Haptics-based Volumetric Modeling Using Dynamic Spline-based Implicit Functions

Jing Hua [*] Hong Qin [†]

State University of New York at Stony Brook

Abstract

This paper systematically presents a novel haptics-based volumetric modeling framework, which is founded upon volumetric implicit functions and powerful physics-based modeling. The volumetric implicit functions incorporate hierarchical B-splines, CSG-based functional composition, and knot insertion to facilitate multiresolution editing and level of details (LODs) control. Our dynamic volumes are semi-algebraic sets of implicit functions and are governed by the principle of dynamics, hence responding to sculpting forces in a natural and predictive manner. The versatility of our volumetric modeling affords users to easily modify both the geometry and the topology of modeled objects, while the inherent physical properties can offer an intuitive mechanism for direct manipulation. Moreover, we augment our modeling environment with a natural haptic interface, in order to take advantage of the additional realism associated with 3D haptic interaction. Coupling physics and haptics with implicit functions can realize all the potentials exhibited by volumetric modeling, physics-based modeling, and haptic interface. Furthermore, in order to directly manipulate existing volumetric datasets as well as point clouds, we develop a hierarchical fitting algorithm to reconstruct and represent discrete datasets using our continuous implicit functions, which permit users to further design and edit those 3D models in real-time using a large variety of haptic toolkits and visualize their interactive deformation at arbitrary resolution.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Haptic I/O; I.3.m [Computer Graphics]: Miscellaneous—Implicit function;

1 Introduction

Volumetric modeling is vital for interactive graphics, virtual environments, and visualization. Despite many modeling advantages of implicit functions over popular parametric geometry [Bloomenthal 1997][Bloomenthal and Wyvill 1990], there are still certain difficulties such as the lack of flexible, interactive design techniques and fast, direct rendering methods which hinder the widespread penetration of implicit functions into volume graphics. There are fewer convenient modeling tools for the intuitive shape control of this type of algebraic volumes. In general, flexible and direct modeling techniques for implicit function based volumes remain under-explored.

To ameliorate it, we integrate volumetric implicit functions and powerful physics-based modeling into one single framework: *Dynamic Volumetric Modeling*, and systematically present a haptics-based volumetric modeling environment, which permits interactive and direct manipulation of volumes characterized by implicit functions in real-time and with realistic force feedback. Furthermore, by using a new hierarchical fitting algorithm, our environment provides a flexible mechanism for users to interact with existing volumetric modeling systems and their datasets of different representations, hence allowing users to manipulate and sculpt the existing point clouds and volumetric datasets using a large number of toolkits available in our environment. Our modeling methodology and the new environment aim to realize all the potentials offered by volumetric modeling, implicit functions, physics-based modeling, and haptic interaction.

Unlike the discrete volumetric object representation used in [Galyean and Hughes 1991], [Wang and Kaufman 1995] and [Perry and Frisken 2001], the modeled object in our system is evaluated as a level-set of volumetric implicit functions defined over a 3D working space. Arbitrary topology and complicated geometry are implicitly defined by the functions. 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 and are continuous, depending on the choice of basis functions. In this paper, the volumetric implicit functions combine the benefits of conventional implicit functions and popular parametric representations. They are defined by a 3D hierarchical organization of the underlying constituent scalar B-splines [Hua and Qin 2001]. Although uniform or non-uniform trivariate B-splines are employed as the underlying function, constituent functions may be of arbitrary type with or without the local control property. It should be noted that in [Hua and Qin 2001] there are neither material quantities nor dynamic behaviors in the system. The direct manipulation was achieved by solving a static linear system. In order to define material attributes and dynamic behaviors in the system and achieve direct haptics-based manipulation, we employ a novel technique to associate physics with implicit functions. Physical attributes are assigned inside the working space, and haptic forces are computed directly from material properties. Consequently, geometric parameters of arbitrary implicit functions are hidden from users through the use of natural, force-based interfaces. Our system synchronizes the geometric and physical representations of modeled objects during the modeling process. Such dynamic and interactive approaches can provide users with a natural, force-based interface as well as a geometric interface at the same time.

Instead of modifying the coefficients associated with the volumetric implicit function as shown in [Raviv and Elber 1999], our sculpting tools facilitate the direct editing of implicit functions' scalar values. Our algorithm can automatically determine all of

[*]e-mail: jinghua@cs.sunysb.edu
[†]e-mail: qin@cs.sunysb.edu

the unknown control coefficients and effectively reconstruct a new volumetric implicit function undergoing the local/global free-form deformation. The additional constraints allow users to gain more sophisticated control over the dynamic models and undertake cross-sectional design tasks (i.e., generating volumes from a set of curve profiles). Our system offers a wider array of intuitive sculpting tools (especially the novel haptics-based tools) responsible for the effective construction of various complicated volumetric shapes with distinct topological types. These tools are both transparent to and independent of the underlying representations. With a standard haptic device, our approach permits users to interactively sculpt virtual materials having real-world properties and feel the physically realistic presence with force feedback throughout the design process. In essence, haptics-based exploration provides additional sensory cues to designers and can afford designers to gain a richer understanding on the 3D nature of volumetric objects.

There is little work on integrating physics-based modeling with volumetric implicit functions and applying haptic tools on volumetric datasets and point clouds due to certain difficulties. Our work aims to incorporate the physics in general and the elasticity in particular into volumetric models characterized by implicit functions and advance the state of the knowledge in the effective integration of volumetric modeling, implicit functions, physics-based modeling, and haptic interaction.

2 Prior Work

Implicit functions are well suited for both scientific visualization and the modeling tasks in computer graphics [Blinn 1982]. In order to design implicit surfaces interactively, [Bloomenthal 1997] and [Bloomenthal and Wyvill 1990] used skeleton methods to construct implicit surfaces. Each skeletal element is associated with a locally defined implicit function. Individual functions are blended to form an implicit surface using a polynomial weighting function that can be controlled by users. Blobby model [Blinn 1982], also known as soft objects [Wyvill et al. 1988], is another popular technique for the design of implicit surfaces. Implicit functions are also used to represent volumes. Commonly-used modeling techniques includes Boolean operations and functional compositions.

