Shape Modeling with Point-Sampled Geometry

Mark Pauly ETH Zürich
Richard Keiser ETH Zürich
Leif P. Kobbelt RWTH Aachen
Markus Gross ETH Zürich

(a) (b) (c)

Figure 1: Objects created with our system. (a) boolean operations with scanned geometry, (b) an Octopus modeled by deforming and extruding a sphere, (c) a design study for a Siggraph coffee mug created by boolean operations, free-form deformation and displacement mapping.

Abstract

We present a versatile and complete free-form shape modeling framework for point-sampled geometry. By combining unstructured point clouds with the implicit surface definition of the moving least squares approximation, we obtain a hybrid geometry representation that allows us to exploit the advantages of implicit and parametric surface models. Based on this representation we introduce a shape modeling system that enables the designer to perform large constrained deformations as well as boolean operations on arbitrarily shaped objects. Due to minimum consistency requirements, point-sampled surfaces can easily be re-structured on the fly to support extreme geometric deformations during interactive editing. In addition, we show that strict topology control is possible and sharp features can be generated and preserved on point-sampled objects. We demonstrate the effectiveness of our system on a large set of input models, including noisy range scans, irregular point clouds, and sparsely as well as densely sampled models.

Keywords: shape modeling, point-sampled geometry, free-form deformation, boolean operations, dynamic sampling.

1 INTRODUCTION

In computer graphics a large variety of geometry representations has been used for reconstruction, modeling, editing, and rendering of 3D objects. On the most abstract level, these representations fall into two major categories: *Implicit* and *parametric* representations. Both categories have their specific advantages and drawbacks. Implicit representations [Velho et al. 2002] such as level sets [Osher and Fedkiw 2002] and radial basis functions [Carr et al. 2001] have a particularly simple algebraic structure. Surfaces are defined as zero-sets of 3D scalar fields, which are generated by a weighted superposition of basis functions $\Phi_i: \mathbb{R}^3 \to \mathbb{R}$. In this setting, surfaces with highly complex topology can be represented easily and the global consistency of the surface is guaranteed by construction. Extreme geometric deformations and even topology changes can be achieved by simply modifying the weight coefficients of the respective basis functions. However, efficient rendering and the precise modeling of sharp features is usually a difficult task, since individual points on implicitly represented surfaces can only be accessed by some ray-casting technique.

Parametric representations like splines [Farin 2002], subdivision surfaces [Zorin and Schröder 2000], or triangle meshes are based on a mapping from a (piecewise) planar domain Ω into $\mathbb{R}^3$. Here the algebraic structure is usually more complicated and tightly correlated with the complexity of the surface. Since point samples on the surface can be obtained by evaluating a function $f: \Omega \to \mathbb{R}^3$, parametric surfaces can be rendered quite efficiently. Moreover, sharp creases along a curve Γ on the surface can be modeled by adjusting the function f along the pre-image $f^{-1}(\Gamma)$ of Γ. On the other hand, extreme deformations and topology changes usually require changes to the domain Ω in order to avoid strong distortions and inconsistencies. The shape of a triangle mesh, for example, can be modified to a certain extend by only changing the vertex positions, but keeping the connectivity fixed. However, if the local distortion becomes too strong or if the surface topology is to be changed, we have to update the mesh connectivity while maintaining strict consistency conditions to preserve a manifold surface structure. In practice, the necessity to re-establish the global consistency of the mapping f makes this kind of operations on parametric representations rather inefficient.

In this paper we propose a hybrid geometry representation that is designed to combine the advantages of implicit and parametric surface models. We use an unstructured cloud of point samples to represent the mere geometric shape. As a second component, we add an implicit surface definition based on the moving least squares projection [Levin , to appear], which provides access to important surface attributes such as the signed distance function or normal information.

We exploit this hybrid structure to integrate robust boolean operations and constrained free-form deformations into a unified shape modeling framework. The surface geometry can be explicitly accessed through the sample points, while the implicit surface model is used for distance and normal queries.

For boolean operations this combination allows us to reliably recover the intersection curves of two objects and place additional samples exactly on the resulting sharp feature line.

For free-form modeling, we are able to perform very large surface deformations using a sampling method that dynamically inserts or deletes sample points when the local sample density becomes too low or too high. During deformations, the point cloud is not treated as a fixed, static representation of the underlying manifold surface, but rather as a set of temporary samples that evolves over time as the surface deforms. This gives the designer great flexibility when modeling a well-defined region of the surface by pulling, pushing or twisting a control handle. In addition, the modeling tool utilizes the signed distance function to provide strict topology control: Intersections of the deformed surface with itself can be detected efficiently and, depending on the design intent, surfaces can be repulsed or merged. For completeness, we have also implemented a blending operator to create smooth transitions between two merged surface parts.

We consider our work as an integrated component of an intuitive modeling system for the creation of 3D content for computer graphics applications. As opposed to industrial design applications, where geometric models are usually constructed from scratch, the creation of 3D models for computer graphics is often based on editing detailed models that have been acquired by some 3D scanning device. Such scanners typically provide dense samplings of some physical prototype. Point-based modeling techniques can directly handle this kind of input data without requiring extensive pre-processing. Additionally, recent progress in the efficient rendering of point-sampled surfaces [Zwicker et al. 2001], [Rusinkiewicz and Levoy 2000], [Botsch et al. 2002] demonstrates that point-based representations are well suited for interactive applications.

1.1 Related work

Since the pioneering report by Levoy and Whitted [Levoy and Whitted 1985], the use of point primitives for shape modeling has been investigated by various researchers. Szeliski and Tonnessen introduced oriented particles [Szeliski and Tonnesen 1992] as a physics-based framework for surface design. A similar idea has been proposed by Witkin and Heckbert for sampling and editing implicit surfaces [Witkin and Heckbert 1994]. We use an adaptation of oriented particles to compute the blend region of two surfaces that have been combined in a boolean union operation. However, we found that full-scale physics-based particle simulations are computationally too involved for interactive modeling of large point-sampled models. Nevertheless, we believe that physics-based simulation provides an expressive and intuitive approach to surface design.

Free-form shape deformations have been studied extensively in the past [Barr 1984], [Sederberg and Parry 1986], [Chang and Rockwood 1994]. Our free-form modeling tool bears some similarity to the *wires* system of [Singh and Fiume 1998], which uses a set of one-dimensional curves to define a smooth deformation field. We extend this scheme by allowing arbitrary subsets of the surface to define the underlying distance functions (see Section 4).

Similarly, surface modeling using boolean operations has been the focus of significant research efforts. For an overview, we refer the reader to [Hoffmann 1989]. A modeling framework based on level sets has been presented in [Museth et al. 2002]. This system clearly demonstrates the strength of implicit representations for performing boolean operations and handling complex surface topology. Interactivity is very limited, however, partly because the model has to be converted to a triangle mesh for display.

Various researchers have addressed the problem of dynamically adapting the sampling density of discrete surfaces in the presence of large geometric deformations. Welch and Witkin [Welch and Witkin 1994] have presented a shape modeling system for triangle meshes that uses vertex split and edge collapse operators to keep the sampling density uniform. A similar approach was taken in [Kobbelt et al. 2000] for multiresolution meshes with dynamic vertex connectivity. Our dynamic sampling method is also based on point splitting operations. The main difference is that we do not have to maintain the consistency of a mesh connectivity graph, which leads to simpler code and increased performance.

2 HYBRID SURFACE MODEL

Our input data consists of an unstructured point cloud $P = \{\mathbf{p}_i = (x_i, y_i, z_i) \,|\, 1 \le i \le n\}$ that approximates some underlying manifold surface. Additional to the geometric position, each sample point can also store a set of scalar attributes, such as color, material properties or texture coordinates.

