Pointshop 3D: An Interactive System for Point-Based Surface Editing

Matthias Zwicker Mark Pauly Oliver Knoll Markus Gross
ETH Zürich

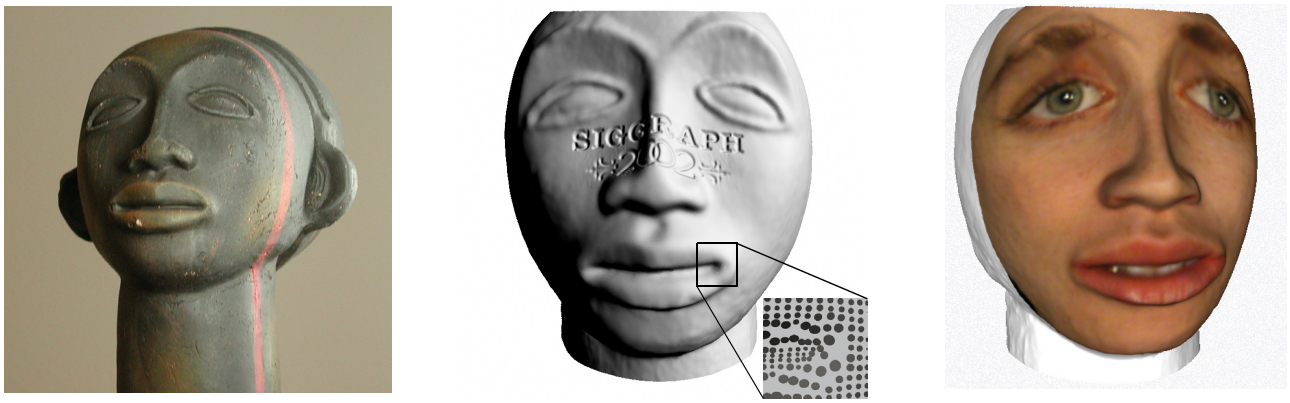


Figure 1: 3D content creation: Scanning of a physical model (left). Editing of the point-sampled object: Carving (middle), texturing (right).

Abstract

We present a system for interactive shape and appearance editing of 3D point-sampled geometry. By generalizing conventional 2D pixel editors, our system supports a great variety of different interaction techniques to alter shape and appearance of 3D point models, including cleaning, texturing, sculpting, carving, filtering, and resampling. One key ingredient of our framework is a novel concept for interactive point cloud parameterization allowing for distortion minimal and aliasing-free texture mapping. A second one is a dynamic, adaptive resampling method which builds upon a continuous reconstruction of the model surface and its attributes. These techniques allow us to transfer the full functionality of 2D image editing operations to the irregular 3D point setting. Our system reads, processes, and writes point-sampled models without intermediate tesselation. It is intended to complement existing low cost 3D scanners and point rendering pipelines for efficient 3D content creation.

Keywords: 3D Content Creation, Point-Based Graphics, Surface Painting, Surface Sculpting, Texture Mapping, Parameterization

1 INTRODUCTION

When 2D digital photography became instrumental, it immediately created the need to efficiently edit and to interactively improve the quality of digital images. Hence, considerable effort has been devoted to the development of such systems, both for the private and for the professional user of digital cameras. This conventional photo editing software includes a variety of individual tools ranging from simple artifact removal or paint brushes to highly specialized image effect filters. The most popular package is undoubtedly *Adobe's Photoshop*, providing a set of powerful tools for user guided alteration of 2D image data.

In recent years advances in 3D digital photography spawned scanning systems that acquire *both* geometry *and* appearance of real-world objects. A major application for such 3D range cameras is for instance the ready creation of 3D internet content for e-commerce applications. However, the process of 3D model production is often quite tedious and requires a variety of different techniques including registration of raw scans, resampling, filtering, sculpting, or re-texturing. The early stages of processing of 3D photos frequently produce 3D point clouds, which are most often converted into triangle meshes for further modeling. In this paper we present an interactive 3D photo editing system which is entirely based on points. It takes an irregular point-sampled model as an input, provides a set of tools to edit geometry *and* appearance of the model, and produces a point-sampled object as an output.

Conceptually, 2D photo editing systems are based on *pixels* as the major image primitive. As a consequence, all editing tools operate on subsets of image pixels, often making heavy use of adjacency and parameterization. Despite the multilayered structure of an image, the regular sampling lattice makes many pixel operations simple and efficient. Furthermore, pixel processing is mostly carried out on color or transparency channels changing appearance attributes of the image only. Geometry is typically less important. If at all, range layers are manipulated by converting them into intensity fields.

In this work we generalize 2D photo editing to make it amenable to 3D photography. While existing 3D geometry-oriented modeling, painting or sculpting systems are either based on polynomials [Alias Wavefront 2001], triangle meshes [Agrawala et al. 1995, Right Hemisphere 2001], implicits [Pedersen 1995, Perry and Frisken 2001], or images [Oh et al. 2001], our approach is completely different in spirit. It is purely founded on irregular 3D points as powerful and versatile 3D image primitives. By generalizing 2D *image pixels* towards 3D *surface pixels* (surfels [Szeliski and Tonnesen 1992, Pfister et al. 2000]) we combine the functionality of 3D geometry based sculpting with the simplicity and effectiveness of 2D image based photo editing.