Recently, [Raviv and Elber 1999] presented a 3D interactive sculpting paradigm that employed a set of scalar uniform trivariate B-spline functions as object representations. Users can indirectly sculpt objects to a desirable shape by directly modifying relevant scalar control coefficients of the underlying functions with virtual sculpting tools. In other related work, the representation of sculpted volumetric objects is primarily of discrete type (e.g., voxels). [Galyean and Hughes 1991] first introduced the concept of volume sculpting and developed a system with simple tools. [Wang and Kaufman 1995] presented a similar sculpting system with sculpting tools of carving and sawing. In a nutshell, those sculpting systems are dependent on the simple, voxel-based operation. The sculpted objects and the sculpting tools are represented using a discrete characteristic function. Unfortunately, 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 the similar idea, Ferley *et al.* presented a rapid shape-prototyping system [Ferley et al. 2000]. [Perry and Frisken 2001] presented a sculpting system, *Kizamu*, for creating digital characters. This system employed adaptively sampled distance fields (ADFs) as volumetric shape representations. [McDonnell et al. 2001] implemented a 3D sculpting system based on dynamic subdivision-based solids. However, much effort has to be taken to change the topology of subdivision-based sculptures.

Generally, each of aforementioned systems suffers from some of the limitations. Traditional modeling techniques may be inconvenient for representing complicated volumes, because modelers are faced with the tedium of indirect shape modification and refinement through time-consuming operations on a large number of control coefficients. In contrast, physics-based models respond to externally applied forces in a very intuitive manner. The dynamic formulation marries the model geometry with time, mass and damping distributions, and constraints via Lagrangian equations of motion. Dynamic models produce smooth, natural motions that are intuitive to control. In addition, they facilitate direct manipulation of complex geometries and topologies.

Free-form deformable models were first introduced to computer graphics by Terzopoulos *et al.* [Terzopoulos et al. 1987]. They employed elasticity theory to construct differential equations that model the behavior of non-rigid curves, surfaces, and solids as functions of time. Deformable models were further developed by [Pentland and Williams 1989], and [Metaxas and Terzopoulos 1992]. [Cani and Desbrun 1997] employed deformable implicit models for animation. Despite the popular use of physics-based models in graphics, less effort has been applied to free-form dynamic interaction between designers and manufactured objects which is especially useful for volumetric modeling. Qin and Terzopoulos introduced D-NURBS surfaces, an extension to traditional NURBS that permits more natural control of the surface geometry [Qin and Terzopoulos 1996]. Note that physical simulation can be used as an effective, interactive tool for building and manipulating a wide range of models.

Haptic rendering is a process of applying forces through the use of force-feedback devices and augmenting a virtual environment with a haptic interaction. [Thompson et al. 1997] derived efficient intersection techniques that can be applied to NURBS-based rigid surfaces. [Dachille et al. 2001] developed a haptic interface to permit the direct manipulation of dynamic B-spline surfaces. Due to the limits of parametric B-spline, it cannot handle arbitrary topology and complicated geometry. [McDonnell et al. 2001] employed haptic toolkits to explore the dynamic subdivision solids. [Avila and Sobierajski 1996] presented a haptic interaction that is suitable for both volume visualization and modeling applications. Despite the widespread utilization of haptics in visual computing areas, haptics-based interaction was primarily applied to parametric representations for shape modeling and sculpting. We integrate the principle of haptic modeling with the direct manipulation of dynamic volumes and employ force-based, haptic tools to directly work on density-centered volumetric datasets and surface geometry characterized by point clouds scattered over solid boundaries.

3 Volumetric Implicit Functions

An arbitrary implicit function in 3D can be generally characterized as:

$$\{(x,y,z)|F(x,y,z)=0\}. \tag{1}$$

Collecting all the level-sets whose return values are greater (or smaller) than a given threshold, we can define an implicit solid:

$$\left\{\begin{array}{l} c = F(x,y,z) \\ c > c_0 \end{array}\right. . \tag{2}$$

In particular, this paper utilizes scalar trivariate B-spline functions as the underlying shape primitives. We shall collect different scalar 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, implicit B-spline functions formulate the scalar value distribution in 3D where implicit solids are uniquely defined as semi-algebraic point sets.

Consider N B-spline patches in the working space, which are located at any location and with any orientation. In general, these patches may be formulated by any scalar B-splines $s(u,v,w)$ with different numbers of control coefficients in order to achieve the goal of multiresolution analysis and LOD control. u,v,w represent three coordinates of the parametric domain. Then the scalar value at the location (x,y,z) can be computed as

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

where $\mathbf{T}_i$ is an affine transformation from the Euclidian space to the parametric domain of patch s_i. For every different patch s_i, there is a corresponding transformation $\mathbf{T}_i$. Now $F(x,y,z)$ becomes a new volumetric implicit function defined over the 3D working space. In essence, (3) is a hierarchical organization of the N patches. For the details about scalar B-spline expressions and the spline-based volumetric implicit functions, please refer to [Hua and Qin 2001].

4 Dynamic Volumetric Models

4.1 Integration of Elasticity with Implicit Solids

In order to introduce physics into our environment, the sculpted object of a B-spline based implicit function is discretized into a voxel raster. Every voxel contains a scalar value, called 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. Figure 1 shows a simple sculpted object and its corresponding voxelmap. This voxelmap defines a function, where the solid particles (colored in red) denote locations in which material exists and the empty particles (colored in gray) denote locations in which there is no material. Note that, in our system the characteristic function is not a binary function, rather it is a continuous function.

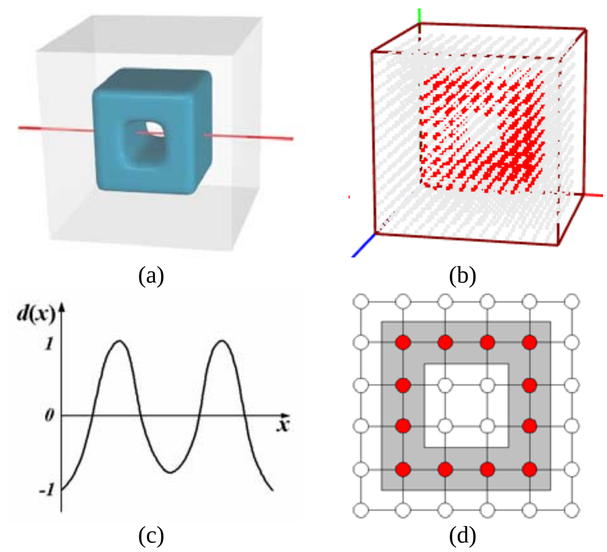


Figure 1: (a) A simple sculpted object. (b) Its corresponding voxelmap. (c) The density distribution along the red line. (d) The voxelmap on a 2D cross-section.

