galyean and Hughes [2] first introduced the concept of volume sculpting and developed a system with simple tools. Wang and Kaufman [3] presented a similar sculpting system with sculpting tools for carving and sawing. These sculpting systems are dependent on simple, voxel-based operations. Only C^0 continuity can be achieved. In order to avoid the spatial aliasing, the sculpted objects and sculpting tools need to undergo an appropriate filtering operation. Based on a similar idea, Ferley et al. presented a rapid shape-prototyping system [22]. Perry and Frisken [5] employed adaptively sampled distance fields (ADFs) as volumetric shape representations for creating digital characters. Most recently, Museth et al. [23] presented a level-set framework for interactively editing implicit surfaces.

2.2 Physics-Based Geometric Design

Physics-based models produce smooth, natural motions that are intuitive to control. Free-form deformable models were first introduced to computer graphics by Terzopoulos et al. [24] and further improved by Pentland and Williams [25] and Metaxas and Terzopoulos [26]. Cani and Desbrun [27] employed deformable implicit models for animating soft objects. Despite the popular use of physics-based models in animation, simulation, and graphics [28], [29], [30], [31], less effort has been devoted to free-form dynamic manipulation of manufactured objects, which is especially useful for providing an intuitive interface for geometric design. Szeliski and Tonnesen [32] introduced oriented particle systems, which can be used to model flexible surfaces. Qin and Terzopoulos [33] introduced D-NURBS surfaces, an extension to traditional NURBS that permits more natural control of the surface geometry. Mandal and Qin [34] further generalized this model to any subdivision scheme. Later, McDonnell et al. [10] extended the dynamic subdivision techniques to solids.

2.3 Haptic Interaction

A good introduction to haptic rendering can be found in [35]. Salisbury and his colleagues [36], [37] developed the PHANToM haptic interface, which has resulted in many haptic rendering algorithms. Morgenbesser and Srinivasan [38] pioneered the concept of force shading. Salisbury and Tarr [39] presented the research work for haptic rendering of simple implicit surfaces. Kim et al. [40] presented a rather different implicit-based haptic rendering technique. Avila and Sobierajski [41] used the PHANToM in a haptic scientific visualization process. Thompson et al. [42] derived efficient intersection techniques that permit direct haptic rendering of NURBS surfaces.

Despite the widespread application of haptics in visual computing areas, haptics-based interaction was mainly applied to touching compliant objects (i.e., haptic rendering), whereas haptic modeling allows designers to directly manipulate objects with force feedback for the purpose of modeling or deforming objects. Dachille et al. [43] developed a haptic interface that permits direct manipulation of dynamic surfaces. Balakrishnan et al. [44] developed *ShapeTape*, a curve and surface manipulation technique that can sense user-steered bending and twisting motions of the rubber tape. The *FreeForm* modeling system presented by SensAble Technologies enables users to easily create products with touch. However, this is purely a haptic enhancement for a traditional system. Haptic feedback is not based on the real dynamics of modeled objects. We integrate the principle of haptic modeling with the direct manipulation of dynamic implicit solids and employ force-based, haptic tools to directly work on scalar fields.

3 Volumetric Implicit Functions

In our modeling environment, the sculpted object is evaluated as level-sets of volumetric implicit functions defined over a 3D working space. Throughout this paper, we utilize trivariate scalar B-spline functions as the underlying shape primitives to build volumetric implicit functions for object representation [9]. The use of scalar B-spline functions is strongly inspired by their attractive properties, including simplicity, generality, local control, etc. These properties make them appropriate for our haptics-based modeling system, which requires fast function evaluation and relatively high smoothness of scalar fields.

3.1 Tensor-Product Scalar B-Splines

The generic B-spline functions are of the following form:

$$s(u,v,w) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} p_{ijk} B_{i,r}(u) C_{j,s}(v) D_{k,t}(w), \qquad (1)$$

where $s(u,v,w)$ represents the scalar value (also called the density value in this paper) at position (u,v,w) in a parametric domain. In (1), p_{ijk} are the scalar control coefficients, $B_{i,r}(u)$, $C_{j,s}(v)$, and $D_{k,t}(w)$ are the basis functions corresponding to p_{ijk}, evaluated at (u,v,w).

3.2 Building Volumetric Implicit Functions

In general, surfaces expressed by an implicit form can be formulated as: $\{(x,y,z) | f(x,y,z) = c\}$. The function f is called the implicit function, which defines the scalar field (or called the density field in this paper). Collecting all the level-sets whose return values are greater (or smaller) than a given threshold, we can define an implicit solid:

$$\begin{cases} w = f(x,y,z) \\ w > w_0. \end{cases} \qquad (2)$$

In our work, we shall collect different B-spline patches defined over the 3D working space to form a volumetric implicit function that can be collectively used to represent objects of complicated geometry and arbitrary topology. Note that, significantly different from frequently used parametric B-splines, scalar B-spline functions formulate the density value distribution in a 3D space where implicit solids are uniquely defined as semi-algebraic point sets. In our environment, we enhance the scalar B-spline representation power by incorporating the modeling advantages from hierarchical splines, generalized CSG-based Boolean operations, and nonuniform knot insertion.

Consider N B-spline patches in the sculpting space, which are located at any location and with any orientation. In general, these patches may be formulated by different numbers of control coefficients in order to achieve the goal of multiresolution analysis and LOD control. Then, the density value at the location (x, y, z) can be computed as

$$f(x,y,z) = \sum_{i=1}^{N} s_i(\mathbf{T}_i(x,y,z)), \quad (3)$$

where $\mathbf{T}_i$ is an affine transformation from the Euclidian space to the parametric domain of patch s_i. In essence, (3) is a hierarchical organization of the N patches. Without loss of generality, we make use of cubic B-splines with nonperiodic knot vectors. In order to make the boundaries of different trivariate patches achieve C^1 continuity, the first and last four layers of control coefficients along three principal directions of the parametric domain should be set to zero.

Furthermore, our system provides CSG-based operations on any user-defined trivariate patch in order to facilitate the rapid construction of complicated models satisfying many feature-oriented requirements. Therefore, complicated geometry is available in our system through the use of

$$f(x,y,z) = \mathbf{\Omega}_{i=1}^{N} s_i(\mathbf{T}_i(x,y,z)), \quad (4)$$

where $\mathbf{\Omega}$ is a Boolean operation such as Union, Intersection, or Difference. In our system, the Boolean operation information will be stored in a tree structure in order to speed up the data query.

Meanwhile, our system allows users to specify a nonuniform knot vector during the initialization phase of the object design session. In addition, users can insert more knots into the current knot vector during the sculpting process. When new knots are inserted, the system will generate corresponding control coefficients and the sculpted object will be reevaluated upon the refined knot vector. Thus, the underlying model is essentially a nonuniform scalar B-spline.

Through the use of different combinations of the aforementioned three techniques, our environment can offer users a large array of modeling operations and enhance the already-powerful shape variation of scalar B-splines with the additional flexibility in a hierarchical fashion.

4 Hierarchical Implicit Function Fitting

In order to allow users to edit existing solid objects, we wish to transform discrete solid representations of modeled objects to the continuous representation in our environment. We shall find a volumetric implicit function f, which implicitly defines any user-specified solid. The volumetric implicit function offers a compact functional description for a set of discrete input data. In previous work, Muraki [45] proposed the algorithm to reconstruct range data using the blobby model. Turk and O'Brien [46] and Carr et al. [47] used radial basis functions to reconstruct and represent point clouds. Our reconstruction algorithm can handle point clouds as well as volumetric data sets.

Let us first discuss the fitting and reconstruction of a point cloud. This issue is essentially an interpolation problem:

Find f such that

$$\begin{aligned} &f(x_i, y_i, z_i) = 0, \quad i = 1, \cdots, n \quad &&\text{(iso-surface points)},\\ &f(x_i, y_i, z_i) = d_i, \quad i = n+1, \cdots, N \quad &&\text{(off-surface points)}, \end{aligned}$$

where $\{(x_i, y_i, z_i) | i = 1, \cdots, n\}$ are points lying on the surface and $\{(x_i, y_i, z_i) | i = n+1, \cdots, N\}$ are points lying off the surface since the density values $d_i \neq 0$.

