Dynamic Implicit Solids with Constraints for Haptic Sculpting

Jing Hua Hong Qin
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400, U.S.A.
Email: {jinghua|qin}@cs.sunysb.edu

Abstract

In this paper we present a novel, interactive shape modeling technique: Dynamic Implicit Solid Modeling, which unifies volumetric implicit functions and powerful physics-based modeling. Although implicit functions are extremely powerful in graphics, geometric design, and shape modeling, the full potential of implicit functions is yet to be fully realized due to the lack of flexible and interactive design techniques. In order to broaden the accessibility of implicit functions in geometric modeling, we marry the implicit solids, which are semi-algebraic sets of volumetric implicit functions, with the principle of physics-based models and formulate dynamic implicit solids. By using "density springs" to connect the scalar values of implicit functions, we offer a viable solution to introduce the elasticity into implicit representations. As a result, our dynamic implicit solids respond to sculpting forces in a natural and predictive manner. The geometric and physical behaviors are tightly coupled in our modeling system. The flexibility of our modeling technique allows users to easily modify the geometry and topology of sculpted objects, while the inherent physical properties can provide a natural interface for direct, force-based free-from deformation. The additional constraints provide users more control on the dynamic implicit solids. We have developed a sculpting system equipped with a large variety of physics-based toolkits and an intuitive haptic interface to facilitate the direct, natural editing of implicit functions in real-time. Our experiments demonstrate many attractive advantages of our dynamic approach for implicit modeling such as intuitive control, direct manipulation, real-time haptic feedback, and capability to model complicated geometry and arbitrary topology.

1. Introduction and Motivation

The efficiency and flexibility of shape modeling are vital to the success of graphics, geometric design, and virtual environments. Despite the prevalence of parametric forms in visual computing fields, the traditional representation of geometric entities such as commonly-used analytic shapes comes from implicit functions because of many of their attractive properties [1]. It has been proved that every rational parametric curve/surface/solid can be represented by an implicit algebraic equation, but not vice versa. In contrast with parametric forms, implicit functions have a number of modeling advantages such as point classification, intersection computation, and unbounded geometry.

Nevertheless, existing techniques associated with implicit functions have certain severe shortcomings. First, effectively digitizing and rendering an implicit function is oftentimes far from trivial. It is extremely difficult to control the shape of implicit solids while re-rendering the modified regions fast enough for their use within an interactive environment. Second, a designer often has no intuitive and quantified understanding on the effect of altering polynomial coefficients or adding/deleting components. The coefficients provide neither direct, natural geometric interpretation nor intuitive insight into the underlying shape. Third, there are no convenient tools for the intuitive shape control of this type of algebraic solids. Moreover, general implicit functions usually have a property of global control. Flexible and direct modeling techniques for implicit solids remain under-explored in general.

Physics-based modeling techniques can alleviate many of these problems by augmenting geometric objects with physical attributes such as mass, damping and stiffness distributions. For implicit functions, however, since their geometry is generated indirectly from the zero-set of their function evaluation, we cannot directly associate physics with the underlying zero-set because of the time-varying nature for the zero-set geometry. Note that, this is perhaps the most difficulty matter that prevents the integration of physics-based modeling and implicit representations. We propose a feasible technique to overcome this difficulty. Therefore, geometric parameters can be hidden from users through the use of natural, force-based interfaces that facilitate direct manipulation of solid objects. Our system synchronizes the geometric and physical representations of objects in order to maintain the underlying geometric structures of the sculpted solids. Such dynamic and interactive approaches can provide users with a natural, force-based interface and geometric interface at the same time. Physics-based modeling does not attempt to replace existing geometry-based interfaces but rather to augment them with additional flexibilities.

We also develop a haptic interface and provide a suite of haptic sculpting tools in our system, in order to further improve physics-based modeling techniques. This is because both haptics and dynamic models depend on real-world physical laws to govern the interaction of dynamic

objects and the realistic simulation. We observe that the integration of haptic interface and dynamic implicit models can maximize the potential offered by both physics-based modeling and implicit functions. Haptics provides users a hand-based mechanism for intuitive, manual interactions within virtual environments towards realistic tactile exploration and manipulation. With a standard haptic device, our approach permits users to interactively sculpt virtual materials having realistic properties and feel the physically realistic presence with force feedback throughout the design process. Force feedback provides additional sensory cues to designers. This tactile exploration can afford designers to gain a richer understanding of the 3D nature.

2. Research Contribution

In this paper we integrate volumetric implicit functions and powerful physics-based modeling into one single framework: *Dynamic Implicit Solid Modeling*, which permits interactive and direct manipulation of implicit solids in real-time. Further enhancing our modeling framework with a haptic interface makes the framework more powerful and realizes more potential of dynamic implicit functions.

In our system, the sculpted object is evaluated as a level-set of a volumetric implicit function defined over a three-dimensional working space. Our volumetric implicit functions [9] integrate implicit functions with parametric representations. It should be noted that in [9] there are neither material quantities nor dynamic behaviors in the system. The direct manipulation was achieved by solving a static linear system. The haptic simulation was derived directly from the geometric representation, and it is yet to reflect the true physical feeling when interacting with real deformable material. We now propose a novel technique to associate physics with implicit representations. Physical attributes are assigned inside the working space. In our framework the inherent control coefficients of the implicit functions dictate the shape geometry, while the physical attributes govern the dynamic behavior and facilitate the direct manipulation. Rather than modifying the coefficients associated with the volumetric implicit function as exhibited in [5], our sculpting tools support the direct editing of implicit functions' scalar values. Our algorithms can automatically determine all of the unknown control coefficients and effectively reconstruct a new volumetric implicit function after the local/global free-form deformation. The additional constraints allow users to gain more sophisticated control over the dynamic models. Our system offers a wider array of intuitive sculpting tools (especially the haptics-based tools) responsible for the effective construction of various complicated geometric shapes with diverse topologies.