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

$$\mathbf{d} = \mathbf{Ap} \tag{4}$$

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

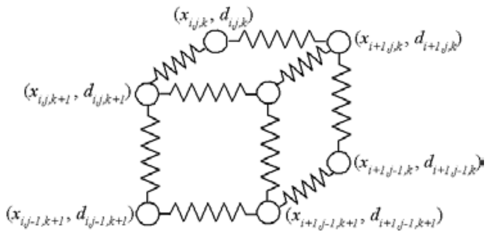

$(x_{i,j,k}, d_{i,j,k})$
$(x_{i+1,j,k}, d_{i+1,j,k})$
$(x_{i,j,k+1}, d_{i,j,k+1})$
$(x_{i+1,j-1,k}, d_{i+1,j-1,k})$
$(x_{i,j-1,k+1}, d_{i,j-1,k+1})$
$(x_{i+1,j-1,k+1}, d_{i+1,j-1,k+1})$

Figure 2: 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 space, $d_{i,j,k}$ is the density at that position.

The discretized density field is assigned 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 oftentimes can be considered to be constant. However, these material distributions are allowed to be modified by users interactively and directly across the volumetric domain in real-time. The discretized field is 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. Figure 2 shows the mass-spring network in the vicinity of a point.

We refer those springs as "density springs". This is because, these springs are unconventional in a sense that they are significantly 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 for implicit functions do not change the geometric position $x_{i,j,k}$ of the mass $m_{i,j,k}$ at all. Instead, they permit the magnitude change of the density $d_{i,j,k}$ of the mass-point $m_{i,j,k}$. Essentially, this new type of springs will attract/dispel density values of neighbors. When users manipulate the implicit solids, the density values are changed by the mass-spring system. Consequently, this results in the deformable behavior of the object's shape modeled by the spline-based volumetric implicit function. So the elasticity has been introduced to our volumetric implicit objects, and our implicit solids now become deformable volumes, which we name as *dynamic volumetric models* or *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 isosurface from the density field instead of from the geometric position of masses. 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 discretized density field is formulated as a discrete simulation of Lagrangian dynamics:

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{D}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f_d}, \tag{5}$$

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 $\mathbf{f} = \mathbf{k}(\mathbf{l} - \mathbf{l}_0)$ according to Hook's law. In our system the geometric positions of mass-points do not move and only density values change. So 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 non-linear phenomena are more desirable.

4.2 Response to Applied Forces

As we know, a deformable model is defined by a given correspondence between applied forces and deformation. In order to allow direct deformation of dynamic volumetric models in a force-based physical manner, we must address the important issue of force mapping, which defines how dynamic models response to applied forces. Note that, the generated forces will be input to the dynamic system as external forces and will also be fed back 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 Figure 3, 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_0 and x_1. 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. So 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.

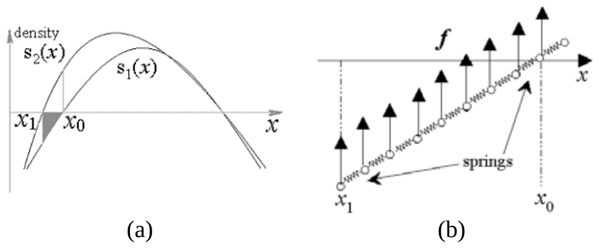


Figure 3: (a) Iso-surface changing from x_0 to x_1 via applied force f, which is proportional to the gray area. (b) Close-up view of the mass-spring network of $s_1(x)$, where f is applied on every mass-point between x_0 and x_1.

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, so

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

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 Figure 3. In our system we define the following conventions to enforce the consistency. The positive force is to increase density value and the negative force is to decrease the density value. Note that, f is decreasing over time as x_0 moves towards x_1.

The force mapping mechanism of our system is very general, which can deal with iso-surface enlarging (as shown above) as well as iso-surface shrinking with the same formula for force calculation. In Figure 3, 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 3D,

$$f = -\int_C s(u,v,w)dc, \tag{6}$$

where C is any force vector, $s(u,v,w)$ is the density distribution function in the 3D working space. When f is input as external forces into (5), the density field will be changed and this will result in the deformation of the dynamic volumetric model.

4.3 Numerical Solver and Simulation

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 $\ddot{\mathbf{d}}$ with $\mathbf{A}\ddot{\mathbf{p}}$, we obtain:

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

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

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

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

Then the model's control coefficients and their velocity can be computed using a forward Euler method:

$$\begin{array}{l} \dot{\mathbf{p}}_{i+1} = \dot{\mathbf{p}}_i + \ddot{\mathbf{p}}_i \triangle t \\ \mathbf{p}_{i+1} = \mathbf{p}_i + \dot{\mathbf{p}}_i \triangle t \end{array} . \tag{8}$$

The updated control coefficients $\mathbf{p}_{i+1}$ are further used to update the discretized field defined by $\mathbf{d}_{i+1} = \mathbf{A}\mathbf{p}_{i+1}$. Here we can see that the simulation does not change the geometric position of mass-points. It only updates the density value of every mass-point. Note that, the volumetric implicit function can be evaluated anywhere at arbitrary resolution once $\mathbf{p}$ is known. After generating the new density field (or say new implicit representation), the new applied forces are calculated and will be applied in subsequent simulation steps.

As a result, the new dynamic approach can continuously evolve the implicit functions. So we refer the implicit functions as *dynamic spline-based volumetric implicit functions*. This approach permit users to directly work on both the level-set geometry and the enclosed material distribution with a continuous visual and haptic feedback. Although the more robust, implicit Euler solver is readily available in our system, we decide to employ a simpler, forward method for the purpose of real-time, haptic interaction.

5 Hierarchical Implicit Function Fitting

In order to allow users to conduct further editing on existing solid objects, we transform the discrete solid representation of modeled objects to the continuous representation in our environment. Then the solids can be sculpted using the interactive tools available in our system. That is to say, we want to reconstruct and represent 3D objects using our own volumetric implicit functions. We shall find a volumetric implicit function f, which implicitly defines any user-specified solid of various types. The volumetric implicit function offers a compact functional description for a set of discrete, input data. It can be evaluated anywhere to produce a mesh at arbitrary, desirable resolution. In the previous work, [Muraki 1991] proposed algorithm to reconstruct range data using blobby model. [Turk and O'Brien 1999] and [Carr et al. 2001] used radial basis functions to reconstruct and represent point clouds. Our simple reconstruction algorithm can handle point clouds as well as volumetric datasets.