The iso-surface points are always given by the point cloud. However, there is still a problem of how to generate the off-surface points and their corresponding d_i. There has been a lot of research work on this topic [46], [47]. One viable solution is a signed-distance field, where the d_i is the distance to the closest iso-surface point. In our system, points outside the solid are assigned negative values, while points inside are assigned positive values. It is not necessary to generate the entire distance field. In our experiments, it is sufficient to produce two off-surface points associated with each iso-surface point, one outside and the other inside. We employ the tagging algorithm recently proposed by Zhao et al. [48] and slightly modify it to compute the required signed distance field, then we make use of a least-square fitting to obtain the volumetric implicit function, whose zero level-set fits the given point cloud.

Generally, using a single B-spline to fit a large data set is impractical since the required number of control coefficients will be too large to handle and the fitting error will be unacceptable. Our system utilizes the volumetric implicit function, (3) or (4), to obtain a hierarchical implicit B-spline representation for the object. An octree-based subdivision scheme is employed to subdivide the working space containing the solid object. Our recursively hierarchical fitting algorithm is illustrated as follows:

1. Create an octree for the entire working space, which contains the fitted object, and subdivide the root node to eight child nodes according to the octree subdivision.
2. Fit a single scalar B-spline to the region of each child node using the least-square technique.
3. Evaluate the mean square error (*MSE*) at node i,

$$\varepsilon_i = \frac{1}{N_i} \sum_{j=1}^{N_i} (d_j - f(x_j))^2,$$

   where N_i denotes the number of sampling points inside the region of node i and d_j is the density value at the sampling point x_j.
4. If ε_i is less than the user-specified error bound ε, then mark the node as a leaf node.
5. Else subdivide the node i to eight child nodes, go to 2.

This algorithm will not stop until all the child nodes are marked as leaf nodes. Then, the discrete point cloud is converted to a continuous spline-based volumetric implicit function which can be evaluated at arbitrary sampling resolution and rendered with the Marching Cubes algorithm. Fig. 2 shows a hierarchical fitting structure. The red color means the *MSE* at that region is greater than the error bound and the region needs to be further subdivided and fitted. Since the number of sampling points is finite, the fitting algorithm will converge as long as enough leaf nodes are generated. The number of leaf nodes needed for fitting at the desired accuracy depends on the user-specified error bound and the data sets.

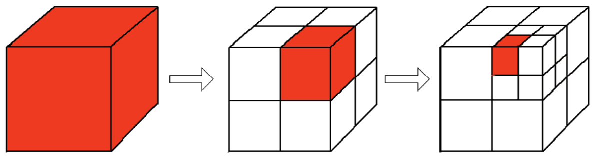


Fig. 2. An octree-based subdivision scheme and a hierarchical fitting structure, where the red color denotes the node that needs to be subdivided.

Our fitting algorithm can handle several types of point clouds, including scattered data sets, damaged data sets with missing information, cross-sectional data sets, and noisy data sets. Fig. 3 shows an example of fitting a nonuniformly distributed point cloud. Besides point clouds, our system can also transform volumetric data sets to the spline-based volumetric implicit functions. In this case, we treat the intensity value at a grid point (i, j, k) as d_{ijk}. Then, the interpolation problem is essentially the same as the one documented above. The fitting algorithm is also the same as the one used in the fitting process for point clouds. Fig. 4 shows a volumetric object and its fitted implicit solid.

5 Dynamic Implicit Solids

Our dynamic implicit solids marry the level-set geometry with physical attributes and other relevant material quantities, offering extra flexibility and advantages in modeling. In this section, we will discuss how to integrate elasticity with implicit models, how to apply force to manipulate implicit models, and how to simulate the dynamic behavior of implicit models.

5.1 Integration of Elasticity with Implicit Solids

In order to incorporate physics into implicit solids, the sculpted object of a B-spline-based implicit function is discretized into a voxel raster. Every voxel contains a density value, sampled at a grid point. The volumetric implicit function described in Section 3 is employed to assign the density value to the sampling points to indicate the material attribute at that location. The function will be used to formulate the density distribution over the 3D working space and represent the sculpted object using a given level-set. Fig. 5 shows a simple sculpted object and its corresponding voxelmap. Note that, in our system, the characteristic function is not a binary function, rather it is a continuous function.

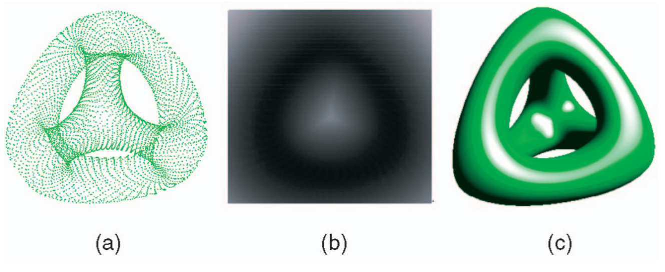


Fig. 3. (a) A point-sampled smooth tetrahedron with four holes. (b) Cross-sectional view of the generated distance field. (c) The reconstructed object represented by a volumetric implicit function.

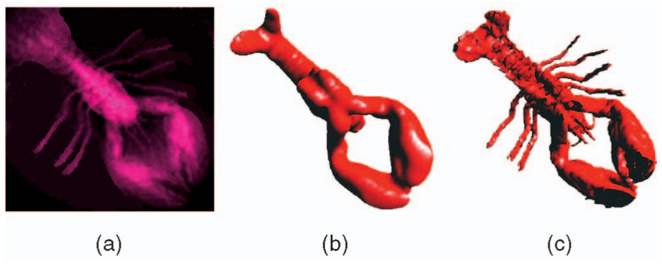


Fig. 4. (a) Volume rendering of the original density field. (b) Fitting with fewer octree layers and control coefficients. (c) The finally reconstructed volume object represented by a volumetric implicit function.

In the discretized working space, we can discretize (3) or (4) and make use of

$$\mathbf{d} = \mathbf{A}\mathbf{p} \tag{5}$$

to formulate the density values associated with the sampling points in a patch, where $\mathbf{A}$ is a sparse basis function matrix that contains weights computed from our spline-based volumetric implicit functions and $\mathbf{p}$ is a vector of the scalar control coefficients. The discretized density field is represented by $\mathbf{d}$.

The discretized density field is then assigned other material quantities such as mass, damping, and stiffness distribution. These values are defined as functions $\mu(u, v, w)$, $\gamma(u, v, w)$, and $\rho(u, v, w)$, respectively, which often can be considered to be constant. However, these material distributions are allowed to be modified by users interactively and directly (see Section 6.3). The discretized field can then be modeled as a collection of mass-points connected by a network of springs across nearest neighbors. Mass-points are located at sampled grid points. Besides the aforementioned quantities, a mass-point $m_{i,j,k}$ has two other attributes, the geometric position $x_{i,j,k}$ and the density value $d_{i,j,k}$ at the position. Here, we use a mass-spring model because of its simplicity and the critical need of real-time haptic volume sculpting. Fig. 6a shows the mass-spring network in the vicinity of a point.

We refer to these springs as "density springs." This is because this new type of spring is unconventional in the sense that they are fundamentally different from ordinary springs commonly used in parametric deformable models, where springs are employed to connect pairs of geometric vertices and modify vertex geometry upon deformation. In contrast, our special springs employed in implicit functions do not intend to change the geometric position $x_{i,j,k}$ of the mass-point $m_{i,j,k}$ at all. Instead, they only permit the

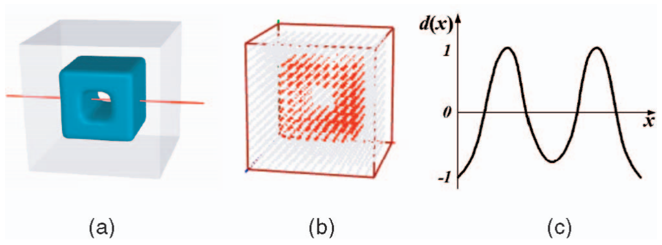
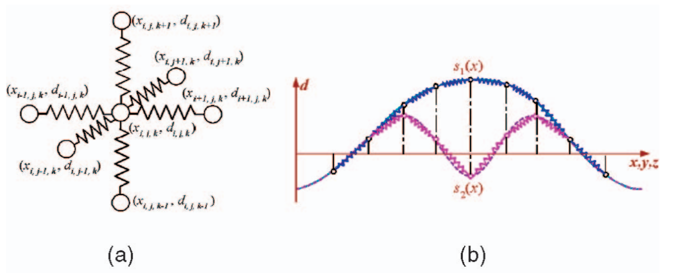


Fig. 5. (a) A simple sculpted object. (b) A corresponding voxelmap. (c) The density distribution along the red line.