We extend this explicit point cloud representation by the moving least squares (MLS) surface model introduced in [Levin , to appear]. The continuous MLS surface S is defined implicitly as the stationary set of a projection operator $\Psi(P, \mathbf{r})$ that projects a point $\mathbf{r} \in \mathbb{R}^3$ onto the MLS surface $S = \{\mathbf{x} \in \mathbb{R}^3 \,|\, \Psi(P, \mathbf{x}) = \mathbf{x}\}$. To evaluate $\Psi(P, \mathbf{r})$ we first compute a local reference plane

$$H = \{\mathbf{x} \in \mathbb{R}^3 \,|\, \mathbf{x} \cdot \mathbf{n} - D = 0\} \tag{1}$$

by minimizing the weighted sum of squared distances

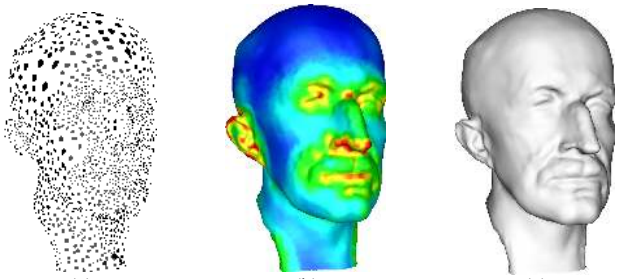
$$\sum_{\mathbf{p} \in P} (\mathbf{p} \cdot \mathbf{n} - D)^2 \phi(\|\mathbf{p} - \mathbf{q}\|), \tag{2}$$

where $\mathbf{q}$ is the projection of $\mathbf{r}$ onto H and ϕ is the MLS kernel function. After transforming all points into the local frame defined by H, a second least squares optimization yields a bivariate polynomial $g(u, v)$ that locally approximates the surface. The projection of $\mathbf{r}$ onto S is then given as $\Psi(P, \mathbf{r}) = \mathbf{q} + g(0, 0) \cdot \mathbf{n}$ (for more details see [Alexa et al. 2001]).

Per point normals can be obtained directly from the reference plane or by evaluating the gradient of the polynomial g. To achieve a consistent orientation, we use a method based on the minimum spanning tree, similar to [Hoppe et al. 1992].

The characteristics of the surface can be controlled by the kernel function ϕ. Typically, a Gaussian $\phi(t) = \exp(-t^2/h^2)$ is chosen, where h is a global scale parameter that determines the feature size of the resulting surface. As has been observed in [Alexa et al. 2001], the MLS projection operator essentially implements a low-pass filter, where the degree of smoothness depends on the scale parameter. This smoothing effect can be exploited for pre-processing noisy scanner data (see also Figure 19).

Adaptive MLS projection. Using a fixed global scale factor can be problematic for non-uniformly sampled surfaces. If h is too large, areas of high sampling density will be smoothed excessively, while numerical instabilities occur in sparsely sampled regions, if h is too small. To deal with this problem we use an extension of



(a) (b) (c)

Figure 2: Adaptive MLS reconstruction of the Max Planck bust. (a) input point cloud with 5,413 points, (b) continuous sampling density map, (c) reconstructed MLS surface.

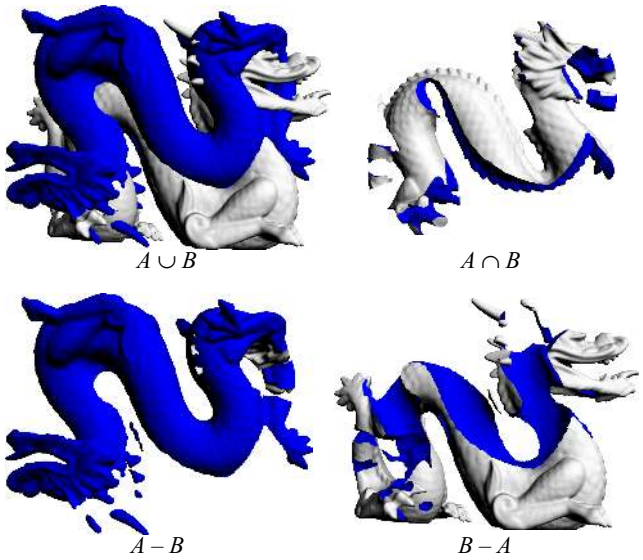
the standard MLS method as proposed in [Pauly et al. 2002], which is based on sampling density estimation. We adapt the MLS kernel to $\phi_{\mathbf{r}}(t) = \exp(-t^2/h_{\mathbf{r}}^2)$, where $h_{\mathbf{r}} = h/\rho(\mathbf{r})$ and $\rho(\mathbf{r}):\mathbb{R}^3 \to \mathbb{R}_+$ is a continuous, smooth function approximating the local sampling density. To compute ρ, we first estimate the local sampling density ρ_i for each $\mathbf{p}_i \in P$ by finding the sphere with minimum radius r_i centered at $\mathbf{p}_i$ that contains the k-nearest neighbors to $\mathbf{p}_i$. Then ρ_i is defined as $\rho_i = k/r_i^2$, where our experiments showed that k should be greater than 6 to ensure stable computations, but less than 20 to avoid excessive smoothing of the density estimation. In a second step, ρ can be interpolated using standard scattered data approximation techniques, e.g. radial basis functions. For more details on the approximation power, stability, and implementational issues related to MLS surfaces, we refer the reader to [Levin 1998] and [Alexa et al. , to appear].

3 BOOLEAN OPERATIONS

A common approach in geometric modeling is to build complex objects by combining simple shapes using boolean operations [Hoffmann 1989] (see Figure 3). In constructive solid geometry (CSG) objects are defined using a binary tree, where each node corresponds to a union, intersection, or difference operation and each leaf stores a base shape. Operations such as ray-tracing, for example, are then implemented by traversing this tree structure. More commonly, surfaces are defined as boundary representations (B-Rep) of solids. Here boolean operations have to be evaluated explicitly, which requires an algorithm for intersecting two surfaces. Computing such a surface-surface intersection can be quite involved, however, in particular for higher order surfaces [Krishnan and Manocha 1997].

For point-sampled geometry we can use the MLS projection operator both for inside/outside classification as well as for explicitly sampling the intersection curve. The goal is to perform a boolean operation on two orientable, closed surfaces S_1 and S_2 that are represented by two point clouds P_1 and P_2, to obtain a new point cloud P° that defines the resulting surface S°. P° consists of two subsets $Q_1 \subseteq P_1$ and $Q_2 \subseteq P_2$ plus a set of newly generated sample points that explicitly represent the intersection curves. Thus in order to perform a boolean operation for point-sampled geometry, we need the following techniques:

- a classification predicate to determine the two sets Q_1 and Q_2,
- an algorithm to find samples on the intersection curve, and
- a rendering method that allows to display crisp features curves using point primitives.



$A \cup B$ $A \cap B$

$A - B$ $B - A$

Figure 3: Union, intersection and difference operations of two complex, non-convex surfaces.