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

Problem 1 *Find f such that*

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

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.*

The iso-surface points are always given by point clouds. However, there is still a problem on how to generate the off-surface points and their corresponding d_i. There has been a lot of research work on this topic [Turk and O'Brien 1999][Carr et al. 2001]. One viable solution for this problem is a signed-distance field, where the d_i is the distance to the closest iso-surface point. Points outside the solid are assigned negative values, while points inside are assigned positive values. Obviously, we don't need to generate the entire distance field. From our experiments, it is sufficient to produce two off-surface points associated with each iso-surface point, one outside and the other one inside. We employ the tagging algorithm recently proposed by [Zhao et al. 2001] and slightly modify it to compute the required signed distance field, then we make use of the least-square fitting to obtain the volumetric implicit function, whose zero-level-set fits the given point clouds.

In general, solid objects are of huge size. So, using a single B-spline to fit the solids is impractical since the size of basis matrices will be too large to handle and the fitting error will be unacceptable. Our system utilizes the volumetric implicit function (See Equation 3) to obtain hierarchical implicit B-spline representations for the objects. The octree-based subdivision scheme is employed to subdivide the working space containing the 3D 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.

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,

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

   where N_i denotes the number of sampling points inside the region of node i.

4. If ε_i is less than the user-specified error bound ε, then mark the node as a leaf node.

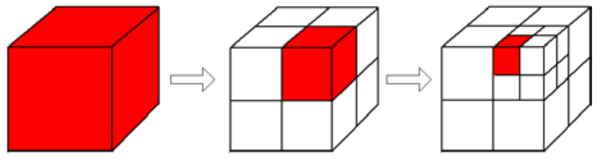


Figure 4: Illustration of an octree-based subdivision scheme and a hierarchical fitting structure, where the red color denotes the node that needs to be subdivided.

5. Else subdivide the node i to eight child nodes and go to 2.

This algorithm will not stop until all the child nodes are marked as leaf nodes. Then the discrete point clouds are converted to a continuous spline-based volumetric implicit function, which can be evaluated at arbitrary sampling resolution and rendered with the Marching Cubes algorithm. This reconstruction process allows users to conduct further editing using the tools available in our modeling environment. Figure 4 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.

Our fitting algorithm can handle several types of point clouds, including scattered datasets, corrupted datasets, cross-sectional datasets, and noisy datasets. Figure 5(a) shows a set of point clouds, which are non-uniformly distributed, Figure 5(b) is the cross-sectional view of the distance field, and Figure 5(c) is the fitted solid. Figure 6(a) shows a torus, in which a portion of the points near the top of the model has been removed. Through the use of the tagging algorithm to generate a signed distance field, a flattened area in the neighborhood of the gap can be created as shown in Figure 6(b). So in Figure 6(c) we can see the hole in the torus is filled nicely with a minimal surface.

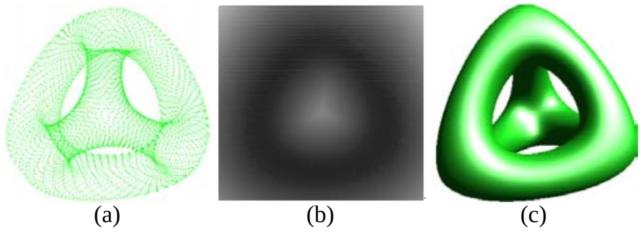


Figure 5: (a) A point-sampled smooth tetrahedra with four holes. (b) Cross-sectional view of the generated distance field. (c) The reconstructed object represented by a volumetric implicit function.

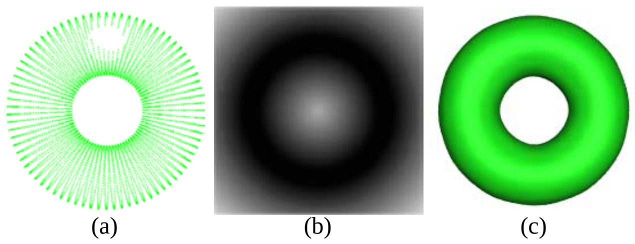


Figure 6: (a) The point clouds of a damaged torus. (b) Cross-sectional view of the distance field. (c) The reconstructed torus represented by a volumetric implicit function (The hole is nicely filled).

Besides point clouds, our system can also transform volumetric datasets to the spline-based volumetric implicit functions. In

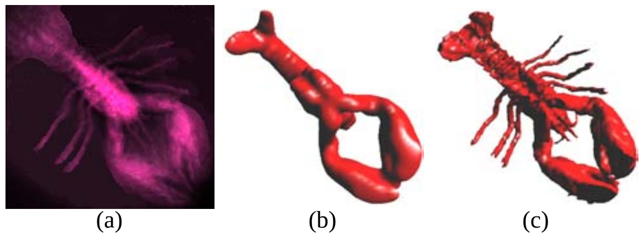


Figure 7: (a) Volume rendering of the original density field. (b) Fitting with fewer octree layers and control coefficients. (c) The reconstructed volume object represented by a volumetric implicit function.

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. Figure 7 shows a volumetric object and its fitted implicit solids.

6 System Description

Our dynamic implicit modeling environment can handle both complicated geometry and arbitrary topologies. Designers can create interesting objects from scratch or from some existing objects through the use of our hierarchical fitting algorithm to transform other datatypes to continuous B-spline implicit representations. The system provides a large number of virtual sculpting tools to a wide spectrum of users. 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 undergone deformation. By using local Marching Cubes technique [Lorensen and Cline 1987][Wyvill et al. 1988], the iso-surface of the object can be displayed interactively.

By integrating physics-based modeling with a haptic interface, our force-based tools further allow users to reach toward an object, feel the physical presence of its shape, and sculpt free-form solids with force feedback. The feedback forces are computed based on the object geometry and the associated physical properties. Figure 8 shows the haptics-based user interface of our modeling environment. In Figure 8, a bending operation on the red lobster is being performed by a user.