$(x_{i,j,k+1}, d_{i,j,k+1})$
$(x_{i,j+1,k}, d_{i,j+1,k})$
$(x_{i-1,j,k}, d_{i-1,j,k})$
$(x_{i+1,j,k}, d_{i+1,j,k})$
$(x_{i,j,k}, d_{i,j,k})$
$(x_{i,j-1,k}, d_{i,j-1,k})$
$(x_{i,j,k-1}, d_{i,j,k-1})$
(a)
d
$s_1(x)$
$s_2(x)$
x,y,z
(b)

Fig. 6. (a) The mass-spring network in the vicinity of a point $P(x_{i,j,k}, d_{i,j,k})$, where $x_{i,j,k}$ represents the position of a mass-point $m_{i,j,k}$ in 3D, $d_{i,j,k}$ is the density at that position. (b) The density field changes from $s_1(x)$ to $s_2(x)$ due to the movement of density springs. The geometric positions of those density springs do not change.

magnitude change of the density $d_{i,j,k}$ located in the mass-point $m_{i,j,k}$. Essentially, this new type of spring will only attract/repel density values of neighbors. When users manipulate the implicit solids, the density values are changed by the mass-spring system. Fig. 6b shows the density field changes from $s_1(x)$ to $s_2(x)$ due to the movement of the density springs, where mass-points move only along the density axis (i.e., the geometric positions of the mass-points do not change). Consequently, this results in the deformable behavior of the object's shape modeled by the level-set of the spline-based volumetric implicit function. Therefore, elasticity has been introduced to our volumetric implicit objects and our implicit solids now become deformable models, which we name *dynamic implicit solids*. Note that, even though the geometry and topology of the network of mass-points do not vary over time, this approach has the capability to model arbitrary topology and complicated geometry since the resulting shape is generated by extracting an isosurface from the density field rather than the geometric position of mass-points. This novel approach affords a systematic mechanism for users to directly manipulate arbitrary implicit functions and their different level-sets without the need to modify their associated control coefficients manually.

The motion equation of the density field associated with all mass-points is formulated as a discrete simulation of Lagrangian dynamics:

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{D}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f_d}, \tag{6}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{D}$ is a damping matrix, $\mathbf{K}$ is a stiffness matrix, and the force at every mass-point in the working space is the summation of all possible external forces: $\mathbf{f_d} = \sum \mathbf{f}_{ext}$. The internal forces are generated by the connecting springs, where each spring has force $f = k(l - l_0)$ according to Hooke's law. l_0 denotes the rest length and l denotes the current length of the spring. They are calculated as the 4D (i.e., 3D position plus 1D density) distance between two mass-points that the spring connects. In our system, the geometric positions of mass-points do not move and only density values change. Hence, only the component of forces along the density axis will be taken into account in the dynamic simulation. The rest length of each spring is determined upon initialization; however, it is free to vary if plastic deformations or other nonlinear phenomena are more desirable.

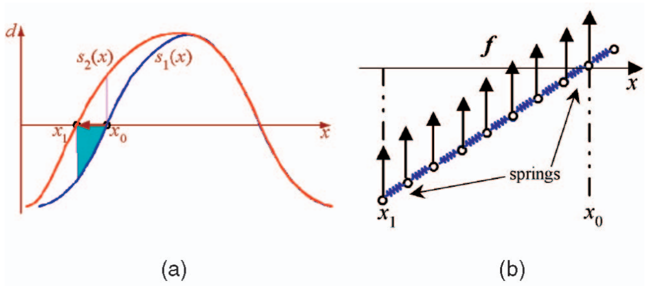


Fig. 7. (a) Iso-surface changing from x_0 to x_1 via applied force f, which is proportional to the gray area. (b) Close-up view of the mass-spring network of $s_1(x)$, where f is applied on every mass-point between x_0 and x_1.

5.2 Response to Applied Forces

In principle, a deformable model is defined by a given correspondence between applied forces and deformation. In order to allow direct deformation of the implicit solids in a force-based manner, we must address the important issue of force mapping, which defines how dynamic models respond to applied forces. Note that the generated forces will be input to the dynamic system as external forces and will also be sent to a haptic device in our system. Therefore, any force mapping algorithm must be meaningful and suitable for both the dynamic simulation and the haptic interaction. To illustrate the concept clearly, we shall use a one-dimensional implicit function to describe how to implement the force mapping mechanism in our system. More complicated situations in 3D can be trivially generalized.

For an arbitrary one-dimensional implicit function, the zero-set is simply a set of points. As shown in Fig. 7, consider that a user wants to move one point of the zero-set, x_0, to x_1. Our system then automatically generates a series of forces f applied on every mass-point between x_1 and x_0. As a result, these forces will increase the density value from $s_1(x)$ to $s_2(x)$ correspondingly at all the affected locations within the interval. Eventually, the density value at x_1 will be zero and the density values between x_0 and x_1 will be greater than zero. Hence, the iso-surface evolves from x_0 to x_1, undergoing real-time deformation controlled by the numerical integration of Lagrangian dynamics. To further convey this idea, we can imagine that the above process is equivalent to the lifting of the "density height" for every affected mass-point via applied forces.

In our force mapping mechanism, the applied sculpting force is calculated directly from the continuous representation by performing integration from the starting point to the ending point along the direction dictated by the force vector. In this one-dimensional example, the force vector is simply a straight line-segment, therefore,

$$f = -\int_{x_0}^{x_1} s_1(x)dx.$$

Because $\int_{x_0}^{x_1} s_1(x)dx$ is less than zero in this example, the minus sign outside the integral operator makes the force positive, matching the case shown in Fig. 7. In our system, we define the following conventions to enforce consistency. The positive force is to increase the density value and the

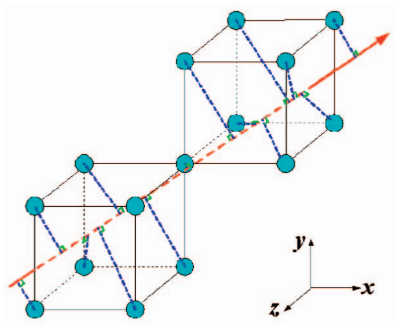


Fig. 8. The force is distributed into mass-points of cells according to the shortest distance from the mass-points to the force vector.

negative force is to decrease the density value. Note that f is decreasing over time as x_0 moves toward x_1.

The force mapping mechanism of our system is very general and can deal with iso-surface enlarging (as shown above) as well as iso-surface shrinking with the same formula for force calculation. In Fig. 7, suppose that the user intends to move x_1 to x_0 instead. Then, the force calculation will be $f = -\int_{x_1}^{x_0} s_2(x)dx$. In this case, f becomes negative, which will decrease the density values between x_0 to x_1 from $s_2(x)$ to $s_1(x)$ correspondingly.

Now, we shall generalize our force mapping technique to a 3D domain,

$$f = -\alpha \int_C s(u, v, w)dc, \tag{7}$$

where C is any force vector which is used as the integration path and $s(u, v, w)$ is the density distribution function in the 3D working space. When f is input as external forces into (6), the density field will be changed and this will result in the deformation of the dynamic implicit solid. The proportional factor α is often set to 1.0. However, if the user is working on a "heavy" or "high-stiffness" model, α can be increased to generate larger forces and quickly make deformations.

In our system, the force vectors associated with force-based, haptic tools can be along arbitrary directions, even curved ones. Therefore, they may not pass the mass-points of the density springs. We transform the sculpting force into the eight mass-points of the cell through which the vector C passes by filtering the shortest distance from the mass-point to the force vector, as illustrated in Fig. 8. The transformed forces at the mass-points that are closer to the force vector are larger.

5.3 Numerical Integration and Simulation

To simulate the behavior of dynamic implicit solids in the haptics-based environment, it is vital to design less costly yet stable time integration methods that take modest time steps. We have implemented both explicit and implicit numerical solvers for the time integration. 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 of more concern. In the interest of space, we only describe the implicit solver. The explicit solver can be derived straightforwardly.