To the authors' best knowledge, there is little work about integrating physics-based modeling with volumetric implicit functions and applying physics-based or haptic tools on density-based volumetric datasets due to the aforementioned difficulties. Our work aims to incorporate the elasticity into implicit functions and advance the state of the knowledge in the effective integration of implicit functions, physics-based modeling, and haptic sculpting.

3. Background Review

3.1 Implicit Functions

Blinn [3] demonstrated that implicit functions are well suited for both scientific visualization and the modeling tasks in computer graphics. Hoffmann [13] systematically reviewed the implicit function techniques including implicitization, parameterization, and parametric/implicit conversion in CAGD. Bajaj and Ihm [14] presented an efficient algorithm to implement Hermite interpolation of low-degree algebraic surfaces with C^1 or G^1 continuity. Note that, neither point nor curve interpolation is an attractive mechanism for defining an implicit surface because it is difficult for designers to predict the surface behavior beyond interpolating curves and points. In order to create the implicit surfaces easily and gain more control on them, Bloomenthal *et al*. [1][2] used skeleton methods to construct implicit surfaces and Hart *et al*. [10] proposed a method of finding the critical points of implicit surfaces.

Implicit functions can also be used to represent a solid. Commonly-used techniques includes Boolean operations and functional compositions. The common feature essential to all implicit solid modeling methods [15] is the creation of an oriented three-dimensional boundary surface which partitions the entire 3D space into two distinct regions, namely the one occupied by the solid interior and the one outside of the defined solid. Recently, Raviv and Elber [5] presented a 3D interactive sculpting paradigm that employed a set of scalar uniform trivariate B-spline functions as underlying representation. Users can indirectly sculpt objects to a desirable shape by directly modifying relevant scalar control coefficients of the underlying functions with geometric tools. In the past, the representation of sculpted objects might be of discrete type, e.g., [4]. Raviv and Elber [5] pioneered the use of a continuous characteristic function for volume sculpting.

3.2 Physics-based Modeling

Conventional geometric modeling may be inconvenient for representing complicated solids, because modelers are faced with the tedium of indirect shape modification and refinement through time-consuming operations on a large number of control vertices. 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, damping 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*. [18] and further developed by Pentland and Williams [19], and Metaxas and Terzopoulos [20]. Qin and Terzopoulos introduced D-NURBS surfaces, an extension to traditional NURBS that permits more natural control of the surface geometry [12].

3.3 Haptic Interface

Haptic rendering is a process of applying forces through the use of force-feedback devices and augmenting a virtual environment with haptic interaction. Thompson *et al*. [6] derived efficient intersection techniques that can be applied to nearly any type of haptic interfaces. Dachille *et al*. [16] developed a haptic interface to permit the direct manipulation of dynamic B-spline surfaces. McDonnell *et al*. [7] employed haptic toolkits to explore the dynamic subdivision solids. Hua *et al*. [9] presented a haptic interface for volume sculpting. Avila *et al*. [8] presented a haptic interaction that is suitable for both volume visualization and modeling applications. Despite the widespread utilizations 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 implicit solids and employ force-based, haptic tools to directly work on density-centered volumetric datasets.

4. Dynamic Volumetric Implicit Functions

4.1 Spline-based Volumetric Implicit Functions

Throughout this paper, we utilize scalar trivariate B-spline functions as the underlying shape primitives for object representation. The use of implicit B-spline functions for solid modeling is strongly inspired by their attractive properties including simplicity, generality, local control, etc. The generic B-spline functions are of the following form:

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

where $s(u,v,w)$ represents the scalar value at position (*u, v, w*) in a parametric domain. *u*, *v*, *w* change from 0 to *U*, *V*, *W*, which represent the size of sampling points along three dimensions of the parametric domain. p_{ijk} are the scalar control coefficients with the domain of *I*, *J*, *K* that are [*0, l-1*], [*0, m-1*], and [*0, n-1*], respectively. In addition, $B_{i,r}(u)$, $C_{j,s}(v)$ and $D_{k,t}(w)$ are the basis functions corresponding to p_{ijk}, evaluated at (*u, v, w*). The degrees of the three basis functions are *r-1*, *s-1*, and *t-1*, respectively. To simplify the mathematical notation, (1) can also be expressed as the following matrix form:

$$\mathbf{d}=(\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{D})\mathbf{p}, \quad (2)$$

where $\otimes$ denotes Kronecker Product, and

$\mathbf{d}=[\cdots,s_{ijk},\cdots]^T (i\in[0,U], j\in[0,V], k\in[0,W])$,

$\mathbf{p}=[\cdots,p_{ijk},\cdots]^T (i\in[0,l-1], j\in[0,m-1], k\in[0,n-1])$,

B, **C**, and **D** are matrices composed of the sampling of basis functions. $(\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{D})$ could be precomputed in order to save runtime computation and improve real-time performance. To simplify the mathematical derivation in the rest of the paper, we use **A** to represent $(\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{D})$. Collecting all the level-sets whose return values are greater (or smaller) than a given threshold, we could define an implicit solid:

$$\begin{cases} w=F(x,y,z) \\ w>w_0 \end{cases}. \quad (3)$$

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 commonly-used parametric B-splines, implicit B-spline functions formulate the scalar value distribution in a 3D space where implicit solids are uniquely defined as semi-algebraic point sets. In our system, we enhance the scalar B-spline representation power by incorporating the modeling advantages from hierarchical splines, generalized CSG-based Boolean operations, and non-uniform 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 number of control coefficients in order to achieve the goal of multiresolution analysis and level-of-details control. Then the scalar value at the location (*x, y, z*) can be computed as

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

where $\mathbf{T}_i$ is an affine transformation from the Euclidian space to the parametric domain of patch s_i. Since the trivariate B-spline has the affine invariance property, this transformation can be easily implemented. For each 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, (4) 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.

Users may also intend to sculpt implicit solids to form sharp features over their boundaries or modify the continuity requirements across their smooth boundaries. In light of this demand from users, 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 readily available in our system through the use of

$$F(x,y,z) = \bigOmega_{i=1}^{N} s_i(\mathbf{T}_i(x,y,z)), \qquad (5)$$

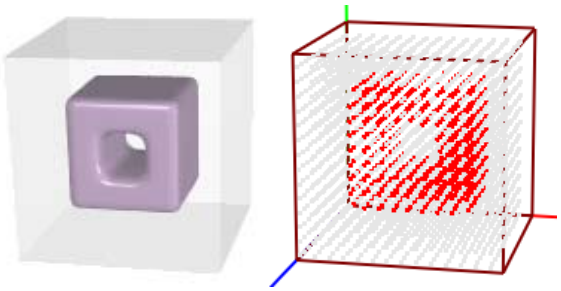
where Ω is a Boolean operation such as Union, Intersection, or Difference. In addition, $\mathbf{T}_i$ and s_i have the same geometric meaning as those appeared in (4). In our system, the Boolean operation information will be stored in a tree structure in order to speedup the data query.

Our system allows users to specify a non-uniform vector during the initialization phase of the object design session. In addition, users could insert more knots anywhere into current knot vector at any time 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 represented by volumetric implicit functions is essentially a non-uniform scalar B-spline.

Through different combinations of these three techniques, our system could offer users a large array of modeling operations and enhance the already-powerful shape variation of implicit B-splines with the additional flexibility in a hierarchical fashion.

4.2 Dynamic Implicit Solids

In order to introduce physics into our system, 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 4.1 is employed to assign the density value to the sample points to indicate if there is material at that location. The function will be used to formulate the density distribution over the 3D working space and represent the sculpted object by a given level-set. Fig. 1 shows a simple sculpted object and its corresponding voxelmap in a 3D space. This voxelmap defines a function, where the solid particles (colored in dark) 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.



(a) (b)

Fig. 1. (a) A simple sculpted object. (b) Its corresponding voxelmap.

In the discretized working space, we can use

$$\mathbf{d} = \mathbf{A}\mathbf{p} \qquad (6)$$

to formulate the density values associated with the sampling points in a patch. **A** is a sparse basis matrix that contains weights given by our spline-based volumetric implicit functions.

The discretized dynamic implicit solid has 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 over the solid model in real-time. The discretized solid is modeled as a collection of mass-points connected by a network of springs across nearest neighbors. Here we use a mass-spring model because of its simplicity and the critical need of real-time haptic volume sculpting. Since we employ volumetric implicit functions as the solid representation, we shall consider our models in 4D space rather than 3D space. Fig. 2 shows the mass-spring network in the vicinity of a mass point.

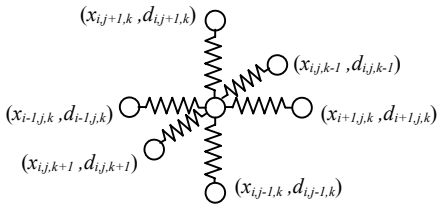


Fig. 2. The mass-spring network in the vicinity of a point $P(x_{i,j,k}, d_{i,j,k})$, where $x_{i,j,k}$ represents the position of a mass point $m_{i,j,k}$ in the 3D space, $d_{i,j,k}$ is the density at that position.

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. Our special springs do not change the geometric position $x_{i,j,k}$ of the mass $m_{i,j,k}$ at all. However, they permit the magnitude change of the density $d_{i,j,k}$ of the mass $m_{i,j,k}$, which is the vector component of the forth dimension. Essentially, this new type of springs will attract density values of neighbors. When users manipulate the dynamic 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 scalar B-splines. 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. We will explain this new technique and its system implementation in details in the following and Section 5.2.

The motion equation of all mass-points is formulated as a discrete simulation of Lagrangian dynamics:

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{D}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f}_{\mathbf{d}}, \qquad (7)$$

where **M** is a mass matrix, **D** is a damping matrix, **K** is a stiffness matrix, and the force at every mass-point in the solid is the summation of all possible external forces: $\mathbf{f}_{\mathbf{d}} = \sum \mathbf{f}_{\mathbf{ext}}$. 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 the system the geometric positions of mass-points do not change 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 desired.

