Dynamic Sculpting and Deformation of Point Set Surfaces

Xiaohu Guo Hong Qin
State University of New York at Stony Brook
Department of Computer Science
Stony Brook, NY 11794-4400, USA
{xguo|qin}@cs.sunysb.edu

Abstract

This paper presents a novel paradigm for point set surface editing, which takes advantages of the potential of implicit surfaces, the strength of physics based modeling techniques, and the simplicity of point sampled surfaces. Our point set surface is evaluated as the zero set of the weighted sum of the collection of the scalar trivariate B-spline functions defined over the local domain of each point sample. The implicit representation of the point set surfaces allows users to easily modify the topology of the sculpted objects. The deformation of the surfaces is conducted by dynamically modifying the local reference domains, as well as their scalar control coefficients. We have developed a variety of sculpting toolkits that can dynamically manipulate the implicit point set surface and easily perform CSG boolean operation on arbitrarily shaped objects. Our research work complements existing point rendering and modeling pipelines for efficient interactive sculpting and deformation.

1. Introductions

Point sampled geometry has attracted more and more attentions in computer graphics. Compared with traditional primitives like triangle mesh, points have shown their advantage such as rendering efficiency and free of connectivity concern, especially when people are dealing with large scanned models coming from laser range and image-based scanning techniques.

Recently many efforts have been focused on both direct rendering techniques [8, 14, 18] and effective modelling mechanisms [1, 12, 17, 11] for point sampled geometry without connectivity. One important challenge to point modeling techniques is to be able to perform fast, interactive modeling tasks which allowing the users to manipulate and sculpt the point clouds intuitively.

In this paper we propose a novel point based geometry representation that is designed for the purpose of dynamic physics-based sculpting and deformation. We use unstructured point samples as the basic modeling and rendering primitive, and add dynamic volumetric implicit functions on top of it, which not only provides the capability to represent surfaces of complex topology, but can also perform fast geometric and physics-based deformations and topology changes. To exploit this unified scheme, we integrate dynamic sculpting and boolean operations into our surface modeling framework.

Instead of using the time consuming moving least square projection operator [1, 11] to determine the shape of the model, our modeling scheme can automatically update its local reference domain and its associated coefficients while doing sculpting, and efficiently reconstruct the new local implicit function undergoing deformation. Our research work is intended for incorporating dynamics and physics into the realm of point based modeling and advance the art of knowledge in this area.

2. Previous Work

Since the pioneering report of Levoy and Whitted [8], the use of point primitive has been investigated by researchers in both shape modeling and rendering. Rusinkiewicz and Levoy [14] introduced a technique named QSplat which uses a hierarchy of spheres of different radii to render a high-resolution model. M. Zwicker et al. [18] introduced the surface splatting technique which directly renders opaque and transparent surfaces from point clouds without connectivity. Later, they present a system called Pointshop 3D [17] for interactive shape and appearance editing of 3D point sampled geometry. The key ingredients of their editing system comprise a point cloud parameterization and a dynamic resampling scheme based on a continuous reconstruction of the surface. But their geometry editing is only limited to normal displacement. Szeliski and Tonnessen [15] introduced a physics-based framework for surface design based on particles. Witking and Heckbert [16] presented a particle-based approach to sampling and editing implicit surfaces. M. Alexa et al. [1] use the framework of moving least square (MLS) projection [6, 7] to approximate a smooth surface defined by a set of points, and introduced some associated techniques to resample the surface to generate an adequate representation of the surface. M. Pauly et al. [12] extend the multiresolution editing concepts, including geometry smoothing, simplification, and offset computation, to point clouds. Most recently, M. Pauly et al. [11] present a free-form shape modeling framework for point sampled geometry using the implicit surface definition of the moving least squares approximation.

Implicit functions are well suited for both scientific visualization and the modeling tasks in computer graphics. To design implicit surfaces interactively, Bloomenthal [2] 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. Our method is very similar to this idea that each point sample has a locally defined distance function, and each individual functions are blended to form an implicit surface. Opalach et al. [9] proposed a simple method for calculating local deformations of implicit surface during collisions. Recently, Raviv and Elber [13] presented a 3D interactive sculpting paradigm that employed a set of scalar uniform trivariate B-spline functions as object representations. Hua and Qin [5] presents a novel haptics-based volumetric modeling framework, which is founded upon volumetric implicit functions and powerful physics-based modeling.