3.1 Classification

For inside/outside classification we define a predicate Ω_P such that for $\mathbf{x} \in \mathbb{R}^3$

$$\Omega_P(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in V \\ 0 & \mathbf{x} \notin V \end{cases} \quad (3)$$

where V is the volume bounded by the MLS surface S represented by the point cloud P. Let $\mathbf{y} \in S$ be the closest point on S from $\mathbf{x}$. It is well-known from differential geometry that for S continuous and twice differentiable, the vector $\mathbf{x} - \mathbf{y}$ is aligned with the surface normal $\mathbf{n}_y$ at $\mathbf{y}$ [DoCarmo 1976]. If surface normals are consistently oriented to point outwards of the surface, then $(\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}_y > 0$ if and only if $\mathbf{x} \notin V$. Since we are given a discrete sample P of the surface S, we replace the closest point $\mathbf{y}$ on S by the closest point $\mathbf{p} \in P$ and classify $\mathbf{x}$ as outside if $(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n}_p > 0$, i.e. if the angle between $\mathbf{x} - \mathbf{p}$ and the normal $\mathbf{n}_p$ at $\mathbf{p}$ is less than $\pi/2$ (see Figure 4 (a)). This discrete test yields the correct inside/outside classification of the point $\mathbf{x}$ if the distance $\|\mathbf{x} - \mathbf{p}\|$ is bigger than the local sample spacing η_p at $\mathbf{p}$. If we are extremely close to the surface, the classification could fail, as illustrated in Figure 4 (b). In this case we compute the exact closest point $\mathbf{y} \in S$ using the MLS projection. Thus we obtain a very robust inside/outside classification that easily handles com-

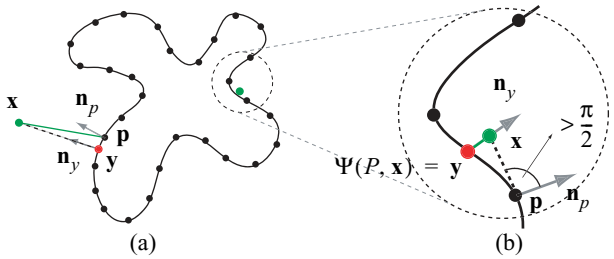


Figure 4: Inside/outside test. For $\mathbf{x}$ very close to the surface, the closest point $\mathbf{p} \in P$ can yield a false classification (right image). In this case, $\mathbf{x}$ is classified according to its MLS projection $\mathbf{y}$.

plex, non-convex surfaces, as illustrated in Figure 3. Since we are only interested in an inside/outside classification, we can significantly improve the performance by exploiting local coherence: $\Omega_P(\mathbf{x}) = \Omega_P(\mathbf{x}')$ for all points $\mathbf{x}'$ that lie in the sphere around $\mathbf{x}$ with radius $\|\mathbf{x} - \mathbf{p}\| - \eta_p$. Thus the number of closest point queries and MLS projections can be reduced drastically, in practice up to 90 percent. Given the classification predicate Ω, the subsets Q_1 and Q_2 can be computed as shown in Table 1.

	Q_1	Q_2
$S_1 \cup S_2$	$\{\mathbf{p} \in P_1 \mid \Omega_{P_2}(\mathbf{p}) = 0\}$	$\{\mathbf{p} \in P_2 \mid \Omega_{P_1}(\mathbf{p}) = 0\}$
$S_1 \cap S_2$	$\{\mathbf{p} \in P_1 \mid \Omega_{P_2}(\mathbf{p}) = 1\}$	$\{\mathbf{p} \in P_2 \mid \Omega_{P_1}(\mathbf{p}) = 1\}$
$S_1 - S_2$	$\{\mathbf{p} \in P_1 \mid \Omega_{P_2}(\mathbf{p}) = 0\}$	$\{\mathbf{p} \in P_2 \mid \Omega_{P_1}(\mathbf{p}) = 1\}$
$S_2 - S_1$	$\{\mathbf{p} \in P_1 \mid \Omega_{P_2}(\mathbf{p}) = 1\}$	$\{\mathbf{p} \in P_2 \mid \Omega_{P_1}(\mathbf{p}) = 0\}$

Table 1: Classification for boolean operations.

3.2 Intersection curves

Taking the union of Q_1 and Q_2 will typically not produce a point cloud that accurately describes the surface S°, since the intersection curve of the two MLS surfaces S_1 and S_2 is not represented adequately. Therefore we explicitly compute a set of sample points that lie on the intersection curve and add them to $Q_1 \cup Q_2$ to obtain the point cloud P°. First we find all points in Q_1 and Q_2 that are close to the intersection curve by evaluating the distance function induced by the MLS projection operator. Then we look at all closest pairs $(\mathbf{q}_1 \in Q_1, \mathbf{q}_2 \in Q_2)$ of these points and compute a point $\mathbf{q}$ on the intersection curve using a Newton-type iteration. This is done as follows (see Figure 5 (a-d)): Let $\mathbf{r}$ be the point on the intersection line of the two tangent planes of $\mathbf{q}_1$ and $\mathbf{q}_2$ that is closest to both points, i.e. that minimizes the distance $\|\mathbf{r} - \mathbf{q}_1\| + \|\mathbf{r} - \mathbf{q}_2\|$. $\mathbf{r}$ is the first approximation of $\mathbf{q}$ and can now be projected onto S_1 and S_2 to obtain two new starting points $\mathbf{q}_1'$ and $\mathbf{q}_2'$ for the iteration. This procedure can be repeated iteratively until the points $\mathbf{q}_1$ and $\mathbf{q}_2$ converge to a point $\mathbf{q}$ on the intersection curve. Due to the quadratic convergence of the Newton iteration, this typically requires less than three iterations.

We use the sampling density estimation of Section 2 to detect whether the sampling resolution of the two input surfaces differs significantly in the vicinity of the intersection curve. To avoid a sharp discontinuity in sampling density, we up-sample the coarser model in this area to match the sampling density of the finer model, using the dynamic sampling method of Section 5.

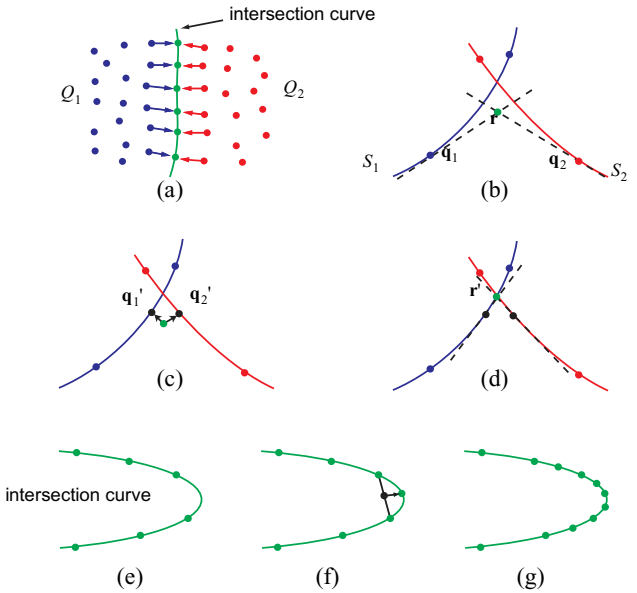


Figure 5: Sampling the intersection curve. (a) closest pairs of points in Q_1 and Q_2, (b) first estimate $\mathbf{r}$, (c) re-projection, (d) second estimate $\mathbf{r}'$, (e-g) adaptive refinement.

Note that the above Newton scheme also provides an easy mechanism for adaptively refining the intersection curve. We can use a simple subdivision rule to create a new starting point for the Newton iteration, e.g. the average of two adjacent points on the curve, and then apply the iteration to create a new sample on the intersection curve (see Figure 5 (e-g)).