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 $\mathbf{A}^T$ and substituting **d** with **Ap**, we obtain:

$$\mathbf{A}^T\mathbf{M}\mathbf{A}\ddot{\mathbf{p}} + \mathbf{A}^T\mathbf{D}\mathbf{A}\dot{\mathbf{p}} + \mathbf{A}^T\mathbf{K}\mathbf{A}\mathbf{p} = \mathbf{A}^T\mathbf{f_d}. \quad (8)$$

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

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

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

$$\ddot{\mathbf{p}} = (\mathbf{A}^T\mathbf{M}\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{f_d} - \mathbf{A}^T\mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^T\mathbf{K}\mathbf{d}), \quad (9)$$

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

$$\begin{aligned} \dot{\mathbf{p}}_{i+1} &= \dot{\mathbf{p}}_i + \ddot{\mathbf{p}}_i \Delta t \\ \mathbf{p}_{i+1} &= \mathbf{p}_i + \dot{\mathbf{p}}_i \Delta t \end{aligned}. \quad (10)$$

The updated control coefficients $\mathbf{p}_{i+1}$ are further used to update the discretized model defined by $\mathbf{d}_{i+1} = \mathbf{A}\mathbf{p}_{i+1}$. As a result, the new dynamic approach can continuously evolve the implicit functions, and therefore permit users to directly work on both the level-set geometry and the enclosed material distribution with a continuous visual feedback. Although the more robust, implicit Euler solver is readily available in our system, we employ a simpler, forward method for the purpose of real-time, haptic sculpting.

5. Interactive Techniques

Our system provides two primary types of sculpting tools. One type of tools is called "geometric tools", and the other one is called "force-based tools". Whenever a sculpting tool is used to sculpt the object, the density values of the working space at some regions will be modified correspondingly. Then the system will reconstruct the volumetric implicit function to represent the new, modified object undergoing deformation. By using local Marching Cubes technique [11][17], the iso-surface of the object could be displayed interactively.

By integrating physics-based modeling with a haptic interface, our force-based tools allow users to reach toward an object, feel the physical presence of its shape, and sculpt free-form solids with force feedback. Through the use of many haptic tools available in our system, users can obtain both intuitive feeling and better understanding of the virtual material. The feedback forces are computed based on the object geometry and the associated physical properties.

Since the sculpted object is discretized in a voxel raster, usually there are many homogeneously empty regions outside the object of interest. If those regions could be quickly separated from the sculpting region, it 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, which can locate where the modification is performed and only locally updates the volumetric implicit function for efficiency purpose. Our system uses Marching Cubes technique to render the iso-surface of the sculpted object. The local update property can speed up the Marching Cubes rendering by only conducting the re-evaluation task on the modified parts.

5.1 Geometric Tools

5.1.1 Geometric Tool Modeling

Tools are represented by any 3D implicit function $w_0 = G(x, y, z)$. It is easy to determine whether a location is inside the tool volume by simply evaluating the function. In order to prevent object spatial aliasing, a filtering operation must be used inside the tool volume. The filtering algorithm used in our system is similar to the one in [4]. Given a location (*x, y, z*), the shortest distance from (*x, y, z*) to the boundary of the tools is computed using the evaluation function. Then this shortest distance is used to filter the density values at the location (*x, y, z*). Here we use a linear filter in the interest of simplicity. The minimal density value is assigned to the boundary and the maximal one is assigned to the center of the tool. The density values at the intermediate locations are linearly interpolated.

5.1.2 Tool-Object Interaction

When users assign a sculpting tool to a new location, the tool is mapped to the coordinate system, which contains the sculpted object. The bounding box of the tool is then computed. The density values inside the tool volume are modified as described in Section 5.1.1. If the tool is to add material, those density values should be greater than the object iso-value. If the tool is to remove material, those density values should be less than the iso-value. After this initial modification on material distribution, we have to reconstruct the volumetric implicit function of B-splines according to the new density distribution. The mathematics of the above manipulation is formulated as follows:

$$\mathbf{A}\mathbf{p}_{new} = \mathbf{d}_{new}, \quad (11)$$

where $\mathbf{d}_{new}$ represents the new density distribution over the sculpted patch region and $\mathbf{p}_{new}$ are new control

coefficients. Because of the local support property of B-splines, only a very small subset of the control coefficients needs to be modified. Hence, we only need to solve this system of linear equations within the tool-sculpting region. Therefore, (9) can be further simplified into:

$$\mathbf{A}'\mathbf{p}_{\text{mod}} = \mathbf{d}_{\text{mod}}, \tag{12}$$

where $\mathbf{d}_{\text{mod}}$ and $\mathbf{p}_{\text{mod}}$ only come from the modified region. $\mathbf{A}'$ is a small subset of the original basis matrix, which is corresponding to the local modified region.

By using the least-square fitting, $\mathbf{p}_{\text{mod}}$ could be derived as follows:

$$\mathbf{p}_{\text{mod}} = \left[\mathbf{A}'^{T}\,\mathbf{A}'\right]^{-1}\mathbf{A}'^{T}\,\mathbf{d}_{\text{mod}}. \tag{13}$$

After the new control coefficients are generated, the system uses the Local Marching Cubes algorithm to render the modified part in order to generate the new iso-surface of the sculpted object.