When using haptic tools, to reduce the latency and maximize the throughput, we resort to a parallel technique that can multi-thread the haptics, graphics, and simulation processes with weak synchronization. This technique leads to a significant performance improvement, and ultimately, a parallel implementation of haptic sculpting, in the presence of high-end multi-processor computing resources.

The haptics loop is implemented in a single thread. It maintains the haptic refresh rate, which is no less than 1KHz. This requirement is critical to the realistic feedback of haptic interaction. In our system, the haptic thread has the highest priority. It computes the haptic force and feed 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 4.3. In order to keep up with interactive frame rate, the physical simulation is limited to a small region by using the techniques described in Section 7.3. 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.

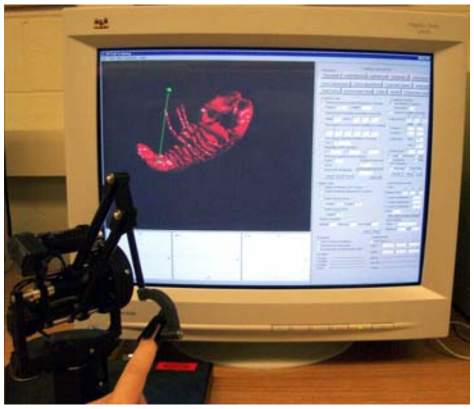


Figure 8: Haptics-based user interface.

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

Since the sculpted object is discretized in a voxel raster, usually there are many homogeneously empty regions outside the object of interest. Separating those regions from the sculpting regions will significantly reduce the memory consumption and speed up the volume rendering and modeling tasks. Therefore, an octree-based data structure is employed in our system (See Figure 4), which can locate where the modification is performed, and then only locally update the volumetric implicit function for efficiency purpose. The local update property can speed up the Marching Cubes rendering by only conducting the re-evaluation task of the modified parts. The octree-based subdivision also helps our fitting algorithm construct a hierarchical structure of the fitted objects.

7 Sculpting Toolkits

Since our volumetric modeling framework integrates geometry and physics, our modeling environment can provide two primary types of sculpting tools: one type of tools is called ”geometric tools”, and the other one is called ”force-based tools”. Geometric tools do not involve in the physical simulation, while the force-based tools afford dynamic property and haptic interaction.

7.1 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 reconstruct the volumetric implicit function of B-splines according to the new density distribution by using the least-square fitting [Hua and Qin 2001]. Our geometric tools can be easily defined using any implicit functions. Therefore, using the geometric tools users can create objects of complicated geometry and arbitrary topology. More importantly, our modeling environment also allows designers to define their own tools with any implicit functions. Geometric tools facilitate precise sculpting operations on volumetric models with interactive speed. Figure 9 shows a number of sculpted examples using our geometric toolkits.

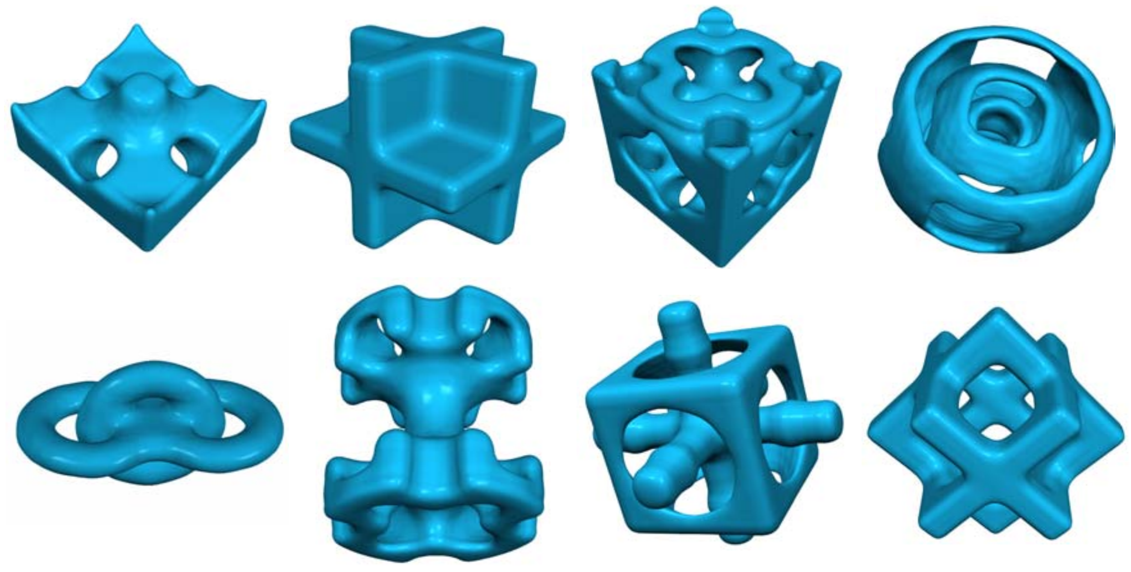


Figure 9: Sculpted examples using geometric tools.

7.2 Force-based Tools

Although geometric tools are powerful to create complicated volumetric shapes, they constitute a purely geometric representation. To perform free-form deformation, modelers have to design complicated tools based on popular function primitives. This kind of shape modification and refinement is time-consuming in general. To alleviate these problems, our modeling environment also offers force-based tools, which allow users to perform direct free-form deformation with ease.

7.2.1 Free-form deformation via Forces

In our system the simple force-based tool allows the user to grab the nearest mass-point in the solid. In addition, our system provides other tools to allow 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 function $\beta(x,y,z)$, which can be constant, Gaussian, spherical, cylindrical, conical, or any other distributions.

For point-based force sculpting, we can use the parametric form $\{(u(t),v(t),w(t))|t \in [t_0,t_1]\}$ to represent C in Equation (6), then we have

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

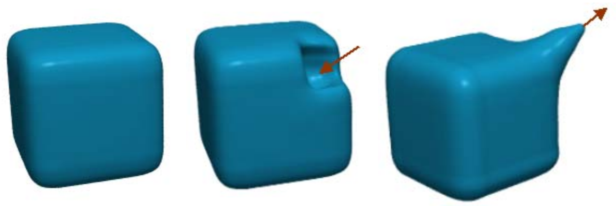
If further assuming the force vector to be a straight line, then C can be formulated as follows:

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

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,

$$f = -l \cdot \int_0^1 s(u_0 + (u_1-u_0)t, v_0 + (v_1-v_0)t, w_0 + (w_1-w_0)t)dt,$$

where $l = \sqrt{(u_1-u_0)^2+(v_1-v_0)^2+(w_1-w_0)^2}$. Figure 10 shows two simple examples for force-based deformation.