3. Local Surface Distance Field

Our idea of Local Surface Scalar Field is stimulated by the moving least square (MLS) surface model [6, 7, 1]. The input data consists of an unstructured point cloud $P = \{p_i | 1 \leq i \leq N\}$ that describes some underlying manifold surface S. Each point sample stores a geometric position as well as a set of other attributes, such as normal or color. While the continuous MLS surface is defined as a set of projection operator that projects a point near surface onto the MLS surface, our model representation is to define a set of local surface scalar field associated with each point sample. We will briefly review the MLS projection operator and discuss our local scalar field construction.

Given a point set P, the MLS surface $S_{MLS}(P)$ is defined by a projection operator Ψ as the points that project onto themselves, i.e. $S_{MLS}(P) = \{x \in R^3 | \Psi(P, x) = x\}$. The projection $\Psi(P, r)$ of any point r near the surface is defined by a two-step procedure:

First a local reference plane $H = \{x | < n, x > -D = 0, x \in R^3\}$, $n \in R^3$, $\|n\| = 1$ is computed by locally minimizing

$$\sum_{i=1}^{N} (< n, p_i > -D)^2 \theta(\|p_i - q\|), \tag{1}$$

where q is the projection of r onto H. θ is a smooth, monotone-decreasing function, e.g. a Gaussian function $\theta(d) = e^{-\frac{d^2}{h^2}}$, with anticipated spacing h between neighboring points.

Then a bivariate polynomial $g(u, v)$ is fitted to the points projected onto the reference plane H using a similar weighted least squares optimization. This process is carried out by minimizing

$$\sum_{i=1}^{N} (g(u_i, v_i) - h_i)^2 \theta(\|p_i - q\|),$$

where (u_i, v_i, h_i) are the coordinates of p_i in the local coordinate system induced by H. For more details on MLS point set surfaces, please refer to [1]. Note that the normals at the points can be obtained from the reference plane, or by evaluating the gradient of the polynomial g, and then use the minimal spanning tree method similar to [3] to achieve a consistent normal orientation.

Instead of locally fitting a bivariate polynomial $g(u, v)$ to the height function in the reference plane H, we choose to fit a volumetric implicit function to the local distance field in the neighborhood of the point sample p_i. Throughout this paper, we utilize scalar trivariate B-spline functions as the underlying shape primitives [13, 4]. The use of implicit B-spline functions is strongly inspired by their attractive properties including simplicity, generality, etc. These trivariate functions are of the form:

$$s(u, v, w) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} P_{ijk} B_{i,r}(u) C_{j,s}(v) D_{k,l}(w), \tag{2}$$

where $B_{i,r}(u)$, $C_{j,s}(v)$, $D_{k,l}(w)$ are the uniform B-spline basis functions of degrees $r - 1$, $s - 1$, and $l - 1$, respectively. P_{ijk} are the scalar coefficients in a volumetric mesh of size $l \times m \times n$, and $s(u, v, w)$ is a scalar function at position (u, v, w) in parametric domain. In this case, the scalar function defines the distance of position (u, v, w) to the surface. The size of the local reference parameter domain can be chosen to enclose all the k-nearest neighbors of p_i. Here the method using k-nearest neighbors as in [12] is to speed up the weighted least square fitting process and simplify the local reference domain.

For the purpose of fast reconstruction of the local distance field, we can sample the trivariate function (2) at a fixed local grids $G = \{(u_i, v_i, w_i) | i \in [0, g]\}$, while g is the number of the sampling grid positions (u_i, v_i, w_i). Then

(2) can be simplified as the matrix form [4]:

$$\mathbf{s} = (\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D})\mathbf{p}, \tag{3}$$

where $\otimes$ denotes Kronecker Product, **s** is the vector of the distance values at the grid positions, and **p** is the vector of control coefficients. When the distance field is modified, we can easily reconstruct the trivariate function (3) by:

$$\mathbf{p} = [(\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D})^T (\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D})]^{-1} (\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D})^T \mathbf{d}, \tag{4}$$

where **d** is the vector of the new distance values at the grids of G. Note that the grid positions (u_i, v_i, w_i) are fixed in the local domain during our simulation, so the matrix ($\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D}$) can be pre-computed in order to improve real time performance.