Demonstrated in Section 5.1.1, our geometric tools could be easily defined using any implicit functions. Therefore, using the geometric tools users can create objects of complicated geometry and arbitrary topology with ease. Fig. 3 shows a number of sculpted examples using our geometric toolkits including *sphere-based carving and addition tools, cylinder-based carving and addition tools, rectangle-based carving and addition tools, torus-based carving and addition tools,* etc.

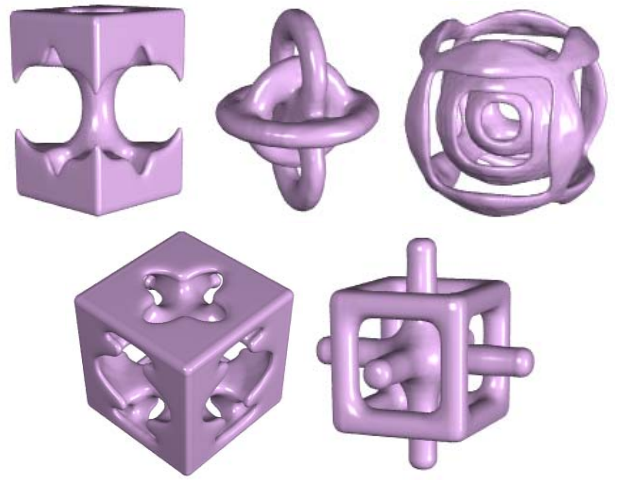


Fig. 3. Sculpted examples using geometric tools.

5.2 Force-based Tools

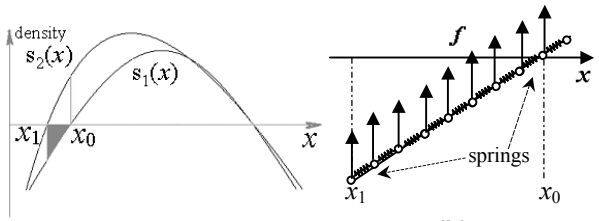
Although geometric tools are powerful to create complicated volumetric shapes, they constitute a purely geometric representation. To perform free-form deformation, modelers have to design complicated tools based on popular function primitives. This indirect shape modification and refinement are time-consuming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

5.2.1 Free-form Deformation via Forces

In order to directly deform the implicit solids via force-based physical manner, we must address the important issue of force mapping. 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. Therefore, any force mapping algorithm should be meaningful and suitable for both the dynamic simulation and the haptic interaction.

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.

Let us consider a point-based force tool first. 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 space can be trivially generalized. For arbitrary one-dimensional implicit function the zero-set is just a set of points. As shown in Fig. 4, suppose 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. 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.



(a) **(b)**

Fig. 4. (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 system the sculpting force is calculated directly from the continuous representation by performing integration from the starting point to the ending point along the direction set 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 $\int_{x_0}^{x_1} s_1(x)dx < 0$ in this example, the minus sign outside the integral operator makes the force positive, matching the case shown in Fig. 4. In our system we define the following conventions to keep 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 force calculation formula. In Fig. 4, suppose that the user intends to move x_1 to x_0 instead, then the force calculation will be

$$f = -\int_{x_1}^{x_0} s_2(x)dx .$$

Obviously, 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 domain,

$$f = -\int_C s(u,v,w)dc , \qquad (14)$$

where C is the force vector, $s(u, v, w)$ is the density distribution function in the 3D working space. Using the parametric form $(u(t), v(t), w(t))$ to represent C, then we have

$$f = -\int_{t_0}^{t_1} s(u(t),v(t),w(t))\sqrt{\dot{u}^2(t)+\dot{v}^2(t)+\dot{w}^2(t)}dt . \qquad (15)$$

If assuming the force vector as 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,

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

where $l = \sqrt{(u_1 - u_0)^2 + (v_1 - v_0)^2 + (w_1 - w_0)^2}$.

In a more general case, if C is a spatial curve instead, a general curve-based tool will be readily available 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 along the curve force vector. The generated forces are applied on all the mass-points sitting on the curve. For the more advanced, area-based tools, our system can discretize 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. Fig. 5 shows several examples for force-based deformation.

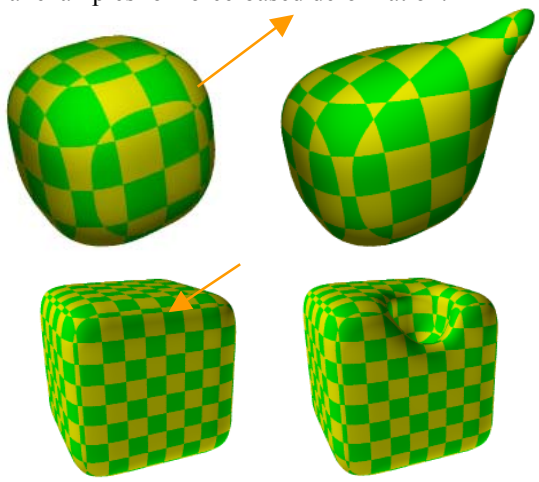


Fig. 5. Free-form deformation with force-based tools, where the arrows denote the directions of the applied forces.

5.2.2 Force Feedback

In order to enhance the realism of the virtual sculpting, our system offers force feedback, which can give 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).

From the standpoint of volume sculpting, the following problems must be addressed in order to provide meaningful force feedback for haptic interaction:

a) Force computational rate: the computational rate must be high and latency must be low in order to offer users a realistic illusion in the virtual environment.

b) Generation of contacting forces: this creates the "feel" of an object. Contacting forces can reflect the user's feeling about the stiffness of the object, its damping distribution, and other material properties. Therefore, the accurate computation of contacting forces is vital.