Point samples provide an abstraction of geometry and appearance, since they discretize surface position and texture. However, as opposed to triangle meshes, they do not store information about local surface connectivity. Unlike 2D pixels, the absence of local topology in combination with the irregularity of the sampling pattern poses great challenges to the design of 3D photo editing tools. We found that the two key ingredients for such tools are *interactive parameterization* and *dynamic resampling*. For instance, surface texturing or carving both demand a flexible parameterization of the point cloud. In addition, points discretize geometry and appearance attributes at the same rate. Thus, fine-grain surface detail embossing with a high resolution depth map can lead to heavy aliasing and requires a dynamic adaptation of the sampling rate.

In the following, we will present a set of methods to solve the problems stated above and integrate them into a versatile system. Specifically, our paper makes the following contributions:

Interactive parameterization (Section 3): By extending prior work on triangle meshes [Levy 2001] we designed a novel method for distortion minimal parameterization of point clouds. The algorithm allows for constraints and enables users to interactively adapt the parameterization to input changes. A multigrid approach accomplishes robust and efficient computation.

Dynamic resampling (Section 4): As a prerequisite, changes of the sampling rate demand a continuous reconstruction of the model surface and of its attributes. To this end, we introduce a novel surface representation based on a parameterized scattered data approximation. In addition, we propose a method which dynamically adapts the number of samples to properly represent fine geometric or appearance details. We combined our sampling strategy with existing texture antialiasing techniques for point-sampled geometry [Zwicker et al. 2001].

Editing framework (Section 5): Our system provides a unified conceptual framework to edit 3D models. It supports a great variety of individual tools to alter the geometry and appearance of irregular point-sampled geometry. The scope of possible operations goes well beyond the functionality of conventional 2D photo editing systems. We implemented re-texturing, sculpting, embossing, and filtering, however, new effect filters can be added very easily. Overall, our system combines the efficiency of 2D photo editing with the functionality of 3D sculpting systems.

Pointshop 3D is *not* intended to be a point-based modeling system. As such, editing of the surface geometry is confined to normal displacements and to moderate changes of the surface structure only. It is rather designed to complement low cost scanning devices [Eyetronics 2001] and point-based 3D viewers [Rusinkiewicz and Levoy 2000, Pfister et al. 2000, Arius3D 2001], yielding a powerful pipeline for efficient 3D content creation and display. Pointshop 3D explores the usability of point primitives for surface editing and constitutes an alternative to conventional polygonal mesh or splines based approaches. Since our algorithms are based on k-nearest neighbor search, input data with a substantial amount of noise or highly irregular sampling distribution, e.g., as acquired by multiple merged range scans, can lead to instabilities. In these cases, the raw scans have to be resampled to a clean point cloud, for example using distance fields [Curless and Levoy 1996]. In general, suitable point data can be obtained by sampling isosurfaces of smooth implicit functions, which is easier than extracting a good mesh. Volumetric methods such as MRI or CT scans can also provide clean input data to our system.

2 SYSTEM OVERVIEW

Our editing framework originates from the motivation to provide a wide range of editing and processing techniques for point-sampled 3D surfaces, similar to those found in common photo editing tools for 2D images. To give an overview of our system we will first describe a typical photo editing operation on an abstract level. Then we will explain how these concepts can be transferred to surface editing, commenting on the fundamental differences between images and surfaces. This will serve as a motivation for the techniques and algorithms described in the following sections. We also introduce an operator notation for general editing operations that will be used throughout the paper.

A 2D image I can be considered a discrete sample of a continuous image function containing image attributes such as color or transparency. Implicitly, the discrete image I always represents the continuous image, and image editing operations are performed directly on the discrete image. The continuous function can be computed using a reconstruction operator whenever necessary.

We describe a general image editing operation as a function of an original image I and a brush image B. The brush image is used as a general tool to modify the original image. Depending on the considered operation, it may be interpreted as a paint brush or a discrete filter, for example. The editing operation involves the following steps: First, we need to specify a parameter mapping Φ that aligns the image I with the brush B. For example, Φ can be defined as the translation that maps the pixel at the current mouse position to the center of B. Next, we have to establish a common sampling grid for I and B, such that there is a one-to-one correspondence between the discrete samples. This requires a resampling operation Ψ that first reconstructs the continuous image function and then samples this function on the common grid. Finally, the editing operator Ω combines the image samples with the brush samples using the one-to-one correspondence established before. We thus obtain the resulting discrete image I' as a concatenation of the operators described above:

$$I' = \Omega(\Psi(\Phi(I)), \Psi(B)) . \quad (1)$$

Our goal is now to generalize the operator framework of Equation (1) to irregular point-sampled surfaces, as illustrated in Figure 2.