In our distance field fitting process, we generate two off-surface points associated with each point sample p_i, one outside and another one inside. Then we employ the weighted least square fitting to obtain the volumetric implicit function, whose zero level set fits the given point samples. We compute the scalar coefficients P_{ijk} so that the weighted least squares error:

$$\sum_{j=1}^{k} (s(u_j, v_j, w_j) - d_j)^2 \theta(\|p_j - p_i\|)$$

is minimized. Here p_i is the current point of our consideration, and p_j is one of its k-nearest neighbors. (u_j, v_j, w_j) is the local parameter coordinate of p_j, and d_j is its distance value to the surface.

The global continuous distance field can be achieved by blending each local implicit primitive associated with each point sample using established implicit blending techniques [2]. Actually we can use a weighting function with finite support that blends individual implicit primitives:

$$s(x, y, z) = \frac{\sum_{i=1}^{N} s_i(x, y, z)\phi_i(x, y, z)}{\sum_{i=1}^{N} \phi_i(x, y, z)}. \tag{5}$$

If (x, y, z) is inside the local region of point sample p_i, $s_i(x, y, z)$ is the distance value evaluated using the trivariate implicit function associated with p_i, otherwise it is simply set to zero. And $\phi_i(x, y, z)$ is a smooth, positive and monotonously decreasing weighting function associate with p_i. Note that in our representation the support of $\phi_i(x, y, z)$ shouldn't exceed the local reference region of p_i. Interpolation of the distance value can be achieved by letting $\phi_i(x, y, z) \to \infty$, when $(x, y, z) \to p_i$.

4. Dynamic Models

In order to introduce physics into our point-based local distance field, we utilize the idea of *Dynamic Volumetric Models* that was first introduced by Hua and Qin in [5]. Our approach is to re-sample the distance field defined by (5) in a region of our interest, which we denote as a global region compared with the point-based local region. Then we discretize the global distance field into a voxel raster. Every voxel contains a scalar value, in our case a distance value, sampled at each grid position using function (5). Note that (5) is defined only around the thin shell of the surface region. To evaluate a location (x, y, z) that is outside the local definition domain of any of the point samples, we can simply approximate the implicit value as its distance to the nearest point in the point set. When we are performing sculpting or deformation, we simulate the dynamics in this global region. In each step of the simulation, if the global distance field is changed on these global grids, we need to reconstruct the local trivariate functions associated with each point by first evaluating the new value of **d**, and then using (4) to update the control coefficients **p**.

Note that after deformation the scalar value associated with each voxel may not be strictly distance value to the iso-surface. So the scalar value should be called density value as in [5]. But in order to maintain consistency with our implicit construction method in Section 3, we still call it distance value below.

4.1. Dynamic Global Volumetric Model

The discretized global distance field is assigned material quantities such as mass, damping, and stiffness distribution. And the discretized distance field is modeled as a collection of mass points connected by a network of springs across nearest neighbor voxels. Actually, the special springs for our implicit function do not change the geometric positions of the voxel mass points, but their distance values. We use the mass spring model to simulate the dynamics of the model as in [5] because of its attractive characteristics of simplicity.

The motion equation of the discretized distance field is formulated as a discrete simulation of Lagrangian dynamics:

$$\mathbf{M_g}\ddot{\mathbf{d}}\mathbf{g} + \mathbf{D_g}\dot{\mathbf{d}}\mathbf{g} + \mathbf{K_g}\mathbf{d_g} = \mathbf{f}_d, \tag{6}$$

where $\mathbf{M_g}$ is the mass matrix, $\mathbf{D_g}$ is the damping matrix, $\mathbf{K_g}$ is the stiffness matrix, and $\mathbf{f_d}$ is the external force vector. We use the notion $\mathbf{d_g}$ to denote the global grids' value compared with the local grid value. The internal forces are generated by the connecting springs, where each spring has force $f = k(I - I_0)$ according to Hook's law. The force mapping mechanism in [5] is also an interesting method to compute the applied sculpting force according to the deformation of the global distance field:

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

where C is any force vector and $s(u,v,w)$ is the distance distribution function in 3D space. For more details on *Dynamic Volumetric Models*, please refer to [5]. Figure 1 shows the mass-spring grids of the global distance field in black color, and the local sampling grid G of one point sample in blue color.