3.3 Rendering sharp creases

For an accurate display of the intersection curves we need to be able to render sharp creases and corners. We use an extension of the *surface splatting* technique presented in [Zwicker et al. 2001]. In this method, each sample point is represented by a *surfel*, an oriented elliptical splat that is projected onto the screen to reconstruct the surface in image space. A point on the intersection curve can now be represented by two surfels that share the same center, but whose normals stem from either one of the two input surfaces. During scan-conversion, each of these surfels is then clipped against the plane defined by the other to obtain a piecewise linear approximation of the intersection curve in screen space (see Figure 6). This concept can easily be generalized to handle corners as shown in Figure 6 (e). Figure 7 shows an example of a difficult boolean operation of two identical cylinders that creates two singularities. While the classification and intersection curve sampling work fine, the rendering method produces artefacts. This is due to numerical instabilities, since the clipping planes of two corresponding surfels are almost parallel. However, such cases are rare in typical computer graphics applications, e.g. digital character design, which are the focus of our work. As such, our algorithms are less suited for industrial manufacturing applications, where handling of degenerated cases is of primary concern.

4 FREE-FORM DEFORMATION

Apart from composition of surfaces using boolean operations, many shape design applications require the capability to modify objects using smooth deformations. These include bending, twist-

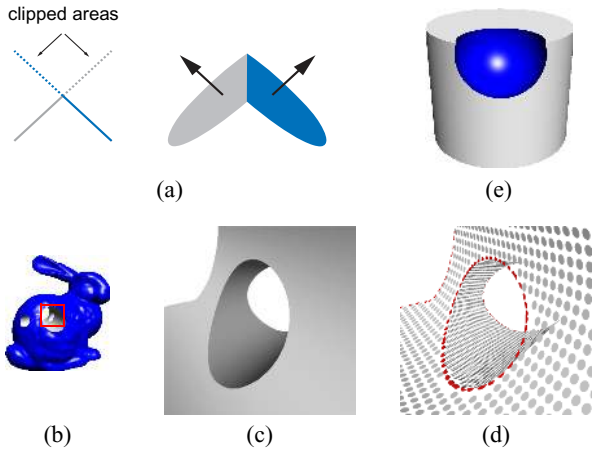


Figure 6: Rendering the intersection curve. (a) mutual clipping of two surfels on the intersection curve, (b) boolean differences on the bunny model, (c) zoom of the intersection curves, (d) sampling distribution, where samples on the intersection curve are rendered using two red half ellipses, (e) an example of a corner.

Figure 7: A difficult boolean difference operation.

ing, stretching and compressing of the model surface. For this purpose we introduce a point-based free-form deformation tool that allows the user to interactively deform a surface by specifying a smooth deformation field.

The user first defines a deformable region $\chi_d \subset S$ on the model surface and marks parts of this region as a control handle. The surface can then be modified by pushing, pulling or twisting this handle. These user interactions are translated into a continuous tensor-field, which for each point in the deformable region defines a translatory and rotational motion under which the surface deforms. The tensor-field is based on a continuously varying scale parameter $t \in [0, 1]$ that measures the relative distance of a point from the handle. The closer a point is to the handle, the stronger will the deformation be for that point. More precisely, let $\chi_1 \subset \chi_d$ be the handle, also called *one-region*, and $\chi_0 = S - \chi_d$ the *zero-region*, i.e. all points that are not part of the deformable region. For both zero- and one-region we define distance measures d_0 and d_1 respectively, as

$$d_j(\mathbf{p}) = \begin{cases} 0 & \mathbf{p} \in \chi_j \\ \min_{\mathbf{q} \in \chi_j}(\|\mathbf{p} - \mathbf{q}\|) & \mathbf{p} \notin \chi_j \end{cases}, \qquad (4)$$

for $j = 0, 1$. From these distance measures we compute the scale parameter t as $t = \beta(d_0(\mathbf{p})/(d_0(\mathbf{p}) + d_1(\mathbf{p})))$, where $\beta{:}[0,1] \to [0, 1]$ is a continuous blending function with $\beta(0) = 0$ and $\beta(1) = 1$. Thus $t = 0$ for $\mathbf{p} \in \chi_0$ and $t = 1$ for $\mathbf{p} \in \chi_1$. Using this scale parameter, we can determine the position of a point $\mathbf{p} \in \chi_d$ after the deformation as $\mathbf{p}' = F(\mathbf{p}, t)$, where F is a deformation function composed of a translatory and a rotational part. We can write $F(\mathbf{p}, t) = F_T(\mathbf{p}, t) + F_R(\mathbf{p}, t)$, where

- $F_T(\mathbf{p}, t) = \mathbf{p} + t \cdot \mathbf{v}$ with $\mathbf{v}$ a translation vector and
- $F_R(\mathbf{p}, t) = R(\mathbf{a}, t \cdot \alpha) \cdot \mathbf{p}$, where $R(\mathbf{a}, \alpha)$ is the matrix that specifies a rotation around axis $\mathbf{a}$ with angle α.

Figure 8 shows a translatory deformation of a plane where the translation vector $\mathbf{v}$ is equal to the plane normal. This figure also illustrates the effect of different choices of the blending function β. In Figure 9, two rotational deformations of a cylinder are shown, while a combination of both translatory and rotational deformations is illustrated in Figure 13.

To perform a free-form deformation the user only needs to select the zero- and one-regions and choose an appropriate blending function. She can then interactively deform the surface by displacing the handle with a mouse or trackball device, similar to [Kobbelt et al. 1998]. This gives our method great flexibility for handling a wide class of free-form deformations, while still providing a simple and intuitive user interface.

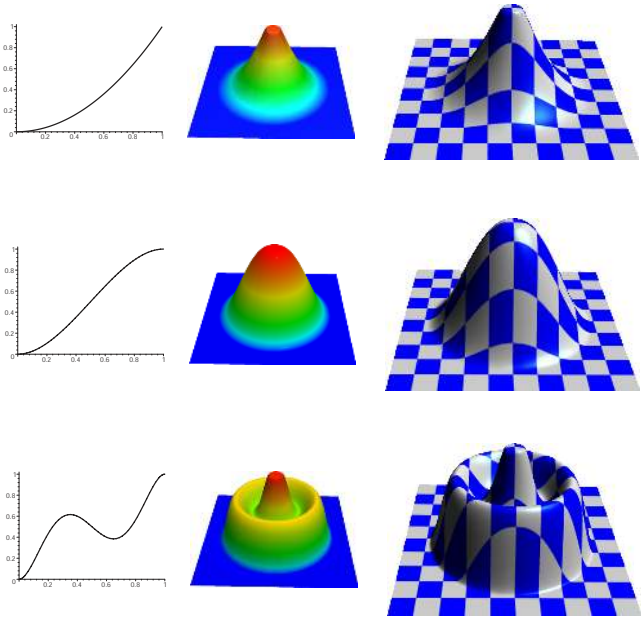


Figure 8: Deformations of a plane for three different blending functions. Left: Blending function, middle: Color-coded scale parameter, where blue indicates the zero region ($t = 0$) and red the one-region ($t = 1$), right: Final textured surface.

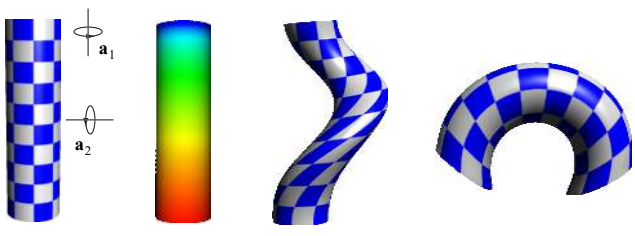


Figure 9: Rotational deformations of a cylinder. (a) original, (b) color-coded scale parameter, (c) rotation around axis $\mathbf{a}_1$, (d) rotation around axis $\mathbf{a}_2$.