Formally, we do this by replacing the discrete image I by a point-based surface S. Hence, we represent a 3D object as a set of irregular samples $S = \{s_i\}$ of its surface. Since the samples s_i are a direct extension of image pixels, we will also call them surfels [Szeliski and Tonnesen 1992, Pfister et al. 2000]. As summarized in Table 1, each surfel stores appearance attributes, including color, transparency, or material attributes, and shape attributes, such as position and normal. Let us now consider what effects the transition from image to surface has on the individual terms of Equation (1).

Parameterization Φ. For photo editing, the parameter mapping Φ is usually specified by a simple, global 2D to 2D affine mapping, i.e., a combination of translation, scaling, and rotation. Mapping a manifold surface onto a 2D domain is much more involved, however. In our system, the user interactively selects subsets, or *patches*, of S that are parameterized, as described in Section 3. In general, such a mapping leads to distortions that can-

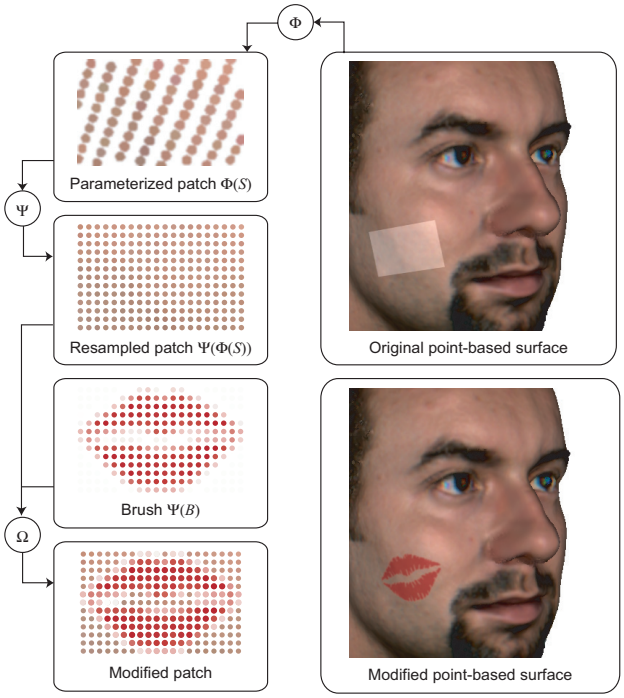


Figure 2: Overview of the operator framework for point-based surface editing.

Table 1: Attributes of a surface sample s_i.

ATTRIBUTE	ABBREVIATION
Position	$\mathbf{x}_i$
Normal	$\mathbf{n}_i$
Color	$\mathbf{c}_i$
Transparency	α_i
Material properties	$\mathbf{m}_i$

not be avoided completely. In Section 3.2, we present an efficient method that automatically minimizes these distortions, and at the same time lets the user intuitively control the mapping.

Resampling Ψ. Images are usually sampled on a *regular* grid, hence signal processing methods can be applied directly for resampling. However, the sampling distribution of surfaces is in general *irregular*, requiring alternative methods for reconstruction and sampling. We apply a scattered data approximation approach for reconstructing a continuous function from the samples, as described in Section 4. We also present a technique for resampling our modified surface function onto irregular point clouds in Section 4.2. A great benefit of our system is that it supports adaptive sampling, i.e., works on a *dynamic* structure. This allows us to concentrate more samples in regions of high textural or geometric detail, while smooth parts can be represented by fewer samples.

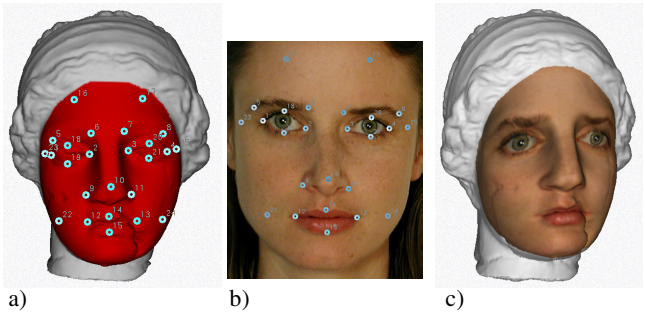
Editing Ω. Once the parameterization is established and resampling has been performed, all computations take place on discrete samples in the 2D parameter domain. Hence we can apply the full functionality of photo editing systems for texturing and texture filtering. However, since we are dealing with texture *and* geometry, the scope of operations is much broader. Additional editing operators include sculpting, geometry filtering and simplification. As will be described in Section 5, all of these tools are based on the same simple interface that specifies a tool by a set of bitmaps and few additional parameters. For example, a sculpting tool is defined by a 2D displacement map, an alpha mask and an intrusion depth.