We use a forward Euler method to compute the modified distance values and their velocity:

$$\dot{\mathbf{dg}}_{i+1} = \dot{\mathbf{dg}}_{i} + \ddot{\mathbf{dg}}_{i}\triangle t,$$

$$\mathbf{dg}_{i+1} = \mathbf{dg}_{i} + \dot{\mathbf{dg}}_{i}\triangle t.$$

In our implementation, the users can select any sculpting region of the object, and perform the deformation just inside this specified global region. This region is independent of the surface definition, and it can help to achieve real time performance by limiting the deformation only inside the region of interest.

4.2. Dynamic Updating of Local Domain

Since we are approximating the surface at each local domain of the point samples, in order to allow the dynamic deformation of the model we need to be able to change their local domain dynamically. The local grids scalar value $\mathbf{d}$ and their velocity $\dot{\mathbf{d}}$ can be simply updated by reconstructing from the eight corner values from $\mathbf{dg}$ and $\dot{\mathbf{dg}}$ stored in each global cell using Gaussian blending. After evaluating the new value for d, we can use (4) to update the control coefficients $\mathbf{p}$.

After we alter the global distance field by user interaction, we must change the points' locations since the point samples are assumed to be on the zero level set of the implicit function. When the distance-field space is deformed, the trajectory of the point sample can be represented as $\{\mathbf{x}(t)|s(\mathbf{x}(t),\mathbf{p}(t)) = 0\}$, here $\mathbf{x}(t)$ is the parameter position of the point in its local reference domain, and $\mathbf{p}(t)$ is the vector of control coefficients. The derivative of $s(\mathbf{x}(t),\mathbf{p}(t))$ with respect to time yields:

$$\frac{ds(\mathbf{x}(t),\mathbf{p}(t))}{dt} = \frac{\partial s(\mathbf{x}(t),\mathbf{p}(t))}{\partial \mathbf{x}}\frac{d\mathbf{x}}{dt} + \frac{\partial s(\mathbf{x}(t),\mathbf{p}(t))}{\partial \mathbf{p}}\frac{d\mathbf{p}}{dt} = 0.$$

To simplify the notation, we represent the gradient $\frac{\partial s(\mathbf{x}(t),\mathbf{p}(t))}{\partial \mathbf{x}}$ by $\nabla_{\mathbf{x}} s$, and replace $\frac{\partial s(\mathbf{x}(t),\mathbf{p}(t))}{\partial \mathbf{p}}$ with $\nabla_{\mathbf{p}} s$. Then the above equation can be re-written as follows:

$$\nabla_{\mathbf{x}} s \cdot \dot{\mathbf{x}} + \nabla_{\mathbf{p}} s \cdot \dot{\mathbf{p}} = 0. \qquad (8)$$

Note that $\nabla_{\mathbf{x}} s$ and $\dot{\mathbf{x}}$ are both vectors. Therefore, there is no unique solution for the point velocity. Dividing $\dot{\mathbf{x}}$ into $(\mathbf{v_n}, \mathbf{v_t}, \mathbf{v_w})$, where $\mathbf{n} = -\frac{\nabla_{\mathbf{x}} s}{\|\nabla_{\mathbf{x}} s\|}$ represents the unit principle normal vector of the iso-surface of distance field, $\mathbf{t}$ represents the unit tangent vector, and $\mathbf{w}$ represents unit binormal vector. Then the dot product in (9) only retains the item containing $\mathbf{v_n}$. Therefore, if we assume that the points are only moving in their normal direction, then we can get its normal velocity:

$$\mathbf{v_n} = \frac{\nabla_{\mathbf{p}} s \cdot \dot{\mathbf{p}}}{\|\nabla_{\mathbf{x}} s\|}\mathbf{n} \qquad (9)$$

4.3. Dynamic Sampling

The point sampling density will be changed while users are performing sculpting or deformation on the surface. To maintain a nice surface quality, we need to insert new sample points while the surface density becomes too low, or we can simplify the surface by eliminating points while the surface is squeezed. We use the up-sampling scheme of [1] for the point insertion. In each modeling step, each point should check its neighboring density by projecting its neighbor points onto its tangent plane. Then we compute the Voronoi diagram of these points. We choose the Voronoi vertex that has the largest circle radius on the tangent plane. If the radius is larger than a specified threshold, we can project the vertex onto the iso-surface of the local distance field. Using this approach, we can achieve a surface density of locally near-uniform. We can also reduce the sampling density using the iterative method proposed in [10]. Actually we didn't implement the surface simplification scheme in our real time simulation in order to save computation overhead.