Since all the discretized points and springs are constrained by the spline-based volumetric implicit function, we shall formulate the motion equation of physical behavior for all the control coefficients that define the scalar B-splines. We augment the discrete Lagrangian equation of motion with geometric and topological quantities related to the volumetric implicit function. By multiplying each side with $\mathbf{A}^\top$ and substituting $\mathbf{d}$ with $\mathbf{Ap}$ (see (5)), we obtain:

$$\mathbf{A}^\top\mathbf{MA}\ddot{\mathbf{p}} + \mathbf{A}^\top\mathbf{DA}\dot{\mathbf{p}} + \mathbf{A}^\top\mathbf{KAp} = \mathbf{A}^\top\mathbf{f_d}. \tag{8}$$

The implicit solver is implemented based on backward Euler integration. Discrete derivatives are computed using backward differences:

$$\ddot{\mathbf{p}}_i = \frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{\Delta t^2}, \tag{9}$$

$$\dot{\mathbf{p}}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2\Delta t}. \tag{10}$$

We derive the time integration formula as follows: To simplify notation, let $\widetilde{\mathbf{M}} = \mathbf{A}^\top\mathbf{MA}$, $\widetilde{\mathbf{D}} = \mathbf{A}^\top\mathbf{DA}$, $\widetilde{\mathbf{K}} = \mathbf{A}^\top\mathbf{KA}$. Furthermore, we represent the forces acting on $\mathbf{p}$, $\mathbf{f_p}$, using $\mathbf{f_d}$: $\mathbf{f_p} = \mathbf{A}^\top\mathbf{f_d}$, then (8) becomes:

$$\widetilde{\mathbf{M}}\ddot{\mathbf{p}} + \widetilde{\mathbf{D}}\dot{\mathbf{p}} + \widetilde{\mathbf{K}}\mathbf{p} = \mathbf{f_p}. \tag{11}$$

Substituting (9) and (10) into (11) yields:

$$\widetilde{\mathbf{M}}\left(\frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{\Delta t^2}\right) + \widetilde{\mathbf{D}}\left(\frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2\Delta t}\right) + \widetilde{\mathbf{K}}\mathbf{p}_i = \mathbf{f}_{\mathbf{p}_i}.$$

After multiplying both sides of the above equation by $2\Delta t^2$ and with some additional algebraic manipulation, we arrive at the hybrid equation of motion:

$$\begin{aligned}\left(2\widetilde{\mathbf{M}} + \Delta t\widetilde{\mathbf{D}} + 2\Delta t^2\widetilde{\mathbf{K}}\right)\mathbf{p}_{i+1} &= 2\Delta t^2\mathbf{f_p} + 4\widetilde{\mathbf{M}}\mathbf{p}_i \\ &\quad - \left(2\widetilde{\mathbf{M}} - \Delta t\widetilde{\mathbf{D}}\right)\mathbf{p}_{i-1},\end{aligned} \tag{12}$$

It is straightforward to employ the conjugate gradient method 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 8. 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.

The updated control coefficients $\mathbf{p}_{i+1}$ are further used to update the discretized field defined by $\mathbf{d}_{i+1} = \mathbf{Ap}_{i+1}$. Note that the simulation does not change the geometric position of mass-points. It only updates the density value of every mass-point. The volumetric implicit function can be evaluated anywhere at arbitrary resolution once $\mathbf{p}$ is known. After generating the new density field (or new implicit representation), the new applied forces are calculated and will be applied in subsequent simulation steps. Fig. 16 illustrates the simulation loop in more detail. This new dynamic approach can continuously evolve the implicit functions and, therefore, permit users to directly

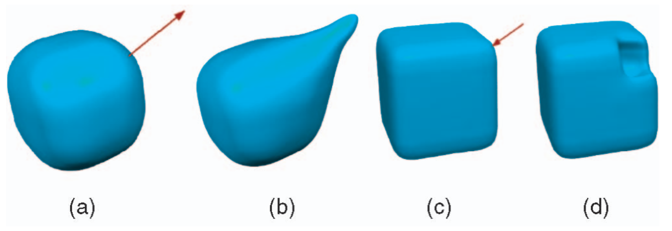


Fig. 9. Deformation with point-based haptic tools, where (a) and (c) show the original objects and the arrows denote the directions of the applied forces. (b) and (d) show deformed shapes.

work on both the level-set geometry and the enclosed material distribution with continuous visual feedback.

6 Interaction Techniques

Our modeling environment provides three primary types of sculpting tools: 1) haptic tools, 2) geometric tools, and 3) constraint-based tools.

6.1 Haptic Modeling

In our system, the simple force-based, haptic tool allows the user to grab the nearest mass-point in the solid. In addition, our system allows users to grab a subset of the mass-points in a nearby region simultaneously. The force is then distributed among nearby points using a user-defined, filtering function $\beta(x, y, z)$, which can be constant, Gaussian, spherical, cylindrical, conical, or any other distribution.

For sculpting with a ***point-based haptic tool***, we can use the parametric form $\{(u(t), v(t), w(t)) | t \in [t_0, t_1]\}$ to represent a force vector C in (7), then we have

$$f = -\alpha \int_{t_0}^{t_1} s(u(t), v(t), w(t)) \sqrt{\dot{u}^2(t) + \dot{v}^2(t) + \dot{w}^2(t)} dt.$$

If we assume the force vector to be a straight line, then C can be formulated as follows:

$$\begin{cases} u(t) = u_0 + (u_1 - u_0)t \\ v(t) = v_0 + (v_1 - v_0)t \qquad t \in [0, 1], \\ w(t) = w_0 + (w_1 - w_0)t \end{cases}$$

where (u_0, v_0, w_0) is the starting point of the force vector C and (u_1, v_1, w_1) is the ending point. Then,

$$f = -\alpha \cdot l \int_0^1 s(u(t), v(t), w(t)) dt,$$

where $l = \sqrt{(u_1 - u_0)^2 + (v_1 - v_0)^2 + (w_1 - w_0)^2}$. Fig. 9 shows two examples for point-based deformation.

In a more general case, if C is a spatial curve instead, a general ***curve-based haptic tool*** can be provided in our system without any additional difficulty. When the user sculpts an object with a curve-based tool, the integral of forces is along the curve force vector. The generated forces are then applied on the object, more accurately, on all the mass-points on or near the curve vector. Fig. 10a shows a curve force-based deformation. Since we use the finger-based haptic device, it cannot support full three-dimensional force feedback. The direction of the force sent back to the user has to be from the point that the user originally select (we call it the reference point) to the current haptic cursor point. For other more advanced ***area-based haptic tools***, users can predefine an arbitrary region and limit the force generation only inside the specified region. Our system discretizes the area into a set of sampled (straight and/or curved) vectors, then performs integration along every sampled vector, and results in a more sophisticated deformation in any user-specified region. Fig. 10b shows an area force-based deformation. In this example, the user first defines a region, "S." Then, the user applies forces perpendicular to the region simultaneously to make deformations. The force sent back to the user is the aggregation of all the applied forces. Note that we divide these tools into three categories, point-based, curve-based, and area-based ones, purely based on their functionality of deforming objects. From a computational point of view, the differences among the tools are the types of integration paths that are used. As for haptic feedback, users can only feel a single linear force whose magnitude is equal to the magnitude of the generated force for deforming objects, while the direction of the haptic force is only limited from the reference point to the current cursor point. Fig. 10c shows a force-based joining, where a point-based haptic tool is employed.

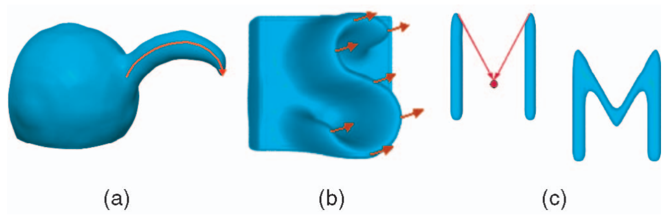


Fig. 10. (a) Free-form sculpting with a curve-based haptic tool. (b) Free-form sculpting with an area-based haptic tool. (c) Joining with a point-based haptic tool. The red arrows indicate the force vectors, where the force generations happen.

Using point-based, curve-based, and area-based haptic tools, a wide range of haptic sculpting operations can be performed, which includes drilling, extrusion, cutting, and joining. If we associate the force vector with certain constraints, then more interesting haptic tools can be created, such as a ***haptic chisel***, ***haptic squirting tool***, ***haptic sweeping tool***, and so on. Fig. 11a shows an operation performed with a haptic chisel, which controls the tool-penetration depth at the contacting points to be a small, constant value along the normal of the tool surface. Therefore, the sculpting forces are generated by integrating only along the array of specified, short force vectors sampled over the trajectories of the tool path (i.e., "H" in this example). The generated forces result in the chisel effect

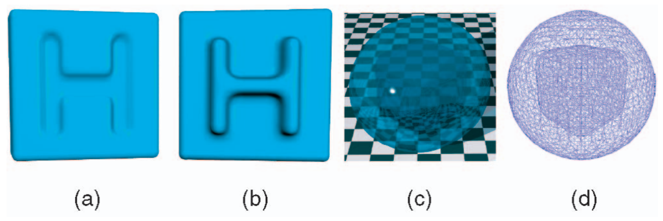


Fig. 11. (a) Chisel operation. (b) Squirting operation. (c), (d) Haptic tools allow users to sculpt solid interior without breaking the outer material.