3 PARAMETERIZATION

In our system, the user interactively selects a subset $\tilde{S}$ of the surface S, which we call a patch. We compute a parameterization of the patch $\Phi: \tilde{S} \rightarrow [0,1] \times [0,1]$ that assigns parameter coordinates $\mathbf{u}_i$ to each point s_i in $\tilde{S}$ and then apply the editing operation on the parameterized patch. The user chooses between two types of interaction schemes to select a patch and compute the parameterization: A selection interaction for large patches, described in Sections 3.1 and 3.2, and a brush interaction for small patches presented in Section 3.3.

3.1 Selection Interaction

In this interaction scheme, the user triggers each step in the evaluation of Equation (1) separately. First, she marks an arbitrary surface patch using a dedicated selection tool and specifies a set of feature points. In a next step, she initiates a constrained minimum distortion parameterization algorithm that uses the feature points, as described in Section 3.2. Then she typically performs a series of editing operations on the parameterized patch, such as filtering or texture mapping. This process is illustrated in Figure 3.



a) b) c)

Figure 3: Selection interaction: a) Patch selection and feature points. b) Texture map with feature points. c) Texture mapping.

3.2 Minimum Distortion Parameterization

We describe a novel algorithm for computing minimum distortion parameterizations of point-based objects. Our approach is based on an objective function, similar to Levy's method for polygonal meshes [Levy 2001]. However, we then derive a discrete formulation for surfaces represented by scattered points without requiring any tesselation. We solve the resulting linear least squares problem efficiently using a multilevel approach by hierarchical clustering of points.

Objective Function. Let us denote a continuous parameterized surface patch by X_S. The patch is defined by a one-to-one mapping $X: [0,1] \times [0,1] \rightarrow X_S \in \mathbb{R}^3$ which for each point $\mathbf{u} = (u, v)^T$ in $[0,1] \times [0,1]$ represents a point $\mathbf{x} = (x, y, z)^T$ on the surface:

$$\mathbf{u} \in [0,1] \times [0,1] \Rightarrow X(\mathbf{u}) = \begin{bmatrix} x(\mathbf{u}) \\ y(\mathbf{u}) \\ z(\mathbf{u}) \end{bmatrix} = \mathbf{x} \in X_S . \qquad (2)$$

The mapping X describes a parameterization of the surface, with $U = X^{-1}$ its inverse. Our method computes a parameterization that optimally adapts to the geometry of the surface, i.e., minimizes metric distortions. Additionally, the user is able to specify a set M of point correspondences between points on the surface $\mathbf{x}_j$ and points in the parameter domain $\mathbf{p}_j$, $j \in M$, to control the mapping. This can be expressed as the following objective function:

$$C(X) = \sum_{j \in M} \{X(\mathbf{p}_j) - \mathbf{x}_j\}^2 + \varepsilon \int_{\Omega} \gamma(\mathbf{u}) d\mathbf{u}, \tag{3}$$

$$\text{where } \gamma(\mathbf{u}) = \int_{\theta} \left(\frac{\partial^2}{\partial r^2} X_{\mathbf{u}}(\theta, r) \right)^2 d\theta, \tag{4}$$

$$\text{and } X_{\mathbf{u}}(\theta, r) = X\left(\mathbf{u} + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \right). \tag{5}$$

The first term in (3) represents the fitting error as the sum of the squared deviations from the user specified data points. The second term measures the smoothness, or distortion, of the parameterization. At each surface point, $\gamma(\mathbf{u})$ integrates the squared curvature of the parameterization in each radial direction using a local polar reparameterization $X_{\mathbf{u}}(\theta, r)$. If $\gamma(\mathbf{u})$ is zero, the parameterization at $\mathbf{u}$ is a so called *polar geodesic map*, which preserves arc length in each radial direction [O'Neill 1966, Welch and Witkin 1994]. With the parameter ε, the user additionally controls the relative weight of the data fitting error and the smoothness constraint. The desired parameterization X can be obtained by computing the minimum of the functional (3). We now describe how to set up and minimize (3) in the discrete case.

Discrete formulation. Given a set of distinct points $\{\mathbf{x}_i\}$ on the surface, our goal is to assign to each point $\mathbf{x}_i$ a point $\mathbf{u}_i$ in the parameter domain, such that the objective function is minimized. In other words, we are solving for the unknown discrete mapping $U{:}\mathbf{x}_i \rightarrow \mathbf{u}_i$ and hence we reformulate (3) by substituting the unknown U for X. Moreover, we assume that the parameterization is piecewise linear, thus the second derivative of U is not defined at the points $\mathbf{x}_i$ in general. As an approximation, we discretize the smoothness criterion by computing at each point $\mathbf{x}_i$ the squared difference of the first derivatives along a set of normal sections. This yields the following objective function $\tilde{C}(U)$:

$$\tilde{C}(U) = \sum_{j \in M} \{\mathbf{p}_j - \mathbf{u}_j\}^2 + \varepsilon \sum_{i=1}^{n} \sum_{j \in N_i} \left(\frac{\partial U(\mathbf{x}_i)}{\partial \mathbf{v}_j} - \frac{\partial U(\mathbf{x}_i)}{\partial \tilde{\mathbf{v}}_j} \right)^2, \tag{6}$$

where n is the number of points in the patch, N_i specifies the set of normal sections, and $\mathbf{v}_j$ and $\tilde{\mathbf{v}}_j$ are unit vectors on the surface given by the normal section.