5. Sculpting Toolkits

Our sculpting system has two classes of tools: geometric/topological tools, which employ embossing/engraving and boolean operations to change the shape of the underlying object; and force-based tools, which afford dynamic property and allow users to perform direct sculpting and deformation with ease.

5.1. Geometric and Topological Tools

Embossing and Engraving: Users can also easily perform embossing or engraving operations on an object using our implicit scheme. They can use an existing image to define the distance value at the point samples. Figure 3(a) shows an image of global map. Then we construct the implicit surface by fitting the distance value associated with each point samples. Finally we displace the points onto the iso-surface of its local distance field, and we can get an embossing or engraving effects on the point set surface. As shown in Figure 3(b), the global map is embossed onto a sphere shape to get a ”virtual earth”. We can easily perform the engraving operations on the surfaces by reversing the distance field along the opposite direction.

Boolean Operations: The major advantage of implicit surface modeling system is that we can easily perform CSG (constructive solid geometry) boolean operations such as union, inter-section and difference between half space primitives, since computing surface-surface intersection requires only the evaluation of the implicit function. For point sampled geometry, M. Pauly proposed to use the MLS projection operator to conduct inside/outside classification, but the projection operation can be rather time consuming. In our implicit scheme, the inside/outside classification can be performed by just evaluating the implicit function (5).

One fundamental drawback of our implicit scheme is that we can not represent objects of sharp features yet. So while we are performing boolean operation, we can treat the resulting surface as two different patches of implicit surface if we want to retain the sharp intersection of the original two surfaces. Otherwise we can treat the resulting surface as one single patch to get a relatively smooth intersection.

In our implementation, we use the rendering scheme proposed in [1], i.e. to sample additional points in the neighborhood of an existing point sample at a resolution sufficient to conform to the screen space resolution. Using this rendering method, it is very simple to render the sharp intersection of the two surfaces. Figure 2 illustrates the idea. We need to sample additional points (in black color) near the red point, and we can just perform the evaluation of the implicit function and discard the points that's outside the surface of the intersected patch.

Using CSG boolean operations, users can create objects of complicated geometry and arbitrary topology by using union, intersection, and difference operations. More specifically, considering two closed surfaces S_1 and S_2 that are represented by two point set P_1 and P_2, our goal is to obtain a new point set P that defines the resulting surface S, while P consists of two subsets $Q_1 \subseteq P_1$ and $Q_2 \subseteq P_2$. Then the operations can be performed by computing Q_1 and Q_2 as shown in Table (1). In Table (1), s1 and s2 are the corresponding implicit surface functions of S1 and S2 respectively, and we assume that they have negative values outside the surface and positive inside.

Figure 4 shows the results of the boolean operations between a torus and a sphere. They are rendered using the technique mentioned above to preserve sharp corners.

5.2. Force based tools

Through our force tool the users can select any of the point location inside the sculpting region, and to simulate the dynamics on the mass points inside this region, the force is distributed among nearby mass points using a user-defined function, which can be Gaussian, constant, spherical, or any other distributions. Beside point selection mode, our system affords users the curve-based tool to bind the force along the user defined curve. Figure 5(a) shows the example of using a point tool to deform the surface of a cube. Figure 5(b) shows applying a curve-based force to sculpt the surface of a plate.

6. Experiments

Our system is implemented on a Microsoft Windows NT PC with a Pentium IV 1.4GHz CPU and 512MB RAM. The entire system is written using Microsoft Visual C++ and the graphics rendering component is built upon OpenGL.

We set the number of free control coefficients and local grids to be only $3 \times 3 \times 3$, and it showed that this is already sufficient for each point sample in our experiments. In order to achieve real time simulation, our dynamic sculpting can only handle small data sets since the discretization of the global distance field is space consuming, and the simulation of the dynamics on large data set is time consuming. Currently in our implementation the rendering speed is also a bottleneck limiting the overall performance. The main reason is that we are using the rendering scheme of [1] and sampling the neighboring area around each point at every time step of simulation. If we use the splatting renderer of PointShop3D instead, or even more efficient software or hardware renderers, the simulation performance can be greatly enhanced. In some cases of our experiments, we simply use delayed rendering approach, i.e. simulating the sculpting and deformation process without rendering the whole shape at each simulation step, but just draw the existing point samples. And the user can switch between the two modes depending on their preference.