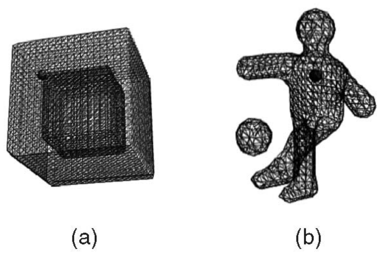


Fig. 12. (a) Haptic iso-surface rendering. (b) Haptics-based probing.

along the trajectories of tool path. Fig. 11b shows a squirting operation, where the generated sculpting forces are opposite to chisel forces along the same trajectories. The haptic sweeping tool allows users to create a sweeping shape easily. The user employs the haptic cursor to pick up a primitive solid object and freely sweep inside the working space. A sweeping shape can be generated along the freeform curve track. More importantly, our modeling environment provides users with tools that can perform more interesting tasks which are impossible to conduct in real-world sculpting. For example, users can sculpt the modeled object anywhere (not just adding/removing material over the solid boundary). Fig. 11c and Fig. 11d show an example that has undergone sculpting only in the solid interior while maintaining its surrounding, boundary material. In this example, the shape of the used haptic tool is a small cube. The user applies it inside the solid sphere to remove materials gradually. The computation of the feedback force is based on the technique presented in [9].

Our system also offers several other special, haptic tools for ***haptic iso-surface rendering*** and ***haptics-based probing***. Through haptic iso-surface rendering, besides feeling the boundary surface of a volumetric object, users can choose any iso-value from the allowable range of the volumetric implicit function and feel its shape. The haptic feedback is generated from the polygonal representation of the iso-surface using the Ghost APIs. Fig. 12a shows two different iso-surfaces with a wireframe display in order to make two surfaces visible at the same time. Users can feel different iso-surfaces of the sculpted object by moving the haptics cursor and navigating over those iso-surfaces. This tool allows users to examine the objects' boundary and their interior structure in a tactile manner. For example, designers can feel how smooth the iso-surface is at the current design level and can also look for any surface anomalies that might be hard to detect visually.

When performing haptics-based probing, users can feel the tiny, tactile difference of an object's stiffness while moving the haptics cursor inside the object (see Fig. 12b). When active, the probing tool exerts a force on the user's finger proportional to the local stiffnesses of the springs within a given radius of the tool. A spring's influence on the aggregate force decreases linearly with increased distance from the tool. The user feels a viscous-like force that smoothly increases and decreases as the 3D cursor moves into and out of regions of high and low stiffness, respectively.

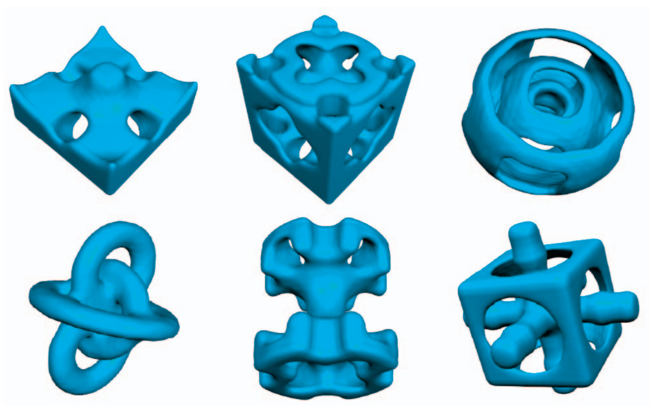


Fig. 13. Sculpted examples using geometric tools.

6.2 Geometric Tools

Geometric tools are represented by any 3D implicit function $c_0 = G(x, y, z)$. When users assign a sculpting tool to a new location, the density values inside the tool volume are modified. We construct a volumetric implicit function of B-splines by using a least-squares fitting [9]. After the new control coefficients are generated, the system uses a local Marching Cubes algorithm to render the modified part. Using the geometric tools, users can create objects of complicated geometry and arbitrary topology. The available geometric toolkits include ***sphere-based carving***, ***cylinder-based carving***, ***rectangle-based carving***, and ***torus-based carving tools***. More importantly, our modeling environment also allows designers to define their own tools, called ***self-defined carving tools***, using any primitive implicit functions. Geometric tools facilitate precise sculpting operations on volumetric models with interactive speed, and can be negated to add material to the sculpture. Fig. 13 shows a number of examples sculpted using our geometric toolkits.

6.3 Constraint-Based Tools

Our system can provide both physics-based constraints and geometric constraints at the same time. Enforcing both kinds of constraints offers additional intuitive control of a shape during the design process. Constraining geometric and physical properties of dynamic implicit solids can facilitate feature-centered design, which can significantly improve the system performance.

Our volumetric implicit function uses B-splines as underlying constituents. Therefore, local support and ***local sculpting*** can be easily accomplished. A designer can specify the region R in which he/she wishes the deformation to occur. Control coefficients and mass-points outside the specified region are not processed by the system and remain fixed. For the localized region R, $\mathbf{d}_R = \mathbf{A}'\mathbf{p}_R$, where $\mathbf{A}'$ is a small subset of the original basis function matrix.

The haptic device we are currently using requires a 1,000Hz refresh rate. This hardware constraint only permits the real-time simulation of thousands of mass-points. Therefore, local sculpting is much more accessible. Fig. 14a shows localized regions with colored semi-transparent boxes which limit physical operations within their bounding boxes.

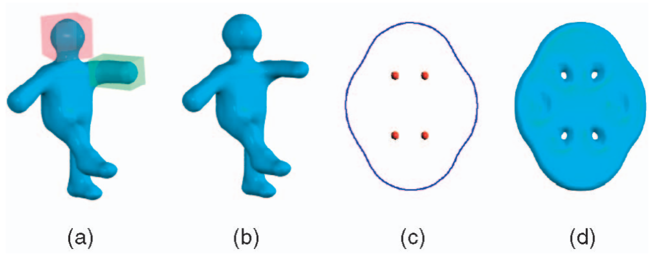


Fig. 14. (a), (b) Inflation and deflation operations performed inside the colored bounding boxes. (c), (d) Shape design using constraint springs. The red points denote off-surface points and the specified curves must be on the iso-surface.

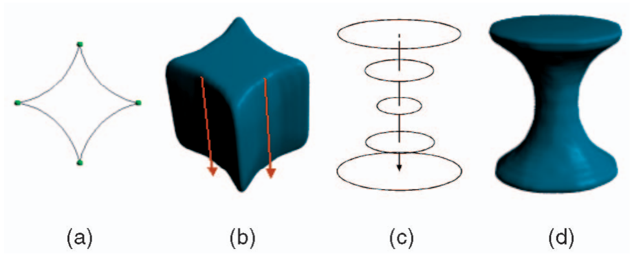


Fig. 15. A cross-sectional design using point constraints and curve constraints. The green points denote on-surface points and the specified curves must be on the iso-surface. The arrows denote the sculpting forces.

Through the use of ***physical property modification tools***, users can locally modify the mass distribution, spring stiffness, or any spring's rest length. Therefore, our virtual material can actually have different physical properties throughout the working space. In our sculpting system, we use a painting metaphor to describe the process of assigning or changing material properties. Using the haptic cursor, springs within a user-supplied radius of the tool are slowly "painted" with increasing or decreasing stiffness, based on which tool is active. The rate at which a spring's stiffness changes decreases linearly with increasing distance from the tool. This functionality allows the user to create smooth transitions from regions of low stiffness to regions of high stiffness. The stiffness distribution can be physically felt by using the aforementioned haptics-based probing tool. Stiffness modification can help users constrain certain parts of a sculpted object subject to soft constraints. Users can fix a region by increasing the stiffness of that part. This is similar to the popular penalty method [43]. Therefore, the applied deformation forces will have less effect on a high stiffness region. Similarly, the user can locally modify the masses or rest lengths through painting. Mass modification allows users to control how quickly one part of a sculpture can move in response to the external force. Regions of high mass density tend to move slowly while less massive parts respond quickly to a deformation force. Rest-length modification can make a specified region deflate or inflate. With the decrease of the rest length, all the springs inside the region generate contracting forces, which results in the effect of solid shrinking. To produce an inflation effect, the rest length is increased instead. Fig. 14b shows an inflation operation on the head and a deflation operation on the left arm of our clay puppet.