4.1 Topology Control

An important issue in shape design using free-form deformation is the handling of self-intersections. During deformation, parts of the deformable region can intersect other parts of the surface, which leads to an inconsistent surface representation. To cope with this problem we need a method for detecting and resolving such collisions.

Collision Detection. Similar to boolean operations, this requires an inside/outside classification to determine which parts of the surface have penetrated others. Thus we can again apply the classification predicate Ω defined in Section 3.1. We start by computing for each sample point $\mathbf{p} \in \chi_d$ the closest point $\mathbf{q} \in \chi_0$. This defines a empty sphere s_p around $\mathbf{p}$ with radius $\|\mathbf{p} - \mathbf{q}\|$. If the point $\mathbf{p}$ only moves within this sphere during deformation, it is guaranteed not to intersect with the zero-region (see Figure 10). So additionally to exploiting spatial coherence as for boolean classification, we can now also exploit the temporal coherence induced by the smooth deformation field. Only when $\mathbf{p}$ leaves s_p do we have to re-evaluate the classification predicate Ω, which at the same time provides a new estimate for the updated sphere s_p.

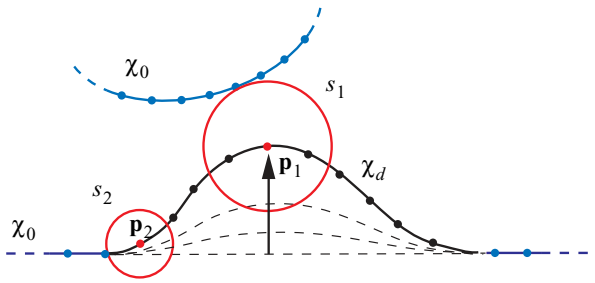


Figure 10: Temporal coherence for collision detection during deformation. The points $\mathbf{p}_1$ and $\mathbf{p}_2$ can move with the spheres s_1 and s_2, resp., without intersecting the zero-region.

Collision Handling. There are different ways to respond to a detected collision. The simplest solution is to undo the last deformation step and recover the surface geometry prior to the collision. Alternatively, we can join the penetrating parts of the surface using a boolean union operation to maintain the validity of the surface. As illustrated in Figure 3, boolean operations typically produce sharp intersection curves. In the context of free-form deformation it is often more desirable to create a smooth blend between the two combined surface parts. Therefore, we have implemented an adaptation of oriented particles [Szeliski and Tonnesen 1992] to smooth out the sharp creases created by boolean operations. The idea is to define inter-particle potentials $\Phi(\mathbf{p}_i, \mathbf{p}_j)$ in such a way that the minimum of the global potential function yields a smooth surface that minimizes curvature. By summing up these potentials we obtain a particle's total potential energy E_i (see [Szeliski and Tonnesen 1992] for details). From this potential energy we can derive positional and rotational forces that are exerted on each particle and compute its path of motion under these forces. Additionally, we apply an inter-particle repulsion force to equalize the particle distribution, and scale all forces with a smooth fall-off function that measures the distance to the intersection curve. Thus we can confine the particle simulation to a small area around the intersection curve without affecting other parts of the surface.

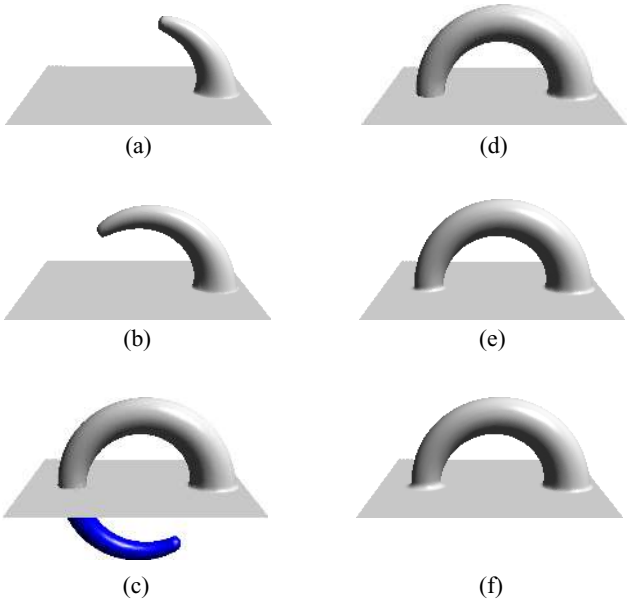


Figure 11: Blending (a - b) intermediate steps of the deformation, (c) collision detection, where the blue part has been detected as self-intersecting, (d), boolean union with sharp intersection curve, (e - f), particle-based blending with different fall-off functions.

Figure 11 shows an editing session, where a deformation creates a self-intersection. After performing a boolean union, the particle simulation creates a smooth transition in the intersection region. The same blending technique can of course also be applied to the intersection and difference operations described in Section 3.

5 DYNAMIC SAMPLING

Large deformations may cause strong distortions in the distribution of sample points on the surface that can lead to an insufficient local sampling density. To prevent the point cloud from ripping apart and maintain a high surface quality, we have to include new samples where the sampling density becomes too low. To achieve this we first need to measure the surface stretch to detect regions of insufficient sampling density (Section 5.1). Then we have to insert new sample points and determine their position on the surface. Additionally, we have to preserve and interpolate the scalar attributes, e.g. color values or texture coordinates (Section 5.2).

5.1 Measuring surface stretch

We use the first fundamental form known from differential geometry to measure the local distortion of a surface under deformation. Let $\mathbf{u}$ and $\mathbf{v}$ be two orthogonal tangent vectors of unit length at a sample point $\mathbf{p}$. The first fundamental form at $\mathbf{p}$ is now defined by the 2×2 matrix

$$\begin{bmatrix} \mathbf{u}^2 & \mathbf{u} \cdot \mathbf{v} \\ \mathbf{u} \cdot \mathbf{v} & \mathbf{v}^2 \end{bmatrix}. \qquad (5)$$

The eigenvalues of this matrix yield the minimum and maximum stretch factors and the corresponding eigenvectors define the principal directions into which this stretching occurs. When we apply a deformation, the point $\mathbf{p}$ is shifted to a new position $\mathbf{p}'$ and the

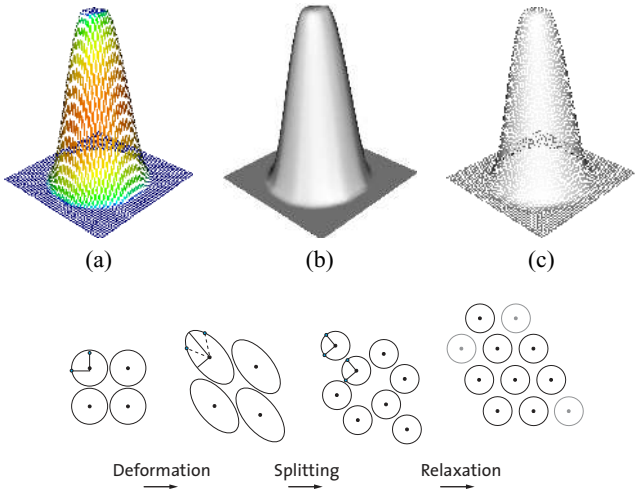


Figure 12: Dynamic sampling. Top row: Deformation of a plane. (a) local stretching: blue corresponds to zero stretch, while red indicates maximum stretch, (b) surface after re-sampling, (c) sampling distribution. Bottom row: illustration of point insertion.

two tangent vectors are mapped to new vectors $\mathbf{u}'$ and $\mathbf{v}'$. Local stretching implies that $\mathbf{u}'$ and $\mathbf{v}'$ might no longer be orthogonal to each other nor do they preserve their unit length. The amount of this distortion can be measured by taking the ratio of the two eigenvalues of (5) (local anisotropy) or by taking their product (local change of surface area). When the local distortion becomes too strong, we have to insert new samples to re-establish the prescribed sampling density. Since Equation (5) defines an ellipse in the tangent plane centered at $\mathbf{p}$ with the principal axes defined by the eigenvectors and eigenvalues, we can easily replace $\mathbf{p}$ by two new samples $\mathbf{p}_1$ and $\mathbf{p}_2$ that we position on the main axis of the ellipse (cf. Figure 12).