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 5.2.1. In order to satisfy the haptic refresh requirement, we use Gaussian Quadrature method for the fast evaluation on the integral. Along each force vector, the continuous function is only evaluated at four sampled points to effectively calculate the integral.

Whenever the 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 property prevents high-variational jerking forces from occurring, which otherwise can potentially injure the user of the haptic device or damage the device.

5.3 Enforcing Additional Constraints

Enforcing constraints offers additional intuitive control of a shape during the physics-based design process. Constraining geometric and physical properties of dynamic implicit solids can facilitate feature-centered design, which can significantly improve the system performance. Typical geometric constraints include point, curve, and normal constraints. In order to achieve the real-time performance in our haptic sculpting system, all the constraints are implemented as additional constraint springs, which transform constraints to external constraint forces and then add them to models.

5.3.1 Point Constraints

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 desired density value. If setting $d' = 0$, 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.

5.3.2 Curve Constraints

Curve constraints are implemented based on the same technique as used in point constraints. A user can specify a curve using a parametric form or an implicit form. Alternatively, the user can sketch it with a free hand. Then the curve is discretized to a set of points with the total number of N. Each spring is connecting between (x_i, y_i, z_i, d_i) and (x_i, y_i, z_i, d_i'), where i ranges from 0 to N. If setting $d_i' = 0$, then the iso-surface interpolates the curve network. Fig. 6 shows an example subject to the point and curve constraints.

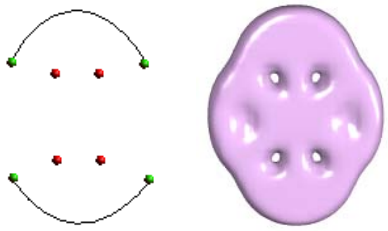


Fig. 6. Illustration of point constraints and curve constraints during a design process. Dark color points indicate off-surface points, light color points denote on-surface points, and the two specified curves must be on the iso-surface.

5.3.3 Sculpting Localized Regions

Since our volumetric implicit function uses B-splines as underlying constituents, 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.

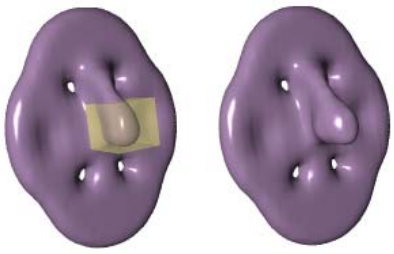
The haptic device we are currently using requires 1000Hz refresh rate. This hardware limitation only permits the real-time simulation of a few thousand mass-points. Therefore, local sculpting is much more attractive for the system to deal with large sculpted objects by constraining the physical simulation to occur in a local region and speeding up frame rates. Fig. 7(a) shows a localized region with a semi-transparent box, which limits physical operations within its boundary.

5.3.4 Physical Property Constraints

Through the use of physical property constraints, users can locally modify the mass distribution, spring stiffness, or rest length for springs. Mass modification allows users to control how quickly a 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.

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. Therefore, the normal deformation forces will have less effect on the density distribution of the high stiffness region.

Rest-length modification can make a specified region deflate or inflate. With the decrease of the rest length, all the springs inside the region generates recovery forces, which results in the effect of solid shrinking. To produce the inflation effect, the rest length is increased instead. Fig. 7(b) shows an inflation operation within the localized region.



(a) (b)

Fig. 7. Inflation operation performed inside the bounding box around the nose of a mask.

6. Implementation

Our system is implemented on a Microsoft Windows NT PC with a 550MHz CPU and 512MB RAM. A

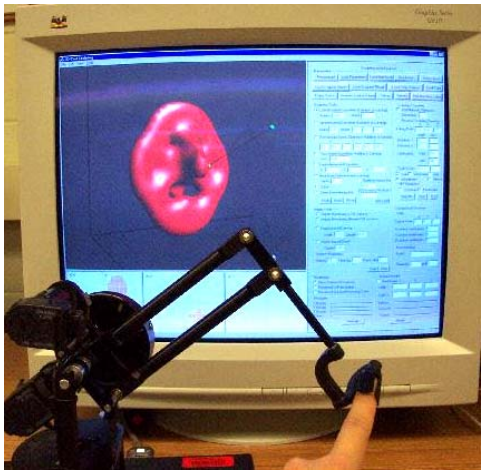


Fig. 8. User interface.

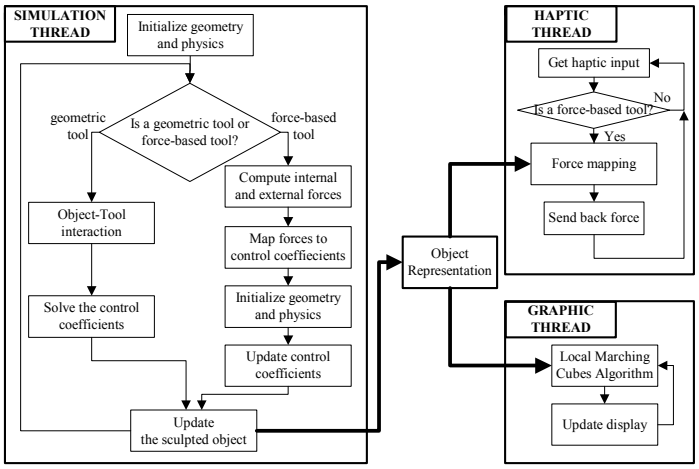


Fig. 9. The structure of multithreads.