We provide a physical mechanism to enforce the linear constraints. The constraints are implemented as additional ***constraint springs***, which transform constraints to external forces and then add them to the modeled solids. This method treats constraints as soft constraints, but it offers better computation performance. Suppose that a user wants to constrain the density value of a mass-point at (x, y, z), an additional high stiffness spring is then attached between (x, y, z, d) and (x, y, z, d'), where d' is the desired density value. If we set $d' = 0$, then (x, y, z) is lying on the iso-surface. Otherwise, (x, y, z) is lying inside the iso-surface if $d' > 0$ or outside the iso-surface if $d' < 0$. Through the use of point constraints, the user can let the iso-surface interpolate a set of points. Curve constraints are implemented based on the same technique used in point constraints. A user can specify a curve using a parametric form or an implicit form. Alternatively, the user can interactively sketch it. Then, the curve is discretized to a set of points. Fig. 14c and Fig. 14d show a constraint-based design example using additional springs. In this design, after defining those springs, the user starts to use force tools to add material inside the curve boundaries. During the design process, the zero level set will not grow outside the curve boundaries.

We also provide a way to enforce ***geometric constraints*** as hard constraints. The linear constraints are expressed as follows:

$$\mathbf{R}_{gc}(\mathbf{d}) = \mathbf{L}\mathbf{d} + \mathbf{b} = 0, \tag{13}$$

where $\mathbf{d}$ is the generalized coordinate vector. Usually, (13) is an underdetermined linear system. Therefore, we can eliminate those constrained variables and express $\mathbf{d}$ with the other unconstrained variables $\mathbf{h}$,

$$\mathbf{d} = \mathbf{G}\mathbf{h} + \mathbf{h}_0. \tag{14}$$

Replacing $\mathbf{d}$ in (6) with (14), the matrices and vectors in (6) are reduced to a unconstrained set of generalized coordinates:

$$\mathbf{MG}\ddot{\mathbf{h}} + \mathbf{DG}\dot{\mathbf{h}} + \mathbf{KGh} = \mathbf{f_d} - \mathbf{K}\mathbf{h}_0. \tag{15}$$

Defining mass, damping, and stiffness matrices of the constrained dynamic system as: $\mathbf{M}_h = \mathbf{G}^\top\mathbf{MG}$, $\mathbf{D}_h = \mathbf{G}^\top\mathbf{DG}$, $\mathbf{K}_h = \mathbf{G}^\top\mathbf{KG}$, $\mathbf{c} = -\mathbf{G}^\top\mathbf{K}\mathbf{h}_0$, the discrete Lagrangian equations of the constrained dynamic system can be obtained from (15) as follows:

$$\mathbf{M}_h\ddot{\mathbf{h}} + \mathbf{D}_h\dot{\mathbf{h}} + \mathbf{K}_h\mathbf{h} = \mathbf{f}_h + \mathbf{c}. \tag{16}$$

In the constrained dynamic system, $\mathbf{h}$ has reduced size compared with $\mathbf{d}$. Hence, $\mathbf{h} = \mathbf{A}'\mathbf{p}'$, where $\mathbf{A}'$ and $\mathbf{p}'$ are subsets of the original matrices, $\mathbf{A}$ and $\mathbf{p}$. We can still use the numerical technique described in Section 5.3 to solve (16). This method treats those constraints as hard constraints. Fig. 15 shows a cross-sectional design using point and curve constraints.

7 System Implementation

Our environment is implemented on a PC with a 2.2 GHz CPU and 1 GB RAM. A PHANToM 1.0 3D Haptic input/output device from SensAble Technologies is employed to provide a natural and realistic force feedback. Our dynamic implicit modeling environment can easily handle

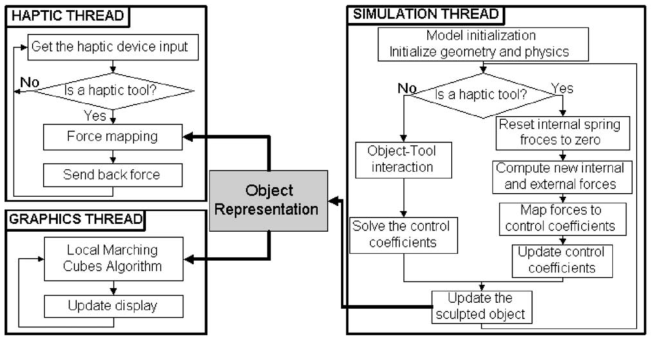

HAPTIC THREAD
Get the haptic device input
Is a haptic tool?
No
Yes
Force mapping
Send back force
GRAPHICS THREAD
Local Marching Cubes Algorithm
Update display
Object Representation
SIMULATION THREAD
Model initialization Initialize geometry and physics
Is a haptic tool?
No
Yes
Object-Tool interaction
Solve the control coefficients
Reset internal spring froces to zero
Compute new internal and external forces
Map forces to control coefficients
Update control coefficients
Update the sculpted object

Fig. 16. Our system is decomposed into three threads operating on the implicit object.

complicated geometry with arbitrary topologies. The system allows designers to create interesting objects in real-time. Designers can start from scratch or from an existing object through the use of our hierarchical fitting algorithm to transform other data-types to continuous B-spline implicit representations. Whenever a sculpting tool is used to sculpt an object, the density values of the working space at the affected regions will be modified correspondingly. Then, the system will reconstruct the volumetric implicit function to represent the new, modified object that has undergone deformation. The iso-surface of the object can be displayed interactively using the Marching Cubes technique [49], [15], [50]. By integrating physics-based modeling with a haptics interface, our force-based, haptic tools further allow users to reach toward an object, feel the physical presence of its shape, and sculpt free-form solids with force feedback. Since the sculpted object is discretized in a voxel raster, usually there are many homogeneously empty regions outside the object of interest. Therefore, an octree-based data structure is employed in our system (see Fig. 2), which can locate where the modification is performed and then only locally update the volumetric implicit function.

When using haptic tools, to reduce the latency and maximize the throughput, we resort to a parallel technique that can multithread the haptics, graphics, and sculpting processes with weak synchronization. This technique leads to a significant performance improvement and, ultimately, a parallel implementation of haptic sculpting, given high-end multiprocessor computing resources. Therefore, our system can be extended to many different configurations. Fig. 16 shows the communication between threads, where thick arrows represent data flow.

The haptic loop is implemented in a single thread. It maintains the haptic refresh rate, which is no less than 1 KHz. This requirement is critical to the realistic feedback of haptic interaction. If the update rate were below the threshold of 1 KHz, users would have an uncomfortable feeling. In our system, the haptic thread has the highest priority. It computes the haptic force and feeds it back to the haptic device.

The simulation loop is implemented in another thread, which controls the physical simulation. It continuously computes the total internal forces and external forces, then updates the physical states of a sculpted object as described in Section 5.3. In order to keep up with interactive frame rates, the physical simulation is limited to a small screen region by using the techniques described in Section 6.3. Usually, users' design intention and their sculpting operations would not exceed this limited region during one design cycle. In order to keep the simulation more stable, we employ a simple adaptive method to adjust the simulation time-step. Essentially, if $\ddot{p}$ is greater than a specified threshold, we shall use half of the previous time step as the current simulation time step.

The graphics loop is developed to handle the rendering of the volumetric objects. The rendering task makes use of the Marching Cubes algorithm and only updates a very small region in order to achieve interactive rendering rates and keep the graphics display consistent with the sculpting operations, physical simulation, and force feedback.

TABLE 1
Timings of Object Interaction with Geometric Tools

Coefficient resolution	Tool size	Update time(ms)
$64 \times 64 \times 64$	$10 \times 10 \times 10$	0.4
$64 \times 64 \times 64$	$20 \times 20 \times 20$	3.0
$64 \times 64 \times 64$	$40 \times 40 \times 40$	23.5
$128 \times 128 \times 128$	$20 \times 20 \times 20$	8.3
$128 \times 128 \times 128$	$40 \times 40 \times 40$	59.8

8 Results and Time Performance

We have conducted a large number of experiments and documented the running time for the sculpting of dynamic implicit solids. For sculpting with geometric tools, the experiments are based on a working space sampled at $128 \times 128 \times 128$. The geometric tool size is given as the number of data points that the tool affects. The results are detailed in Table 1.

Haptics-based deformation is time-critical. In this case, sometimes we may need to sacrifice the simulation loop speed and graphics loop speed in order to satisfy the haptics device refresh rate. Currently, our system permits the real-time simulation of thousands of mass-points. We have recorded the timings for physical simulations using various configurations of the control coefficients and the discretized space samples (see Table 2). This table contains the timings for one loop of the simulation thread and the graphical update under the condition that the haptics loop is always performed within 1ms.