Directional Derivatives. We compute the directional derivatives $\partial U(\mathbf{x}_i)/(\partial \mathbf{v}_j)$ and $\partial U(\mathbf{x}_i)/(\partial \tilde{\mathbf{v}}_j)$ in (6) as illustrated in Figure 4: At each point $\mathbf{x}_i$, we collect a set $N_i = \{i_1 \ldots i_k\}$ containing the indices of its k nearest neighbors, typically $k = 9$. For each neighbor $\mathbf{x}_j$, $j \in N_i$, we determine the plane P defining the normal section, which is given by the normal $\mathbf{n}_i$ at $\mathbf{x}_i$ and the vector $\mathbf{v}_j = \mathbf{x}_i - \mathbf{x}_j$. We then choose the two points $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$, $\alpha, \beta \in N_i$, such that the angles between $\mathbf{v}_\alpha = \mathbf{x}_\alpha - \mathbf{x}_i$ and $\mathbf{v}_\beta = \mathbf{x}_\beta - \mathbf{x}_i$ and the plane P are minimal, while the angles between $\mathbf{v}_j$ and $\mathbf{v}_\alpha$, and between $\mathbf{v}_j$ and $\mathbf{v}_\beta$ are bigger than 90 degrees. Otherwise, the normal section crosses the boundary of the patch, hence we ignore it. This procedure is sufficient to handle patches with boundaries. Next, we compute the direction $\tilde{\mathbf{v}}_j$ of the intersection line of the plane P and the plane given by $\mathbf{x}_i$, $\mathbf{x}_\alpha$, and $\mathbf{x}_\beta$ (see Figure 4).

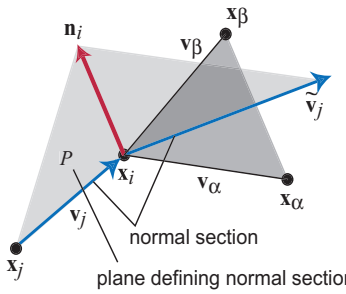


Figure 4: Computing directional derivatives using normal sections.

Assuming a piecewise linear mapping U between $\mathbf{x}_i$ and $\mathbf{x}_j$, the directional derivative at $\mathbf{x}_i$ along $\mathbf{v}_j$ is simply

$$\frac{\partial}{\partial \mathbf{v}_j} U(\mathbf{x}_i) = \frac{\mathbf{u}_i - \mathbf{u}_j}{\|\mathbf{v}_j\|}. \tag{7}$$

Likewise, we compute the derivative along $\tilde{\mathbf{v}}_j$ as described in [Levy 2001, Levy and Mallet 1998] by assuming a piecewise linear mapping on the triangle defined by the points $\mathbf{x}_i, \mathbf{x}_\alpha, \mathbf{x}_\beta$. This leads to a linear expression of the form

$$\frac{\partial}{\partial \tilde{\mathbf{v}}_j} U(\mathbf{x}_i) = a_i \mathbf{u}_i + a_\alpha \mathbf{u}_\alpha + a_\beta \mathbf{u}_\beta, \tag{8}$$

where the coefficients a_i, a_α, a_β are determined by the points $\mathbf{x}_i, \mathbf{x}_\alpha, \mathbf{x}_\beta$, as presented in detail in [Levy and Mallet 1998].

In contrast to Floater's shape preserving weights [Floater and Reimers 2001, Floater 1997], our method can be used as an *extrapolator*, since we do not enforce the coefficients of (8) to be a convex combination. As a consequence, we do not have to specify a convex boundary. Still, our method has the reproduction property: If all points lie in a plane and at least three or more points obeying an affine mapping are given as fitting constraints, the resulting parameterization will be an affine mapping, too. Moreover, we do not need to construct a local triangulation at each point as in [Floater and Reimers 2001] to establish our constraints. Note that the parameterization is not guaranteed to be bijective. It is rather left to the user to select a suitable patch and appropriate point correspondences to obtain the desired mapping.

Multigrid Least Squares Solver. The discrete objective function of (6) is now a sum of squared linear relations of the general form

$$\tilde{C}(U) = \sum_j \left(\mathbf{b}_j - \sum_{i=1}^{n} \begin{bmatrix} a_{j,i} & 0 \\ 0 & a_{j,i} \end{bmatrix} \mathbf{u}_i \right)^2 = \|\mathbf{b} - A\mathbf{u}\|^2, \tag{9}$$

where $\mathbf{u}$ is a vector of all unknowns $\mathbf{u}_i = (u_i, v_i)^T$ and the coefficients $a_{j,i}$ result from (7) and (8). We compute this linear least squares problem using normal equations and conjugate gradient methods [Ashby et al. 1990]. The convergence of such iterative solvers can be further accelerated by efficient multilevel techniques. To this end, we designed a hierarchical strategy as illustrated in Figure 5. In a top-down pass, we contract the system by recursively clustering the unknowns $\mathbf{u}_i$. The clustering is driven

by the spatial proximity of the corresponding surface points $\mathbf{x}_i$, and each cluster yields one unknown on the current level. In a bottom-up pass, we solve (9) starting with the coarsest level. The solution is then prolonged by assigning it as an initial value to the next higher resolution level. This process is repeated recursively up to the original resolution.