5.2 Filter operations

Whenever a splitting operation is applied, we need to determine both the geometric position and the scalar function values for the newly generated sample points. Both these operations can be described as the application of a filtering operator: If we apply the operator to the sample positions, we call it a *relaxation filter*, while we call it an *interpolation filter*, if we apply it to function values.

Relaxation. Introducing new sample points through a splitting operation creates local imbalances in the sampling distribution. To obtain a more uniform sampling pattern, we apply a relaxation operator that moves the sample points within the surface (see Figure 12). Similar to [Turk 1992] we use a simple point repulsion scheme with a repulsion force that drops linearly with distance. We can thus confine the radius of influence of each sample point to its local neighborhood, which allows very efficient computation of the relaxation forces. The resulting displacement vector is then projected into the point's tangent plane to keep the samples on the surface.

Interpolation. Once we have fixed the position of a new sample point $\mathbf{p}$ using the relaxation filter, we need to determine its associated function values. This can be achieved using an interpolation filter by computing a local average of the function values of neighboring sample points. We need to be careful, however. The relaxation filter potentially moves all points of the neighborhood of $\mathbf{p}$.

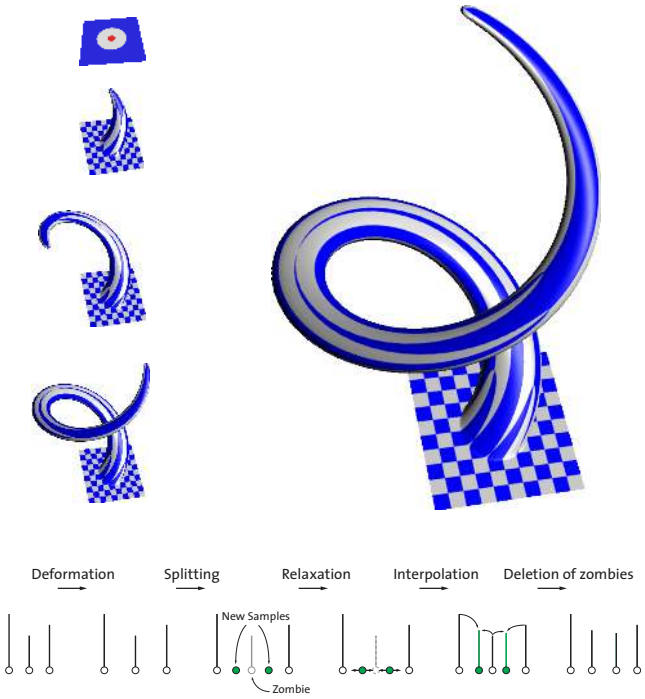


Figure 13: A very large deformation using a combination of translatory and rotational motion. The left column shows intermediate steps with the top image indicating zero- and one-regions. Each point of the surface carries texture coordinates, which are interpolated during re-sampling and used for texturing the surface with a checkerboard pattern. The bottom row illustrates this interpolation process, where the function values are indicated by vertical lines.

This tangential drift leads to distortions in the associated scalar functions. To deal with this problem we create a copy of each point that carries scalar attributes and keep its position fixed during relaxation. In particular, we maintain for each sample that is split a copy with its original data. These points will only be used for interpolating scalar values, they are not part of the current geometry description. Since these samples are *dead* but their function values still *live*, we call them *zombies*. Zombies will undergo the same transformation during a deformation operation as living points, but their positions will not be altered during relaxation. Thus zombies accurately describe the scalar attributes without distortions. Therefore, we use zombies only for interpolation, while for relaxation we only use living points. After an editing operation is completed, all zombies will be deleted from the representation. Figure 13 illustrates our dynamic re-sampling method for a very large deformation that leads to a substantial increase in the number of sample points. While the initial plane consists of 40,000 points, the final model contains 432,812 points, clearly demonstrating the robustness and scalability of our method in regions of extreme surface stretch.

5.3 Downsampling

Apart from lower sampling density caused by surface stretching, deformations can also lead to an increase in sampling density, where the surface is squeezed. It might be desirable to eliminate samples in such regions while editing, to keep the overall sampling

distribution uniform. However, dynamically removing samples also has some drawbacks. Consider a surface that is first squeezed and then stretched back to its original shape. If samples get removed during squeezing, surface information such as color will be lost, which leads to increased blurring when the surface is re-stretched. Thus instead of dynamic sample deletion we perform an optional "garbage collection" at the end of the editing operation. To reduce the sampling density, we use the iterative simplification method introduced in [Pauly et al. 2002], which successively contracts point pairs according to a quadric error metric.

6 RESULTS AND DISCUSSION

We have implemented the algorithms presented in the previous sections as plug-ins to Pointshop3D, a publicly available texturing and sculpting tool for point-sampled models [Zwicker et al. 2002]. The integration of our shape modeling functionality with the appearance modeling tools provided by Pointshop3D has been very smooth, since both are based on unstructured point clouds as the fundamental geometry representation.

Figure 14 shows boolean operations and deformation in connection with displacement fields to illustrate the potential for multiresolution surface modeling [Zorin et al. 1997]. The original model has been low-pass filtered using a large scale parameter in the MLS kernel function (see Section 2) to create a base domain for the multiresolution decomposition. Detail vectors can easily be computed by projecting the points of the original model onto the base domain using the MLS projection operator. These detail vectors can then be added back to the deformed base domain to enable intuitive editing semantics in a multiresolution sense. The same projection method can also be used to build a spectral decomposition of a point-sampled surface to apply various filtering operations similar to [Guskov et al. 1999] or [Pauly and Gross 2001].

Figure 15 shows the model of the Octopus, whose shape has been created from a single sphere entirely using the free-form deformation tool (see also Figure 1 (b)). First a global deformation has been applied to create an ellipsoid, then the tentacles have been pulled out using a similar deformation as the one shown in Figure 13. The eyes and suckers have been added using displacement mapping and the model has finally been textured using the paint tool of Pointshop3D. This example illustrates that dynamic re-sampling is essential when dealing with large deformations. The initial sphere contains 69,706 points, while the final model contains 295,220 points.

Figure 16 illustrates that the sharp intersection curves created by a boolean difference operation are well preserved during a subsequent deformation. The final model of the spiral consists of 69,268 points.

In Figure 17 the handle of the coffee mug (222,955 points) has been created using the deformation tool in connection with a boolean union operation and particle-based blending as described in Section 4.1. The interior of the mug has been cut out with a boolean difference and the dragon head has been added using a union operation. Finally, the Siggraph logo has been embossed and the model has been textured.

Figure 18 shows boolean operations with two sparsely and non-uniformly sampled models, illustrating that our methods work well for a wide class of input models. First the skull has been deformed to better match the shape of the head. Then a boolean difference of the head with a plane and a subsequent union with the skull have been performed, yielding a model of 25,020 points.