We have performed many experiments and recorded the running time for both the construction of implicit surface, and the updating time of the sculpting operations. The results are detailed in the Tables (2) and (3). We do not include the rendering time into the simulation time in Table (3) since the rendering time is heavily dependent on the viewing resolution in our implementation. From Table (3) we can see that the number of point samples inside the sculpting region is the fundamental factor influencing the system performance, since the local domain associated with each point sample has to be updated. So it is very important to limit the sculpting region just on the surface of interest.

Using our implicit point set surface modeling framework, we have created several interesting objects. Figure 6 shows the ”PG 2003” logo created by force based curve-tool and boolean operations. We began with six cubic blocks, and then apply the curve-tool by drawing the character as curves on the surface. After the deformation of the blocks, we use small torus shape to link them together using the $\cup$ operations.

7. Conclusions

We have presented a dynamic sculpting and deformation scheme of the point set surface that is based on locally defined trivariate B-spline implicit function. The key feature of our system is the integration of point set surfaces with dynamic sculpting and deformation using dynamic updating of the local reference domain. We have also performed boolean operations on the point surfaces based on this scheme and proposed the corresponding techniques to render sharp corners.

Several further improvements to extend our current work are possible in the near future. We can integrate the sculpting and deformation operation with haptics interface to fully realize the potential of the implicit point set surface, and users can have realistic force feedback when directly manipulating the point set surfaces. And we will further explore the dynamic re-sampling scheme so that we can perform dynamic topology changing of the point set surface such as sketch based editing.

8. Acknowledgements

This research was supported in part by the NSF grants IIS-0082035 and IIS-0097646, and Alfred P. Sloan Fellowship.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, 9(1):3–15, January-March 2003.

[2] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 2(24):109–116, March 1990.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH*, pages 71–78, 1992.

[4] J. Hua and H. Qin. Haptic sculpting of volumetric implicit functions. *Proceedings of 9th Pacific Conference on Computer Graphics and Applications*, pages 254–264, 2001.

[5] J. Hua and H. Qin. Haptics-based volumetric modeling using dynamic spline-based implicit functions. *Proceedings of IEEE Symposium on Volume Visualization and Graphics*, pages 55–64, 2002.

[6] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[7] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization, Springer-Verlag*, 2003, to appear.

[8] M. Levoy and T. Whitted. The use of points as a display primitive. *Technical Report85-022, University of North Carolina at Chapel Hill*, 1985.

[9] A. Opalach and M. Cani-Gascuel. Local deformations for animation of implicit surfaces. In W. Straßer, editor, *13th Spring Conference on Computer Graphics*, pages 85–92, 1997.

[10] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. *IEEE Visualization*, 2002.

[11] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *SIGGRAPH*, 2003.

[12] M. Pauly, L. Kobbelt, and M. Gross. Multiresolution modeling of point-sampled geometry. *TH Zurich Technical Report*, 2002.

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

[14] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. *SIGGRAPH*, pages 343–352, 2000.

[15] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *SIGGRAPH*, 1992.

[16] A. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. *SIGGRAPH*, 1994.

[17] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop3d: An interactive system for point-based surface editing. *SIGGRAPH*, 2002.

[18] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. *SIGGRAPH*, pages 371–378, 2001.

Table 1. The Classification of our implicit boolean operations.

	Q_1	Q_2
$S_1 \cup S_2$	$\{p \in P_1 \| s_2(p) < 0\}$	$\{p \in P_2 \| s_1(p) < 0\}$
$S_1 \cap S_2$	$\{p \in P_1 \| s_2(p) \geq 0\}$	$\{p \in P_2 \| s_1(p) \geq 0\}$
$S_1 - S_2$	$\{p \in P_1 \| s_2(p) < 0\}$	$\{p \in P_2 \| s_1(p) \geq 0\}$

Table 2. The running time for the implicit surface construction.

$\sharp$ Point Samples	Construction Time (s)
674	2.8528
4348	19.277
6542	26.85
8171	37.79
35947	169.803

Table 3. The running time for the dynamic simulation.