PHANToM 1.0 3D Haptic input/output device from Sensable Technologies is employed to provide a natural and realistic force feedback. The entire system is written in Microsoft Visual C++ and the graphics rendering module is built upon OpenGL. Fig. 8 shows the system interface.

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, should the high-end multi-processor environment is available. Therefore, our system can be readily extended to many different configurations. Fig. 9 shows the structure of the multithreads, where thick arrows represent data flow and thin arrows represent control flow.

The haptic 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. If the update rate were below the threshold of 1KHz, users would have an uncomfortable feeling. In our system, the haptic thread has the highest priority.

The simulation loop is implemented in another thread. It controls the physical simulation. In order to keep up with the frame rate, the physical simulation is limited to a small region by using the techniques described in Section 5.3.3. Usually users' design intention and their sculpting operations would not exceed this limited region during one design cycle. In order to keep the system more stable we employ a simple adaptive method to adjust the simulation time-step. Essentially, if $\ddot{p}_i$ 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 volumetric objects. The rendering task makes use of local Marching Cubes algorithm and only updates the very small region in order to achieve interactive rendering rate and make graphics display consistent with sculpting operations, physical simulation and force feedback.

7. Results

We have developed a modeling system for haptic sculpting of dynamic volumetric implicit functions based on non-uniform scalar B-splines and physics-based modeling. The dynamic implicit solids can be generated with a varying number of control coefficients and sampling rates and with any user-specified physical properties. We have conducted a large number of experiments and documented the running time for the sculpting of dynamic implicit solids. The experiments are based on a working space sampled at $128\times128\times128$. The geometric tool size is given as the number of data points that the tool affects. The results are detailed in Table 1.

Control coefficient resolution	Tool size	Update time (ms)
$64\times64\times64$	$10\times10\times10$	1.5
	$20\times20\times20$	11
	$40\times40\times40$	92
$128\times128\times128$	$10\times10\times10$	2
	$20\times20\times20$	20
	$40\times40\times40$	151

Table 1: Run time of object interaction with geometric Tools.

Control coefficient resolution	Sampling points	Explicit time (ms)
$5\times5\times5$	$5\times5\times5$	0.5
	$10\times10\times10$	3.2
$10\times10\times10$	$10\times10\times10$	22
	$15\times15\times15$	50
$15\times15\times15$	$15\times15\times15$	120

Table 2: Physical simulation timings using an explicit solver.

We have also examined the timings achieved for physical simulations using various configurations of the control coefficients and the discretized space samples (see Table 2).

Within our dynamic implicit modeling framework and without using any other external resources, we have created several interesting objects and scenes from scratch as shown in Fig. 10 (see color plates).

8. Conclusion

We have presented a novel haptics-based dynamic solid modeling framework that employs trivariate scalar non-uniform B-splines as underlying representation. All the volumetric objects sculpted in our modeling system are characterized by piece-wise implicit functions. 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 proposed force mapping technique can be easily extended to any other haptic sculpting applications without additional difficulties. More importantly, our physics-based force tools can be directly employed to act on density-based volumetric datasets. We have also incorporated three popular modeling techniques: hierarchical B-splines, CSG-based functional composition, and knot insertion into our framework, making our dynamic implicit solid modeling techniques even more powerful and flexible to handle both complicated geometry and arbitrary topologies.

Our experiments have demonstrated that our modeling framework and direct editing techniques can not only overcome the existing disadvantages associated with conventional modeling of implicit functions, but also realize all the potentials exhibited in both implicit functions and physics-based modeling in visual computing fields. The powerful 3D haptics-based interface of our system is more intuitive and natural than 2D mouse-based interfaces, making it possible for our dynamic implicit solid modeling framework to appeal to a spectrum of users ranging from highly-trained engineering designers, computer professionals, artists, to even computer illiterates. Our sculpting system permits designers to create real-world, complicated models in real-time.

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, Honda Initiation Grant, and Alfred P. Sloan Fellowship.

References

[1] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, Vol. 24, No. 2, pp 109-116, March 1990.

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

[3] J. F. Blinn. Generalization of algebraic surface drawing. *ACM Trans. On Graphics*, Vol. 1, No. 3, pp 235-256, July 1982.

[4] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. *Computer Graphics*, Vol. 25, No. 4, pp 267-274, July 1991.

[5] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proc. of 5th ACM Symposium on Solid Modeling and Applications*, pp 246-257, 1999.

[6] T. V. Thompson, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Proc. of the 1997 Symposium on Interactive 3D Graphics*, pp 167-176, 1997.

[7] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk. Virtual Clay: A real-time sculpting system with Haptic Toolkits. In *Proc. of the 2001 Symposium on Interactive 3D Graphics*, pp 179-190, 2001.

[8] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. In *Proc. of the 7th IEEE Visualization '96*, pp 197-204, 1996.

[9] J. Hua and H. Qin. Haptic sculpting of volumetric implicit functions. In *Proc. of Ninth Pacific Conference on Computer Graphics and Applications*, pp 254-264, 2001.