(a) (b) (c)

Figure 10: Deformation with point-based force tools, where (a) shows the original object and the arrows in (b) and (c) denote the directions of the applied forces.

In a more general case, if C is a spatial curve instead, a general curve-based tool can be defined in our system without any additional difficulty. Users can pre-define a curve and limit the force mapping only along that curve. Therefore, when users sculpt the object with the curve-based tool, the integral of forces is then along the curve force vector. The generated forces are applied on all the mass-points on the curve. For other more advanced, area-based tools, our system discretizes the area into a set of sampled (straight and/or curved) tracks, then perform integration along every track, and result in a more sophisticated deformation in any user-specified area. In a nutshell, area-based tools allow users to manipulate a set of mass-points instead of only one point. Figure 11(a) shows a curve and area based force deformation and Figure 11(b) shows a force-based joining.

If we associate the force vector with certain constraints, then more interesting tools can be created, such as chisel, squirt, bending tools and so on. Figure 12(a) shows an operation performed with a virtual chisel, which controls the tool-penetration depth at the contacting points to be a small, constant value along the normal of the tool surface. So the sculpting forces are generated by integrating only along the array of specified, short force vectors sampled over the trajectories of tool path (i.e., ”H” in this example). The generated forces result in the chisel effect along the trajectories of tool path. Figure 12(b) shows a squirt operation, where the generated sculpting forces are opposite to chisel forces along the same trajectories. More importantly, our modeling environment provides virtual tools to users so that they can perform some more inter-

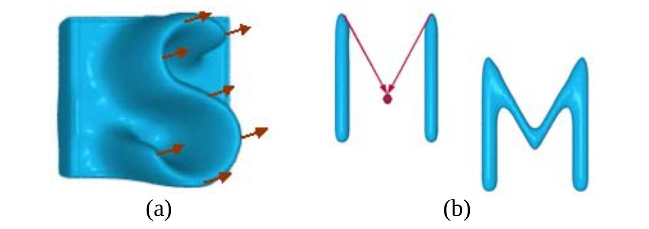

(a) (b)

Figure 11: (a) Free-form deformation with curve-based and area-based force tools (The 3D ”s” shape is extruded from a flat base by the force tools). (b) Joining with a force tool.

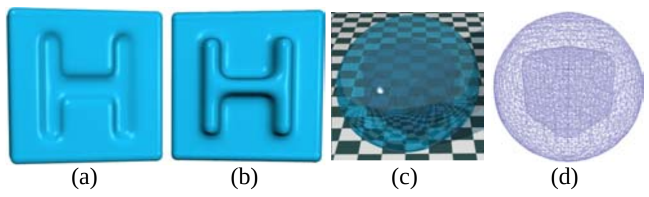

(a) (b) (c) (d)

Figure 12: (a) Chisel operation. (b) Squirt operation. (c-d) Force tools allow users to sculpt solid interior without breaking the outer material.

esting 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). Figure 12(c) and 12(d) show an example undergone sculpting only in solid interior while maintaining its surrounding, boundary material unchanged.

7.2.2 Haptic Feedback

In order to enhance the realism of virtual sculpting, our modeling environment offers force feedback, which can offer users a realistic feel of the virtual objects. Thus, users can gain a richer understanding of their sculpted model. Our work significantly extends the notion of simply touching compliant objects (i.e., haptic rendering) to interactively and directly sculpting of virtual solids (i.e., haptic modeling).

By adding external input forces based on the user’s actions, the iso-surface deforms according to the physical properties of the model. The external forces that apply to the model are generated using the technique described in Section 4.2. In order to satisfy the haptic refreshing requirement, we use Gaussian Quadrature method for the fast evaluation of the integral. Along each force vector, the continuous function is only evaluated at four sampled points to effectively calculate the integral.

Whenever a force-based tool pulls or pushes on mass-points to introduce forces along the force vector, an equal and opposite force is generated at the other end of the force vector (i.e., the user’s cursor). The force is calculated at 1000Hz and transmitted to the haptic device where the force magnitude is then converted to motor torques leading to a real force at the cursor position. When the iso-surface gradually moves toward the user’s cursor, the force decreases gradually to zero. Then the user is connected directly to the iso-surface. The gradual decreasing approach prevents high-variational jerking forces from occurring, which otherwise can potentially injure the user of the haptic device or damage the device.

7.3 Constraints

Our system can provide geometric constraints. Enforcing geometric constraints offers additional intuitive control of a shape during the physics-based design process. Constraining geometric properties of dynamic volumes can facilitate feature-centered design, which can significantly improve the system performance.

Our volumetric implicit function uses B-splines as underlying constituents, so local support can be easily accomplished. Designers 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 matrix.

The haptic device we are currently using requires 1000Hz refresh rate. This hardware limitation only permits the real-time simulation of several thousand mass-points. Therefore, local sculpting is much more attractive for the system to deal with large sculpted objects by constraining the physical simulation to occur in a local region and speeding up frame rates.

Typical geometric constraints include point, curve, and normal constraints. In our modeling environment we provide two mechanisms to implement geometric constraints.

The mathematical mechanism expresses the linear constraints as follows:

$$\mathbf{c}(\mathbf{d}) = \mathbf{L}\mathbf{d} + \mathbf{b} = 0, \tag{9}$$

where $\mathbf{L}$ is a matrix of coefficients, $\mathbf{d}$ is the generalized coordinate vector, and $\mathbf{b}$ is a constant vector. Usually (9) is an under-determined linear system. So we can eliminate those constrained variables and express $\mathbf{d}$ with the rest unconstrained variables $\mathbf{h}$ using the robust linear least-square solver of singular value decomposition:

$$\mathbf{d} = \mathbf{G}\mathbf{h} + \mathbf{h}_0. \tag{10}$$

Replacing $\mathbf{d}$ in (5) with (10), the matrices and vectors in (5) are reduced to a minimal unconstrained set of generalized coordinates:

$$\mathbf{MG}\ddot{\mathbf{h}} + \mathbf{DG}\dot{\mathbf{h}} + \mathbf{KGh} = \mathbf{f}_{\mathbf{d}} - \mathbf{Kh}_0. \tag{11}$$