Within our dynamic implicit modeling environment and using all the available geometric, physical, and haptic tools that we have developed, we have sculpted many interesting objects and several virtual-world scenes (see Fig. 17).

TABLE 2
Timings of Physical Simulation and Graphical Update

Coefficient resolution	Sampling points	Simulation time(ms)
$5 \times 5 \times 5$	$10 \times 10 \times 10$	1.0
$10 \times 10 \times 10$	$10 \times 10 \times 10$	6.9
$10 \times 10 \times 10$	$15 \times 15 \times 15$	15.8
$15 \times 15 \times 15$	$15 \times 15 \times 15$	38.2
$15 \times 15 \times 15$	$20 \times 20 \times 20$	74.9

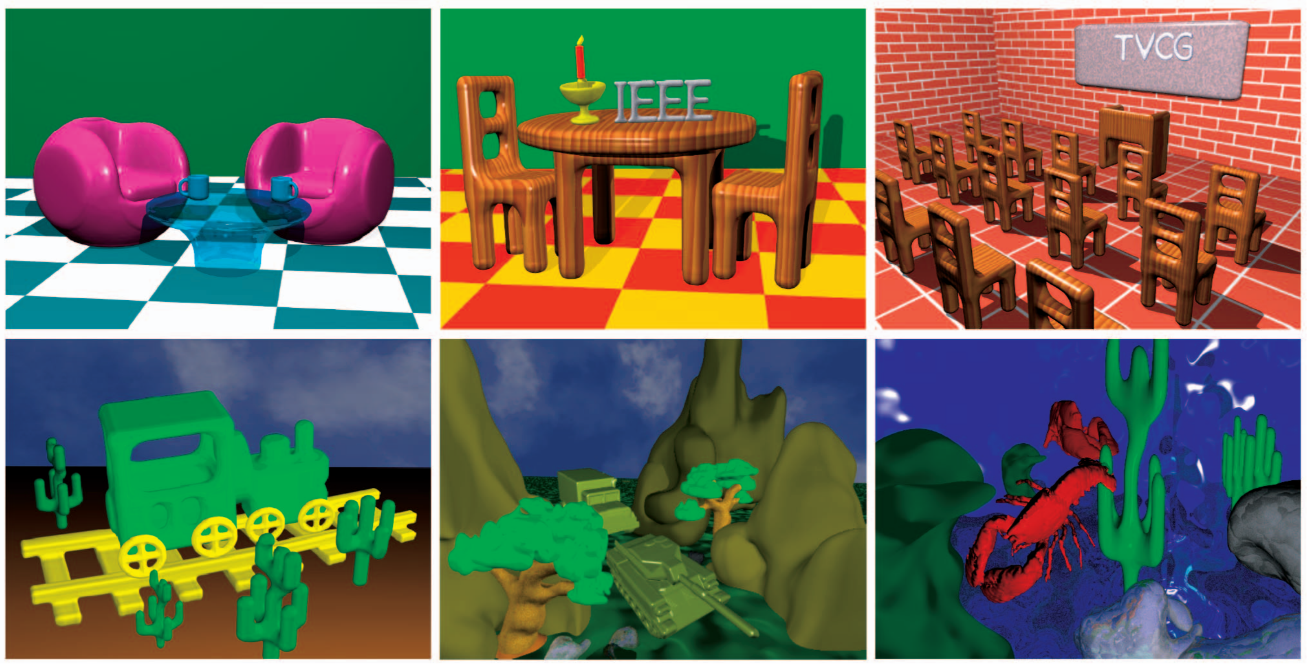


Fig. 17. Several sculptures and scenes created entirely within our haptics-based dynamic implicit solid modeling environment.

9 Conclusions

We have presented a novel haptics-based dynamic implicit solid modeling environment that employs trivariate scalar nonuniform B-splines as its underlying representation. We have proposed a new approach that unifies implicit functions, parametric representations, and physics-based modeling within a single haptics-based solid modeling framework. We have developed a large variety of algorithms and toolkits that afford designers the intuitive mechanism of interactive and direct manipulation of implicit solids with force feedback in real-time. The novel force mapping technique along with the concept of "density springs" (for dynamic modeling of implicit functions) can be straightforwardly extended to the haptic sculpting of any other types of implicit functions without additional difficulties. Moreover, our physics-based haptic tools can be directly employed to act on density-based volumetric data sets as well as solids implicitly defined by point clouds with the help of our hierarchical implicit function fitting algorithm. To facilitate multiresolution editing and LOD control, we have also incorporated three popular modeling techniques, hierarchical B-splines, CSG-based functional composition, and knot insertion, into our environment, making our dynamic implicit solid modeling techniques even more powerful and versatile to handle real-world solids of both complicated geometry and arbitrary topologies.

Acknowledgments

The authors would like to thank the associate editor, Dr. Roger Crawfis, and the anonymous reviewers for their careful and constructive reviews that have greatly helped them to improve the quality of this paper. Also, the authors would like to thank Christopher Carner and Kevin T. McDonnell for proofreading the draft. This research was supported in part by US National Science Foundation (NSF) CAREER award CCR-9896123, NSF grant DMI-9896170, NSF ITR grant IIS-0082035, NSF grants IIS-0097646, ITR IIS-0326388, ACI-0328930, the Alfred P. Sloan Fellowship, and the Honda Initiation Award.

References

[1] A.A.G. Requicha and J.R. Rossignac, "Solid Modeling and Beyond," *IEEE Computer Graphics and Applications,* vol. 12, no. 5, pp. 31-44, 1992.
[2] T.A. Galyean and J.F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," *Computer Graphics,* vol. 25, no. 4, pp. 267-274, 1991.
[3] S.W. Wang and A.E. Kaufman, "Volume Sculpting," *Proc. 1995 Symp. Interactive 3D Graphics,* pp. 151-156, 1995.
[4] S. Frisken, R. Perry, A. Rockwood, and T. Jones, "Adaptive Sampled Distance Fields: A General Representation of Shape for Computer Graphics," *SIGGRAPH '00 Proc.,* pp. 249-254, 2000.
[5] R.N. Perry and S.F. Frisken, "Kizamu: A System for Sculpting Digital Characters," *SIGGRAPH '01 Proc.,* pp. 47-56, 2001.
[6] J. Bloomenthal, *Introduction to Implicit Surfaces,* J. Bloomenthal et al., eds. Morgan Kaufmann, 1997.
[7] A. Raviv and G. Elber, "Three Dimensional Freeform Sculpting via Zero Sets of Scalar Trivariate Functions," *Proc. Fifth ACM Symp. Solid Modeling and Applications,* pp. 246-257, 1999.
[8] G. Turk and J.F. O'Brien, "Modelling with Implicit Surfaces that Interpolate," *ACM Trans. Graphics,* vol. 21, no. 4, pp. 855-873, 2002.
[9] J. Hua and H. Qin, "Haptic Sculpting of Volumetric Implicit Functions," *Proc. Ninth Pacific Conf. Computer Graphics and Applications,* pp. 254-264, 2001.
[10] K.T. McDonnell, H. Qin, and R.A. Wlodarczyk, "Virtual Clay: A Real-Time Sculpting System with Haptic Toolkits," *Proc. 2001 Symp. Interactive 3D Graphics,* pp. 179-190, 2001.
[11] J. Hua and H. Qin, "Haptics-Based Volumetric Modeling Using Dynamic Spline-Based Implicit Functions," *Proc. IEEE Symp. Volume Visualization and Graphics 2002,* pp. 55-64, 2002.
[12] J.F. Blinn, "Generalization of Algebraic Surface Drawing," *ACM Trans. Graphics,* vol. 1, no. 3, pp. 235-256, 1982.
[13] J. Bloomenthal and B. Wyvill, "Interactive Techniques for Implicit Modeling," *Proc. 1990 Symp. Interactive 3D Graphics,* pp. 109-116, 1990.
[14] B. Wyvill and G. Wyvill, *Using Soft Objects in Computer Generated Animation.* New York: Springer-Verlag, 1986.

[15] G. Wyvill, C. McPheeters, and B. Wyvill, "Data Structure for Soft Objects," *The Visual Computer,* vol. 2, no. 4, pp. 227-234, 1988.

[16] J. Bloomenthal and K. Shoemake, "Convolution Surfaces," *SIGGRAPH '91 Proc.,* pp. 251-256, 1991.