[10] J. C. Hart, A. Durr, and D. Harsh. Critical points of polynomial meatballs. In *Proc. Implicit Surfaces 98, Eurographics/SIGGRAPH Workshop*, pp 69-76, June 1998.

[11] 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, July 1987.

[12] H. Qin and D. Terzopoulos. D-NURBS: a physics-based framework for geometric design. *IEEE Trans. on Visualization and Computer Graphics*, Vol. 2, No. 1, pp 85-96, Mar. 1996.

[13] C. Hoffmann. Implicit curves and surfaces in CAGD. *IEEE Computer Graphics and Applications*, Vol. 13, No. 1, pp 79-88, 1993.

[14] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, Vol. 11, No. 1, pp 61-91, 1992.

[15] J. L. Blechschmidt and D. Nagasuru. The use of algebraic functions as a solid modeling alternative. *Advances in Design Automation*, B. Ravani Ed., *ASME Design Conference*, Chicago, IL, pp 33-41, Sept. 1990.

[16] F. Dachille IX, 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.

[17] G. Wyvill. C. McPheeters and B. Wyvill. Data structure for soft objects. *The Visual Computer*, Vol. 2, No. 4, pp 227-234, 1988.

[18] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, Vol. 21, No. 4, pp 205-214, July 1987.

[19] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, Vol. 23, No. 3, pp 215-222, 1989.

[20] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constrains. *Computer Graphics*, Vol. 26, No. 2, pp 309-312, July 1992.

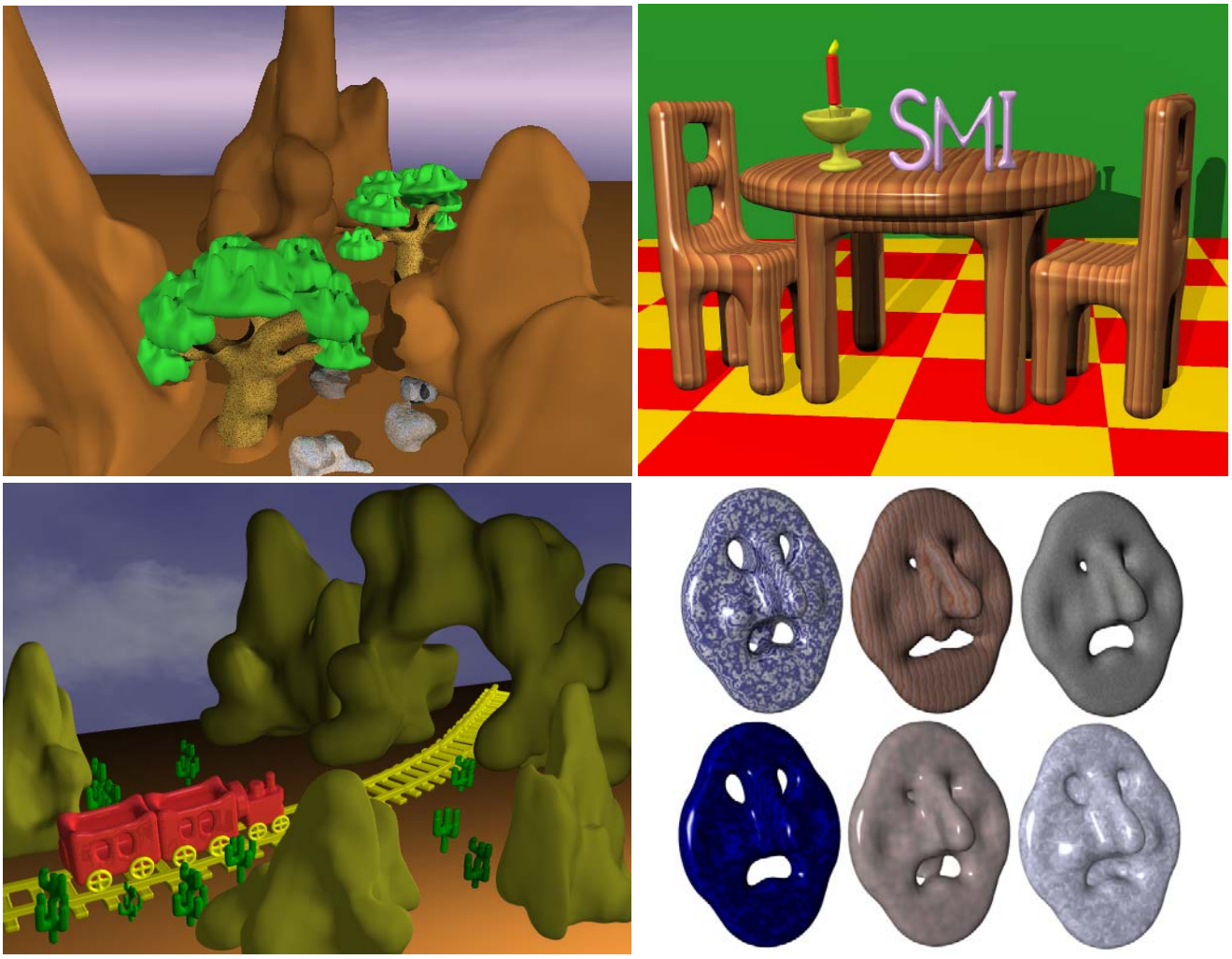


Fig. 10. Several sculptures and scenes created entirely with our system and rendered with POV-Ray.