Multiplying each side with $\mathbf{G}^\top$, then (11) becomes:

$$\mathbf{G}^\top\mathbf{MG}\ddot{\mathbf{h}} + \mathbf{G}^\top\mathbf{DG}\dot{\mathbf{h}} + \mathbf{G}^\top\mathbf{KGh} = \mathbf{G}^\top\mathbf{f}_{\mathbf{d}} - \mathbf{G}^\top\mathbf{Kh}_0.$$

Defining mass, damping, and stiffness matrices of the constrained dynamic system as:

$$\mathbf{M}_h = \mathbf{G}^\top\mathbf{MG}, \qquad \mathbf{D}_h = \mathbf{G}^\top\mathbf{DG},$$

$$\mathbf{K}_h = \mathbf{G}^\top\mathbf{KG}, \qquad \mathbf{c} = -\mathbf{G}^\top\mathbf{Kh}_0,$$

the discrete Lagrangian equations of the constrained dynamic system can be formulated as follows:

$$\mathbf{M}_h\ddot{\mathbf{h}} + \mathbf{D}_h\dot{\mathbf{h}} + \mathbf{K}_h\mathbf{h} = \mathbf{f}_h + \mathbf{c}. \tag{12}$$

In the constrained dynamic system, $\mathbf{h}$ has reduced size comparing with $\mathbf{d}$. Hence, $\mathbf{h} = \mathbf{A}'\mathbf{p}'$, where $\mathbf{A}'$ and $\mathbf{p}'$ are subsets of the original matrices. We can still use the numerical technique described in Section 4.3 to solve (12). This method treats those constraints as hard constraints. Figure 13 shows cross-sectional design using the point and curve constraints.

8 Experiments

Our system is implemented on a Microsoft Windows NT PC with a 550MHz CPU and 512MB RAM. A PHANToM 1.0 3D Haptic input/output device from Sensable Technologies is employed to provide a natural and realistic force feedback. The entire software system is written in Microsoft Visual C++ and the graphics rendering module is built upon OpenGL.

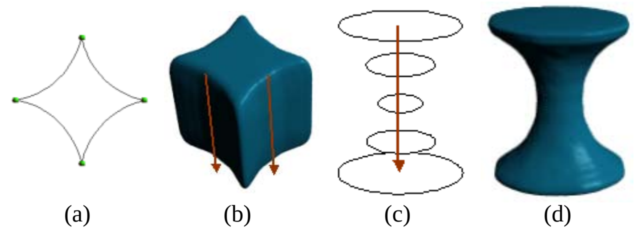


Figure 13: Illustration of cross-sectional design using point constraints and curve constraints. Green points denote on-surface points, and the specified curves must be on the iso-surface. The arrows denote the sculpting forces.

We have developed a modeling environment for haptic sculpting of dynamic volumetric models based on non-uniform scalar B-splines and physics-based modeling. The dynamic volumes can be generated with a varying number of control coefficients and sampling rates and with any user-specified physical properties. For force-based deformations, it is time critical since it affords real-time haptic feedback. In this case, sometimes we may need 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 several thousand mass-points. We have recorded the timings for physical simulations using various configurations of the control coefficients and the discretized space samples (See Table 1). This table contains the timings only for the simulation loop under the condition that the haptics loop is always performed within 1ms.

Control coefficient resolution	Mass points	Explicit time
$5 \times 5 \times 5$	$5 \times 5 \times 5$	0.5 ms
$5 \times 5 \times 5$	$10 \times 10 \times 10$	3.2 ms
$10 \times 10 \times 10$	$10 \times 10 \times 10$	22 ms
$10 \times 10 \times 10$	$15 \times 15 \times 15$	50 ms
$15 \times 15 \times 15$	$15 \times 15 \times 15$	120 ms

Table 1: Physical simulation timings using an explicit solver. The first and second columns denote how many control coefficients and mass points in the simulated region.

Within our dynamic, volumetric environment and using all the available geometric, physical, and haptic tools we have developed (and without using any other external resources), we have sculpted many interesting objects and composed them into several virtual-world scenes (See Figure 14). It only took 20 to 40 minutes to create each of them with our system. Figure 14(a) demonstrates a 3D ”VOLVIS 2002” logo. They are formed and stretched from flat panels using force-based tools. Figure 14(b) demonstrates a ”virtual coffee room”, which is composed of a pair of sofa, a glass table, and two coffee mugs. Figure 14(c) shows a ”water world”, which is composed of stones, marine plants, and lobsters. The lobsters are fitted from an existing dataset and further edited using force-based tools in our modeling environment. Figure 14(d) illustrates a ”terrain field”, which consists of trees, terrains, mountains, stones, a truck, and a jeep. The truck and jeep are fitted from the volumetric datasets using our hierarchical fitting procedure.

9 Conclusion

We have presented a novel haptics-based, dynamic, volumetric modeling environment that employs trivariate scalar non-uniform B-splines as its underlying representation. All the volumetric objects sculpted in our modeling environment are characterized by piece-wise implicit functions. We have proposed a novel approach that unifies implicit functions, parametric representations, and physics-based modeling within a single haptics-based volumetric 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 function based volumetric models 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 datasets as well as solids implicitly defined by point clouds. The powerful 3D haptics-based interface of our environment is more intuitive and natural than conventional 2D mouse-based interfaces, making it possible for our dynamic, volumetric framework to appeal to a spectrum of users ranging from highly-trained engineers, computer professionals, artists, to even naive users (with little computer skill). Our volumetric modeling system permits designers to create real-world, complicated volumetric models and scenes in real-time.

For the future work, several issues shall be addressed to improve our research. At present, our system can only simulate several thousand mass points in real-time. It is difficult and far from trivial to perform global deformation on large datasets. A multi-resolution model is more desirable to achieve better resolution control when dealing with global deformation on large models.

Acknowledgements

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

AVILA, R. S., AND SOBIERAJSKI, L. M. 1996. A haptic interaction method for volume visualization. In *Proceedings of the 7th IEEE Visualization ’96*, 197–204.

BLINN, J. F. 1982. Generalization of algebraic surface drawing. *ACM Trans. on Graphics 1*, 3, 235–256.

BLOOMENTHAL, J., AND WYVILL, B. 1990. Interactive techniques for implicit modeling. *Computer Graphics 24*, 2, 109–116.