Figure 19 demonstrates that our modeling framework is well suited for scanned surfaces. The smoothing effect of the MLS projection has been used to de-noise the input data set. After this minimal pre-processing, free-form deformations and boolean operations can be directly applied to the acquired point cloud. This example also illustrates a typical cut-and-paste operation. The ear has been extracted from the Max Planck model and pasted onto the scanned surface using a boolean union, resulting in a surface consisting of 100,269 points.

Implementation. The central computational primitive used in our algorithms is closest points query: Given a point $\mathbf{x} \in \mathbb{R}^3$ find the point(s) $\mathbf{p} \in P$ such that $\|\mathbf{x} - \mathbf{p}\|$ is minimal. For example, the MLS projection operator uses closest point queries to determine the initial estimate for the base point $\mathbf{q}$ in Equation (2) and to collect all points that contribute to the least-squares optimization.

We use kd-trees for these spatial query tasks, since they feature efficient construction and fast query response time. For example, building a kd-tree of 300,000 points takes 0.23 seconds, while a query for the ten closest points takes between 4.5 and 6.2 microseconds (on a 2.8 GHz Pentium IV). Kd-trees are static data structures, i.e. dynamically updating point positions and inserting or deleting points is computationally expensive. Yet the dynamic sampling paradigm (Section 5) is at the heart of our processing methods. A closer look at our algorithms reveals, however, that this does not pose serious problems from an implementational point of view:

- For boolean classification (Section 3.1), we are dealing with static objects that can be positioned relative to each other by the user. Since the resulting affine transformation can be incorporated into the query, no updates of the kd-tree are necessary.
- For free-form deformation (Section 4) we need closest point queries to evaluate the distance functions d_j (see Equation 4) for computing the scale factor t. This is done once at the beginning of the modeling session, so no dynamic updates of the kd-tree are required while the user interactively deforms the surface.
- For collision detection (Section 4.1) we need to evaluate the classification predicate Ω for dynamically varying point locations. Since performance is most critical, we restrict our algorithm to only detect collisions of the deformable region with the zero-region, which is described by a static point cloud. Thus our current method cannot handle collisions of the deformable region with itself.
- When dynamically sampling the surface (Section 5), we need closest point information for samples in the deformable region to apply the relaxation and interpolation filters. Since these samples vary in position and new samples are inserted dynamically, we cannot use the kd-tree efficiently. Therefore we compute these neighborhood relations at the beginning of the interaction and cache them during deformation (see also [Linsen and Prautzsch 2002]). When new points are inserted into the point cloud, we can dynamically propagate existing neighborhood information without re-computing the correct spatial relations.

Thus updates to the kd-tree are only required at the end of an editing session, but not during interactive modeling. Note that additional structural information such as a mesh connectivity graph would not be of much use for the above computations. Only for relaxation and interpolation would the explicit neighborhood information be beneficial. However, keeping the internal mesh data structures consistent, while new samples are inserted continu-

ously and old samples change their relative position, would be cumbersome and inefficient. Additionally, the structural simplicity of the point-based representation leads to a much simpler implementation and avoids most of the numerical instabilities that occur when triangles meshes are re-meshed or boolean operations are applied. Nevertheless, all algorithms described in this paper can also be applied to triangle meshes, which can be understood as point clouds with a special connectivity graph.

Performance. Currently our point-based shape modeling system can handle models of up to 1 million samples at interactive rates on a desktop PC. When collision detection is enabled, the framerate is reduced by 10% - 30% depending on the specific configuration. Central to achieving such high performance is the simplicity and compactness of our point-based surface representation, which leads to concise and efficient algorithms, reduced memory traffic and enhanced cache performance. We also exploit spatial and temporal coherence during boolean classification (see Section 3.1) and collision detection (Section 4.1). In our current implementation a limiting factor regarding performance is rendering speed. We use the purely software-based, high quality surface splatting renderer of Pointshop3D (see [Zwicker et al. 2001]). Our experiments show that during deformation more than 50% of the total computation time are devoted to rendering, so we expect significant speed-ups if instead more efficient software renderers (e.g. [Botsch et al. 2002]) or hardware supported renderers (e.g. [Rusinkiewicz and Levoy 2000]) are used.

7 CONCLUSIONS & FUTURE WORK

We have presented a shape modeling system based on a single universal geometry representation that consists of an unstructured cloud of sample points, which lie on the surface of an object. This raw point cloud is enhanced by an efficient tool to locally estimate the signed distance function of the underlying surface. All the demonstrated functionality is built on top of this hybrid representation. The main features of our system are the integration of boolean operations and free-form deformations, a dynamic sampling framework to handle large model deformations, a technique to represent, sample and render sharp corners in point-sampled models, and an efficient method to detect self-intersection for explicit topology control. Due to the structural simplicity of our representation, all these algorithms are efficient, easy to implement and applicable to a wide range of input models.

In the future we want to investigate ways to extend our surface model to handle more complex shapes, including hairy or furry models, or natural objects such as plants. We also believe that dynamic simulations of virtual materials can easily be incorporated into our system and that our deformation method is well suited for animation.

Acknowledgements

This work is supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation, financed by ETH Zurich and the German Science Foundation (DFG).

References

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T.. Computing and rendering point set surfaces. In *IEEE TVCG, to appear*.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2001. Point set surfaces. In *Proc. of IEEE Visualization 2001*.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Proceedings of SIGGRAPH 1984*.

BOTSCH, M., WIRATANAYA, A., AND KOBBELT, L. 2002. Efficient high quality rendering of point sampled geometry. In *Eurographics Workshop on Rendering 2002*.

CARR, J., BEATSON, R., CHERRIE, J., MITCHELL, T., FRIGHT, W., MCCALLUM, B., AND EVANS, T. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of SIGGRAPH 2001*.

CHANG, Y. AND ROCKWOOD, A. 1994. A generalized de casteljau approach to 3d free-form deformation. In *Proceedings of SIGGRAPH 1994*.

DOCARMO, M. P. 1976. *Differential Geometry Of Curves and Surfaces*. Prentice Hall.

FARIN, G. 2002. *Curves and Surfaces for CAGD, 5rd ed.*. Academic Press.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 1999*.

HOFFMANN, C. M. 1989. *Geometric and solig modeling: An introduction*. Morgan Kaufmann.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH 1992*.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH 1998*.

KOBBELT, L. P., BAREUTHER, T., AND SEIDEL, H.-P. 2000. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3).

KRISHNAN, S. AND MANOCHA, D. 1997. An efficient surface intersection algorithm based on lower-dimensional formulation. In *ACM Transactions on Graphics 16(1)*.

LEVIN, D.. Mesh-independent surface interpolation. In *Advances in Comp. Math., to appear*.

LEVIN, D. 1998. The approximation power of moving least-squares. In *Math. Comp. Vol 67, No. 224*.

LEVOY, M. AND WHITTED, T. 1985. The use of points as a display primitive. In *UNC-Chapel Hill Computer Science Tech. Report #85-022, 1985*.

LINSEN, L. AND PRAUTZSCH, H. 2002. Fan clouds - an alternative to meshes. In *Proc. of Dagstuhl Seminar on Theo. Foundations of Computer Vision - Geometry, Morphology, and Computational Imaging*.

MUSETH, K., BREEN, D., WHITAKER, R., AND BARR, A. 2002. Level set surface editing operators. In *ACM Transactions on Graphics, Vol. 21(3), July 2002*.

OSHER, S. AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, New York.

PAULY, M. AND GROSS, M. 2001. Spectral processing of point-sampled geometry. In *Proceedings of SIGGRAPH 2001*.