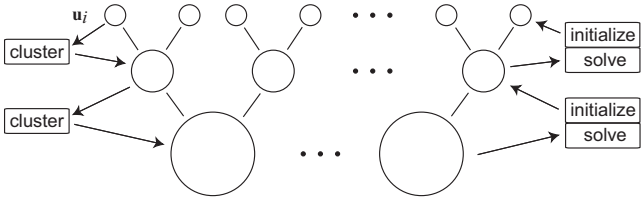


Figure 5: Multilevel scheme to solve Equation (9).

Figure 3 depicts an example of our parameterization technique. We first apply the minimum distortion parameterization on a complex surface patch, controlling the mapping by specifying a set of corresponding feature points. Finally, we perform a texture mapping operation.

3.3 Brush Interaction

Here, the user has to select a brush image and an editing operation first. Then he moves the brush over the surface while continuously triggering painting events. Each painting event corresponds to one evaluation of Equation (1), which is performed as follows (see Figure 6): We center the brush image at the user defined location on the surface, typically at the current mouse position. We then align the brush with the tangent plane at this point and orthogonally project the surface points $\mathbf{x}_i$ onto the brush image plane. For each point having its projection inside the brush image, we assign the resulting coordinates in the brush image plane as its parameter coordinates, yielding $\mathbf{u}_i = \Pi(\mathbf{x}_i)$, where Π denotes the orthogonal projection. This process combines the patch selection and parameterization step using a simple projection. It is therefore suitable only for small patches with limited curvature. Finally we apply the pre-selected editing operation, as illustrated in Figure 6c.

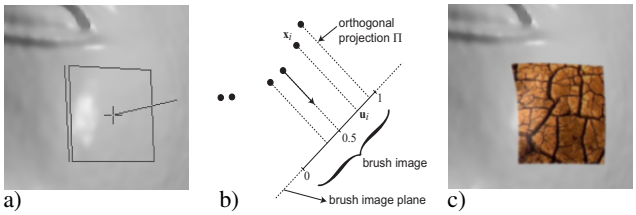


Figure 6: Brush interaction: a) Aligning the brush tool to the tangent plane. b) Patch selection and parameterization by orthogonal projection. c) Editing operation, e.g. texture mapping.

4 RESAMPLING

Given a set of sample points S representing some parameterized, continuous surface X_S, the resampling operator strives to generate a new sample $S' = \Psi(S)$ of the same surface. The challenge of this operation is to minimize information loss while avoiding sampling artifacts. Resampling consists of two separate steps: First, the reconstruction step should provide a smooth, accurate approximation of the continuous surface X_S, i.e., of all its shape and appearance attributes as in Table 1. To avoid aliasing, the actual sampling step should then properly band-limit X_S before evaluating it at the new sampling locations.

4.1 Reconstruction

As described in Section 3, during a typical editing session the user repeatedly modifies the surface parameterization. Hence a paramount objective of the surface reconstruction procedure is that it can be quickly recomputed under changes of the parameterization. We therefore apply the following approach consisting of two stages: first, we perform a local surface fitting step, followed by a parameter matching step. The local fitting procedure is independent of the global parameterization computed in Section 3, hence it can be performed as a preprocess. The parameter matching step then uses the global parameterization as a common frame of reference for the local fits, and blends them to a smooth surface. We present two alternative techniques to perform the matching step: For general patches being parameterized using our method from Section 3.2, we develop an optimization based technique. For the parameterization by projection approach described in Section 3.3, we use a more efficient, analogous matching by projection technique.

Local Surface Fitting. At each point $\mathbf{x}_i$, we compute a local approximation of the surface using its k nearest neighbors, denoted by the index set N_i. This requires a local parameterization of the neighbors, which we compute by projecting the points $\mathbf{x}_j$, $j \in N_i$, onto tangent plane at $\mathbf{x}_i$. We denote the local parameter coordinates of the $\mathbf{x}_j$ by $\mathbf{t}_j^i$. As a local surface approximation, we compute polynomial fitting functions $\phi_i(\mathbf{t}^i)$ using a scheme similar to [Welch and Witkin 1994]. However, it turned out that for our purpose a linear fit, i.e., the tangent plane, is sufficient. We then compute a reconstruction kernel $r_i(\mathbf{t}^i)$ that will be used to blend the local fits. It can be interpreted as a weight indicating the confidence that the fitting function accurately represents the surface. Currently, we use radially symmetric Gaussians centered at $\mathbf{t}_i^i$. Their variance σ_i is determined by computing the radius d_i of the smallest circle containing the $\mathbf{t}_j^i$, where $j \in N_i$. Typically, we choose $k = 9$ and $\sigma_i = 1.5 \cdot d_i$.