BLOOMENTHAL, J. 1997. *Introduction to implicit surfaces*. Morgan Kaufmann. Edited by J. Bloomenthal with C. Bajaj, J. Blinn, etc.

CANI, M. P., AND DESBRUN, M. 1997. Animation of deformable models using implicit surfaces. *IEEE Trans. on Visualization and Computer Graphics 3*, 1, 39–50.

CARR, J. C., BEATSON, R. K., AND CHERRIE, J. B. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH’01*, 67–76.

DACHILLE, F., QIN, H., AND KAUFMAN, A. E. 2001. A novel haptics-based interface and sculpting system for physics-based geometric design. *Computer-Aided Design 33*, 5, 403–420.

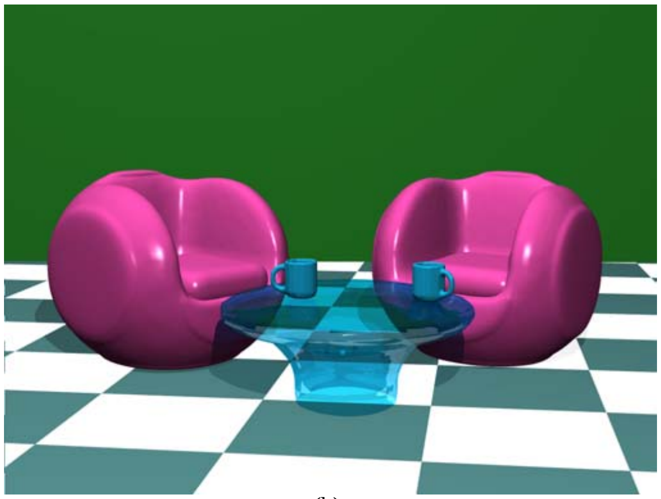
FERLEY, E., CANI, M. P., AND GASCUEL, J.-D. 2000. Practical volumetric sculpting. *The Visual Computer 16*, 7, 469–480.

GALYEAN, T. A., AND HUGHES, J. F. 1991. Sculpting: An interactive volumetric modeling technique. *Computer Graphics 25*, 4, 267–274.

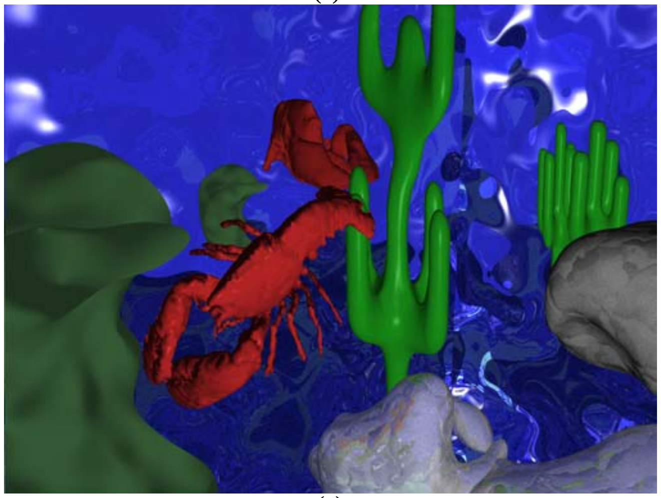
HUA, J., AND QIN, H. 2001. Haptic sculpting of volumetric implicit functions. In *Proceedings of 9th Pacific Conference on Computer Graphics and Applications*, 254–264.

VOL
VIS
2002

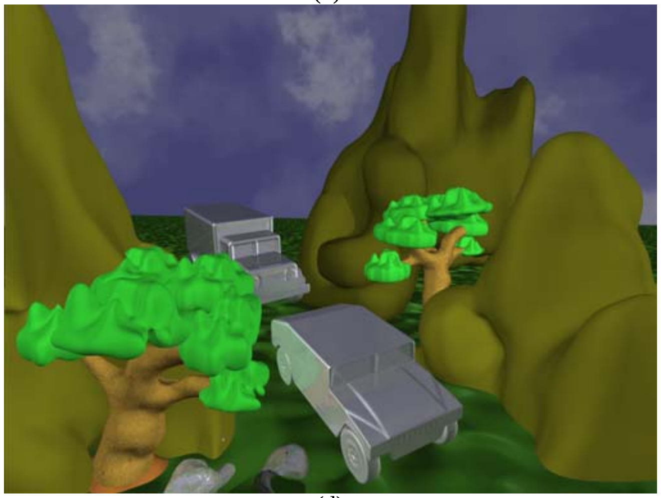
(a)



(b)



(c)



(d)

Figure 14: Several sculptures and scenes created entirely with our environment and finally rendered with ray tracing.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4, 163–169.

MCDONNELL, K. T., QIN, H., AND WLODARCZYK, R. A. 2001. Virtual clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 179–190.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. *Computer Graphics 26*, 2, 309–312.

MURAKI, S. 1991. Volumetric shape description of range data using blobby model. *Computer Graphics 25*, 4, 227–235.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics 23*, 3, 215–222.

PERRY, R. N., AND FRISKEN, S. F. 2001. Kizamu: a system for sculpting digital characters. In *SIGGRAPH'01*, ACM Press, 47–56.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A physics-based framework for geometric design. *IEEE Trans. on Visualization and Computer Graphics 2*, 1, 85–96.

RAVIV, A., AND ELBER, G. 1999. Three dimensioinal freeform sculpting via zero sets of scalar trivariate functions. In *Proceedings of 5th ACM Symposium on Solid Modeling and Applications*, 246–257.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics 21*, 4, 205–214.

THOMPSON, T. V., JOHNSON, D. E., , AND COHEN, E. 1997. Direct haptic rendering of sculptured models. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 167–176.

TURK, G., AND O'BRIEN, J. F. 1999. Shape transformation using variational implicit surface. In *SIGGRAPH'99*, 335–342.

WANG, S. W., AND KAUFMAN, A. E. 1995. Volume sculpting. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, 151–156.

WYVILL, G., MCPHEETERS, C., AND WYVILL, B. 1988. Data structure for soft objects. *The Visual Computer 2*, 4, 227–234.

ZHAO, H., OSHER, S., AND FEDKIW, R. 2001. Fast surface reconstruction and deformation using the level set method. In *Proc. Of the IEEE Workshop on Variational and Level Set Methods in Computer Vision*.