PAULY, M., GROSS, M. H., AND KOBBELT, L. P. 2002. Efficient simplification of point-sampled surfaces. In *Proc. of IEEE Visualization 2002*.

RUSINKIEWICZ, S. AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000*.

SEDERBERG, T. AND PARRY, S. 1986. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH 1986*.

SINGH, K. AND FIUME, E. 1998. Wires: A geometric deformation technique. In *Proceedings of SIGGRAPH 1998*.

SZELISKI, R. AND TONNESEN, D. 1992. Surface modeling with oriented particle systems. In *Proceedings of SIGGRAPH 1992*.

TURK, G. 1992. Re-tiling polygonal surfaces. In *Proceedings of SIGGRAPH 1992*.

VELHO, L., GOMES, J., AND FIGUEIREDO, L. H. 2002. *Implicit Objects in Computer Graphics*. Springer Verlag, New York.

WELCH, W. AND WITKIN, A. 1994. Free form shape design using triangulated surfaces. In *Proceedings of SIGGRAPH 1994*.

WITKIN, A. AND HECKBERT, P. 1994. Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH 1994*.

ZORIN, D. AND SCHRÖDER, P. 2000. *Subdivision for Modeling and Animation*. SIGGRAPH 2000 Course Notes.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH 1997*.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop3d: An interactive system for point-based surface editing. In *Proceedings of SIGGRAPH 2002*.

ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. 2001. Surface splatting. In *Proceedings of SIGGRAPH 2001*.

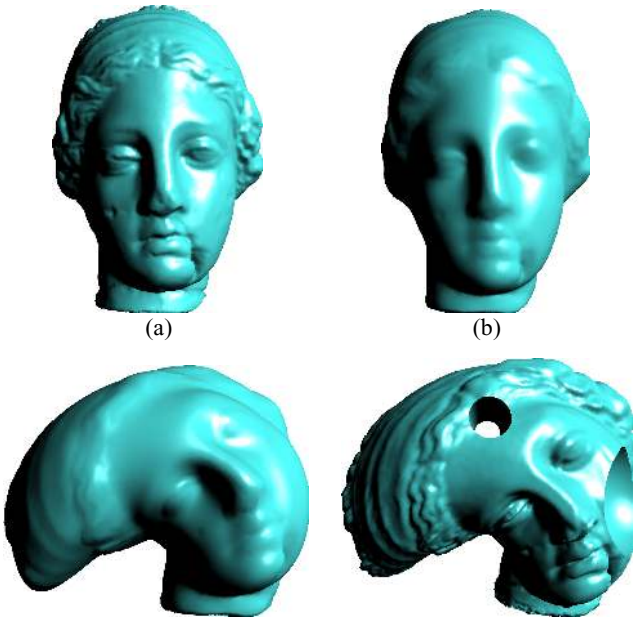


Figure 14: Multiresolution modeling on the Igea model. (a) original model, (b) smooth base domain, (c) deformed base domain, (d) final model, where the displacement coefficients have been scaled by a factor of two to achieve an enhancement effect. Additionally, two boolean difference operations have been applied.

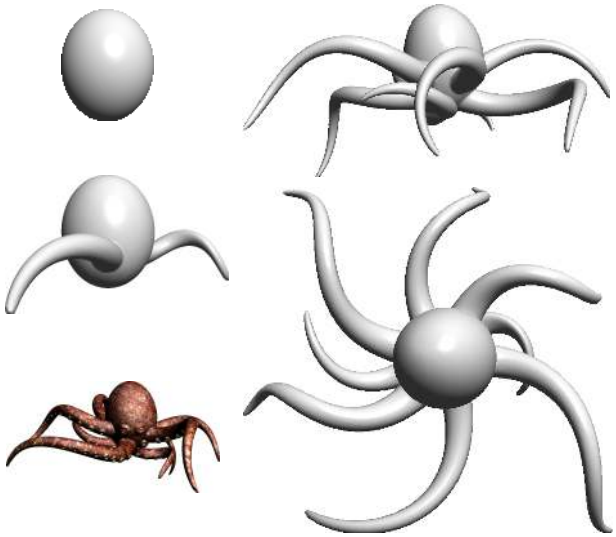


Figure 15: Creating an Octopus from a sphere using the deformation tool with dynamic sampling (see Figure 1 (b)).

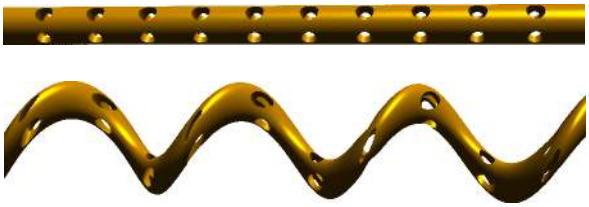


Figure 16: Combination of boolean operations and subsequent deformation.

Figure 17: Creating the Siggraph coffee mug using boolean operations, deformation, collision detection and particle-based blending (see Figure 1 (c)).

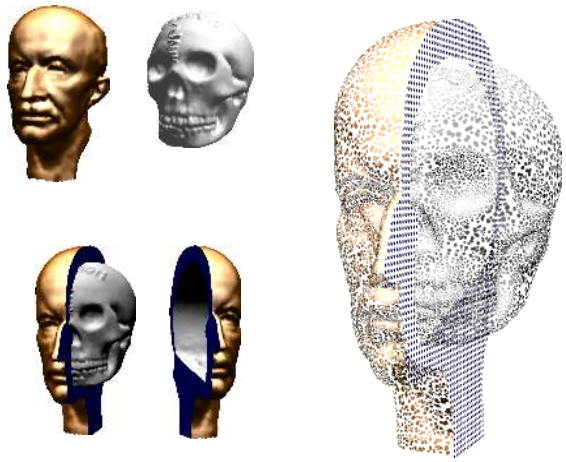


Figure 18: Boolean operations of the Max Planck bust, a plane and the skull model (see Figure 1 (a)).

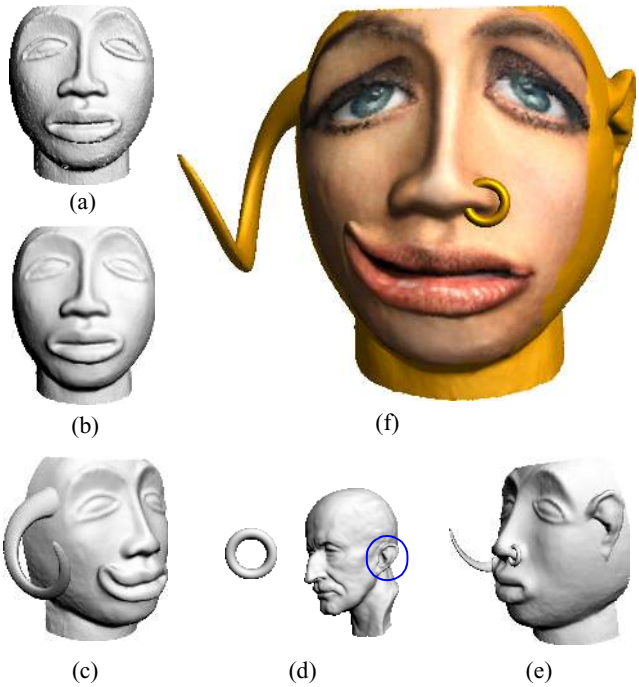


Figure 19: Boolean operations and deformations on scanned data: (a) noisy range scan, (b) surface smoothed by MLS projection, (c) surface after local deformations, (d) objects used for boolean union, (e) surface after boolean union, (f) final textured surface.