Parameter Matching by Optimization. This step brings all local parameterizations into one common frame of reference. For each local parameterization, we look for a mapping φ_i: $\mathbf{t}^i \to \mathbf{u}$ such that the local parameter coordinates $\mathbf{t}_j^i$ of a point $\mathbf{x}_j$ match its patch parameter coordinates $\mathbf{u}_j$ computed in Section 3.2, hence $\mathbf{u}_j = \varphi_i(\mathbf{t}_j^i)$, $j \in N_i$. In our method, we restrict the mappings φ_i to be affine and compute them by minimizing

$$\sum_{j \in N_i} (\mathbf{u}_j - \varphi_i(\mathbf{t}_j^i))^2, \tag{10}$$

which is again a linear least squares problem. Since both the local parameterization and the patch parameterization are smooth in a neighborhood N_i, local affine mappings φ_i provide a sufficient approximation and have lead to good results in our system.

Instead of applying this two-step approach with a local fit followed by parameter matching, we could also directly compute the fitting functions in the patch parameter domain. However, our scheme is more efficient when the global parameterization changes often. We then have to recompute the parameter matching only, instead of recomputing the fitting functions and the reconstruction kernels.

Parameter Matching by Projection. In Section 3.3, we compute the parameter coordinates $\mathbf{u}_i$ of a point $\mathbf{x}_i$ as $\mathbf{u}_i = \Pi(\mathbf{x}_i)$, where Π denotes an orthogonal projection. In the

same way, we can then project the fitting functions ϕ_i to the patch parameter domain, i.e., $\mathbf{u} = \Pi(\phi_i(\mathbf{t}^i))$, thus $\varphi_i(\mathbf{t}^i) = \Pi(\phi_i(\mathbf{t}^i))$. Inverting this projection amounts to ray-tracing the fitting functions. For linear basis functions, this can be implemented by an efficient scan conversion.

Blending the Fitting Functions. After establishing the mappings φ_i from the local parameterizations to the patch parameter domain, we define fitting functions $\tilde{\phi}_i(\mathbf{u})$ and reconstruction kernels $\tilde{r}_i(\mathbf{u})$ in the patch parameter domain as $\tilde{\phi}_i(\mathbf{u}) = \phi_i(\varphi_i^{-1}(\mathbf{u}))$ and $\tilde{r}_i(\mathbf{u}) = r_i(\varphi_i^{-1}(\mathbf{u}))$. We now obtain a continuous surface function $X_S(\mathbf{u})$ as the weighted sum

$$X_S(\mathbf{u}) = \left(\sum_{i=1}^{n} \tilde{\phi}_i(\mathbf{u}) \tilde{r}_i(\mathbf{u}) \right) \Big/ \left(\sum_{i=1}^{n} \tilde{r}_i(\mathbf{u}) \right). \quad (11)$$

of fitting functions $\tilde{\phi}_i$ and reconstruction kernels $\tilde{r}_i$.

Our approach is similar in spirit to the construction of point set surfaces introduced in [Alexa 2001], in that both methods use local parameterizations and polynomials to approximate the surface. However, instead of implicitly defining the surface by a projection operator, we blend the local approximations using a global parameterization.

In our system, we use linear fitting functions for the surface position, and constants for the other surface attributes listed in Table 1. We illustrate curve reconstruction using constant and linear fitting functions in Figure 7. In Figure 7a, the constant fitting functions simply reproduce the data points, whereas in 7b, linear fitting functions result in straight lines aligned with the tangential directions at the data points. Clearly, linear functions lead to a more accurate approximation of the data points (Figure 7c).

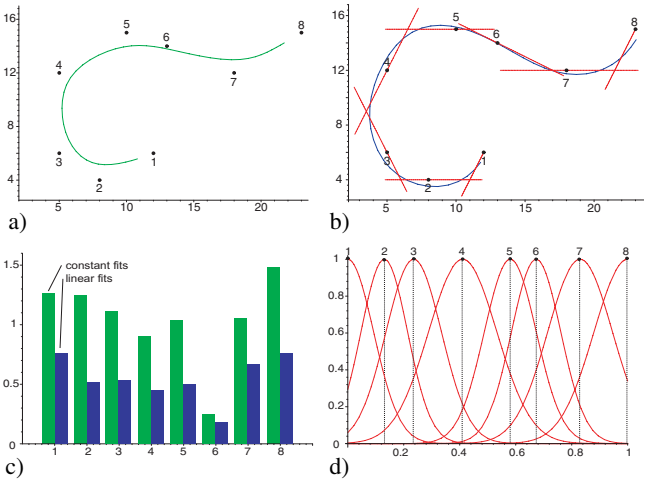


Figure 7: Curve reconstruction: a) Reconstruction with constant fitting functions. b) Reconstruction with linear fitting functions. c) Absolute error at the data points. d) The reconstruction functions in the parameter domain.