$\sharp$ Mass Points	$\sharp$ Points Inside	Updating Time (s)
$40 \times 40 \times 40$	225	0.066838
$40 \times 40 \times 40$	450	0.169078
$40 \times 40 \times 40$	910	0.381281
$50 \times 50 \times 50$	361	0.116774
$60 \times 60 \times 60$	514	0.219434

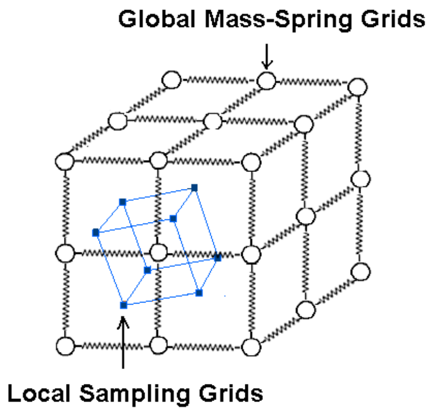

Figure 1. The mass-spring grids (in black color) in the global sculpting domain and the sampling grids (in blue color) in a point's local domain.

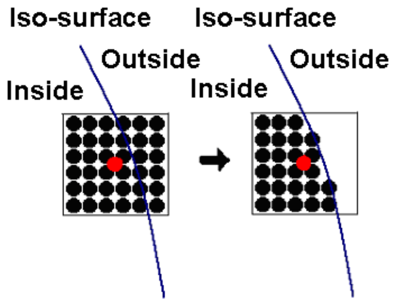

Figure 2. The rendering of the point samples (in red color) near the intersection curve (in blue color).

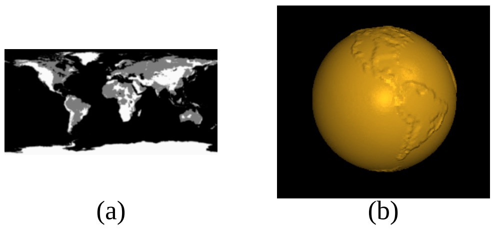

Figure 3. (a) A grey scale image of the global map, which is used to define the local distance field; (b) The embossed sphere.

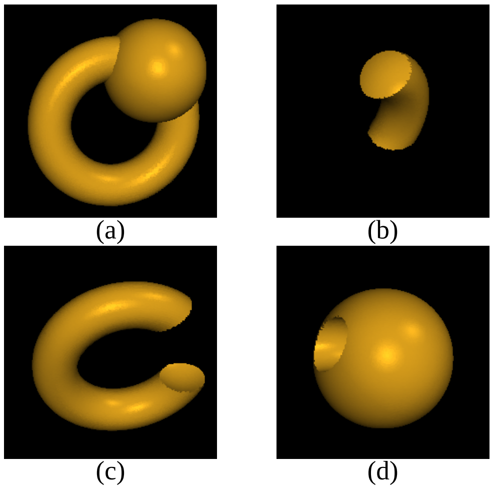

Figure 4. (a) TORUS ∪ SPHERE; (b) TORUS ∩ SPHERE; (c) TORUS − SPHERE; (d) SPHERE − TORUS.

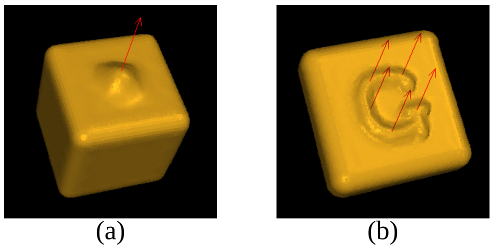

Figure 5. (a) The result of applying a point-force to deform the surface of a cube; (b) Using a curve tool to sculpt a plate.

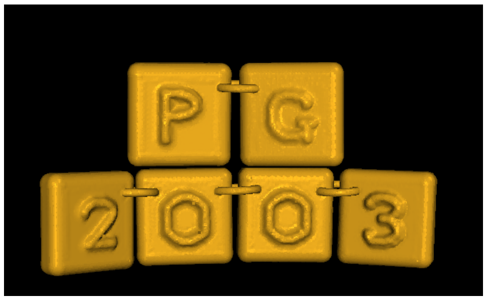

Figure 6. The PG 2003 Logo created using force based curve-tool and boolean operations.