[17] J. McCormack and A. Sherstyuk, "Creating and Rendering Convolution Surfaces," *Computer Graphics Forum,* vol. 17, no. 2, pp. 113-120, 1998.

[18] J.C. Hart, A. Durr, and D. Harsh, "Critical Points of Polynomial Meatballs," *Proc. Implicit Surfaces '98, Eurographics/SIGGRAPH Workshop,* pp. 69-76, 1998.

[19] A. Witkin and P. Heckbert, "Using Particles to Sample and Control Implicit Surfaces," *SIGGRAPH '94 Proc.,* pp. 269-278, 1994.

[20] W. Martin and E. Cohen, "Representation and Extraction of Volumetric Attributes Using Trivariate Splines: A Mathematical Framework," *Proc. Seventh ACM Symp. Solid Modeling and Applications,* pp. 234-240, 2001.

[21] B. Schmitt, A. Pasko, and C. Schlick, "Constructive Modeling of Frep Solids Using Spline Volumes," *Proc. Sixth ACM Symp. Solid Modeling and Applications,* pp. 321-322, 2001.

[22] E. Ferley, M.P. Cani, and J.-D. Gascuel, "Practical Volumetric Sculpting," *The Visual Computer,* vol. 16, no. 7, pp. 469-480, 2000.

[23] K. Museth, D.E. Breen, R.T. Whitaker, and A.H. Barr, "Level Set Surface Editing Operators," *SIGGRAPH '02 Proc.,* pp. 330-338, 2002.

[24] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," *Computer Graphics,* vol. 21, no. 4, pp. 205-214, 1987.

[25] A. Pentland and J. Williams, "Good Vibrations: Modal Dynamics for Graphics and Animation," *Computer Graphics,* vol. 23, no. 3, pp. 215-222, 1989.

[26] D. Metaxas and D. Terzopoulos, "Dynamic Deformation of Solid Primitives with Constraints," *Computer Graphics,* vol. 26, no. 2, pp. 309-312, 1992.

[27] M.P. Cani and M. Desbrun, "Animation of Deformable Models Using Implicit Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 3, no. 1, pp. 39-50, Jan.-Mar. 1997.

[28] M. Kass and G. Miller, "Rapid, Stable Fluid Dynamics for Computer Graphics," *SIGGRAPH '90 Proc.,* pp. 49-57, 1990.

[29] D. Baraff and A. Witkin, "Dynamic Simulation of Non-Penetrating Flexible Bodies," *SIGGRAPH '92 Proc.,* pp. 303-308, 1992.

[30] N. Foster and D. Metaxas, "Modeling the Motion of Hot, Turbulent Gas," *SIGGRAPH '97 Proc.,* pp. 181-188, 1997.

[31] J.F. O'Brien, A.W. Bargteil, J.K. Hodgins, "Graphical Modeling and Animation of Ductile Fracture," *SIGGRAPH '02 Proc.,* pp. 23-26, 2002.

[32] R. Szeliski and D. Tonnesen, "Surface Modeling with Oriented Particle Systems," *SIGGRAPH '92 Proc.,* pp. 185-194, 1992.

[33] H. Qin and D. Terzopoulos, "D-NURBS: A Physics-Based Framework for Geometric Design," *IEEE Trans. Visualization and Computer Graphics,* vol. 2, no. 1, pp. 85-96, Mar. 1996.

[34] C. Mandal, H. Qin, and B.C. Vemuri, "A Novel Fem-Based Dynamic Framework for Subdivision Surfaces," *Proc. Fifth ACM Symp. Solid Modeling and Applications,* pp. 191-202, 1999.

[35] J.K. Salisbury, D. Brocki, T. Massiet, N. Swarupf, and C. Zillest, "Haptic Rendering: Programming Touch with Virtual Objects," *Proc. 1995 Symp. Interactive 3D Graphics,* pp. 123-130, 1995.

[36] J.K. Salisbury and C. Tarr, "Phantom-Based Haptic Interaction with Virtual Objects," *IEEE Computer Graphics and Applications,* vol. 17, no. 5, pp. 6-10, 1997.

[37] C.B. Zilles and J.K. Salisbury, "A Constraint-Based God-Object Method for Haptic Display," *Proc. Int'l Conf. Intelligent Robots and Systems,* pp. 3146-3152, 1995.

[38] H.B. Morgenbesser and M.A. Srinivasan, "Force Shading for Haptic Perception," *Proc. 1996 ASME Int'l Mechanical Eng. Congress and Exposition, Dynamic Systems and Control Division,* pp. 407-412, 1996.

[39] J.K. Salisbury and C. Tarr, "Haptic Rendering of Surfaces Defined by Implicit Functions," *Proc. ASME Sixth Ann. Sympo. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* pp. 15-21, 1997.

[40] L. Kim, A. Kyrikou, G.S. Sukhatme, and M. Desbrun, "An Implicit-Based Haptic Rendering Technique," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots,* 2002.

[41] R.S. Avila and L.M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Proc. Seventh IEEE Visualization '96,* pp. 197-204, 1996.

[42] T.V. Thompson, D.E. Johnson, and E. Cohen, "Direct Haptic Rendering of Sculptured Models," *Proc. 1997 Symp. Interactive 3D Graphics,* pp. 167-176, 1997.

[43] F. Dachille, H. Qin, and A.E. Kaufman, "A Novel Haptics-Based Interface and Sculpting System for Physics-Based Geometric Design," *Computer-Aided Design,* vol. 33, no. 5, pp. 403-420, 2001.

[44] R. Balakrishnan, G. Fitzmaurice, G. Kurtenbach, and K. Singh, "Exploring Interactive Curve and Surface Manipulation Using a Bend and Twist Sensitive Input Strip," *Proc. 1999 Symp. Interactive 3D Graphics,* pp. 111-118, 1999.

[45] S. Muraki, "Volumetric Shape Description of Range Data Using Blobby Model," *Computer Graphics,* vol. 25, no. 4, pp. 227-235, 1991.

[46] G. Turk and J.F. O'Brien, "Shape Transformation Using Variational Implicit Surface," *SIGGRAPH '99 Proc.,* pp. 335-342, 1999.

[47] J.C. Carr, R.K. Beatson, and J.B. Cherrie, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," *SIGGRAPH '01 Proc.,* pp. 67-76, 2001.

[48] H. Zhao, S. Osher, and R. Fedkiw, "Fast Surface Reconstruction and Deformation Using the Level Set Method," *Proc. IEEE Workshop Variational and Level Set Methods in Computer Vision,* pp. 194-201, 2001.

[49] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics,* vol. 21, no. 4, pp. 163-169, 1987.

[50] J. Wilhelms and A.V. Gelder, "Octrees for Faster Isosurface Generation," *ACM Trans. Graphics,* vol. 11, no. 3, pp. 201-227, 1992.

Jing Hua is a PhD candidate in computer science at the State University of New York at Stony Brook, where he is also a research assistant in the SUNYSB Center for Visual Computing (CVC). He received the BS degree (1996) in electrical engineering from the Huazhong University of Science and Technology in Wuhan, People's Republic of China. He received the ME degree (1999) in pattern recognition and artificial intelligence from the Institute of Automation, Chinese Academy of Sciences in Beijing, People's Republic of China. In 2001, he received the MS degree in computer science from the State University of New York at Stony Brook. His research interests include geometric and physics-based modeling, scientific visualization, interactive 3D graphics, human-computer interaction, and computer vision. He is a student member of the IEEE and ACM. For more information see http://www.cs.sunysb.edu/~jinghua.

Hong Qin received the BS degree (1986) and the MS degree (1989) in computer science from Peking University in Beijing, People's Republic of China. He received the PhD degree (1995) in computer science from the University of Toronto. He is an associate professor of computer science at the State University of New York at Stony Brook, where he is also a member of the SUNYSB Center for Visual Computing. From 1989-1990, he was a research scientist at the North-China Institute of Computing Technologies. From 1990-1991, he was a PhD candidate in computer science at the University of North Carolina at Chapel Hill. From 1996-1997, he was an assistant professor of Computer & Information Science & Engineering at the University of Florida. In 1997, he was awarded the US National Science Foundation (NSF) CAREER Award and, in September 2000, was awarded a newly established NSF Information Technology Research (ITR) grant. In December 2000, he received a Honda Initiation Grant Award and, in April 2001, was selected as an Alfred P. Sloan Research Fellow by the Sloan Foundation. He is a member of the ACM, IEEE, SIAM, and Eurographics.