4.2 Sampling

The actual sampling includes two aspects: First we need to find a suitable resampling operator Ψ that specifies the location of the new samples. Then, we evaluate the surface function according to the new sampling distribution.

Resampling Operators. We provide three resampling operators specifying different resampling distributions, i.e., sets of new sampling locations $\{\mathbf{u}_i\}$ in the patch parameter domain:

Brush Resampling Ψ_S. In this method, we use the original surface points as the resampling grid. Hence we have to resample the brush, yielding a new sample of the brush $B_S = \Psi_S(B)$. Resampling the surface conceptually results in the same sample $S = \Psi_S(S)$, therefore it is not necessary to perform this operation. The advantage of this method is that we do not have to insert any new surface points, and there is no information loss in S due to resampling.

Surface Resampling Ψ_B. In many operations, such as texture mapping, we want to resample the surface at the sampling distribution of the brush B that represents the texture, avoiding any loss in texture quality. Hence we generate a new sample of the surface $S_B = \Psi_B(S)$. Since the brush is often sampled on a regular grid, the evaluation of the surface function can be optimized using incremental calculations, e.g., for evaluating the Gaussian weight functions and polynomial fitting functions [Zwicker et al. 2001].

Adaptive Resampling Ψ_A. If the sampling density of the surface or the brush varies significantly, it occurs that in some areas in a patch the surface sampling distribution is finer, and in others the brush sampling density. In this case, both operators Ψ_S and Ψ_B fail to preserve detail of either the brush or the surface. Therefore we propose a simple adaptive resampling operator Ψ_A that locally decides whether to use samples of S or of B. The decision is based on the comparison of the radii of the Gaussian reconstruction functions in S and B, since these radii directly correspond to the local sampling density.

Band-Limiting the Surface Function. The goal of this process is to avoid aliasing artifacts when evaluating the surface function at the resampling grid. This can be achieved by properly band-limiting the continuous function before sampling. In a regular signal processing framework, band-limiting is performed by convolving the function with a suitable low-pass filter. Our approach is inspired by signal processing, approximating this procedure with irregular sampling distributions, however. Given a resampling operator Ψ specifying a set of new sampling locations $\{\mathbf{u}_i\}$, we first compute corresponding new reconstruction kernels $\bar{r}_i(\mathbf{u})$ as described in Section 4.1. To sample the surface attributes of a point s_j, we approximate the convolution of the surface function X_S with the reconstruction function $\bar{r}_j$ and evaluate it at $\mathbf{u}_j$:

$$s_j = \left(\sum_{i=1}^{n} \phi_i(\mathbf{u}_j) \rho_i(\mathbf{u}_j) \right) \Big/ \left(\sum_{i=1}^{n} \rho_i(\mathbf{u}_j) \right) \approx (X_S \otimes \bar{r}_j)(\mathbf{u}_j), \quad (12)$$

where $\rho_i(\mathbf{u}) = \tilde{r}_i(\mathbf{u}) \otimes \bar{r}_j(\mathbf{u})$ is also called a resampling filter. With Gaussian weight functions, the resampling filter can be computed explicitly [Zwicker et al. 2001]. Note that this resampling procedure is applied to all surface attributes, such as color, position, normal, etc.

5 SURFACE EDITING

The resampling method of Section 4 provides samples of the surface $S_\Psi = \Psi(\Phi(S))$ and of the brush $B_\Psi = \Psi(B)$ with identical sampling distribution. We can thus combine the two by applying an editing operator directly on the discrete coefficients. Note that both S_Ψ and B_Ψ represent all the surfel attributes of

Table 1. Depending on the intended functionality, an editing operator will then manipulate a subset of these surface attributes, such as texture or material properties. In the following we will describe some of the editing operators that we have implemented in our system. A prime will denote the manipulated attributes, e.g., $\mathbf{x}_i'$ describes the position of a surfel of the edited surface. Quantities that stem from the brush B_Ψ are marked with a bar, e.g., $\bar{\mathbf{c}}_i$ is the color of a brush sample. All other variables are part of the surface function S_Ψ.

Painting. Painting operations modify surface attributes by alpha-blending corresponding surface and brush coefficients. For example, the surface texture can be altered by applying the painting operator on the color values, i.e., $\mathbf{c}_i' = \bar{\alpha}_i \cdot \mathbf{c}_i + (1-\bar{\alpha}_i) \cdot \bar{\mathbf{c}}_i$, where $\bar{\alpha}_i$ is an alpha value specified in the brush function (see Figure 9a). Similarly, painting can be applied to other attributes such as transparency or material properties.

Sculpting. We have implemented two variations of sculpting operations that modify the geometry of the surface. The first option is to apply *normal displacements* to the surfel positions, i.e., $\mathbf{x}_i' = \mathbf{x}_i + \bar{d}_i \cdot \mathbf{n}_i$, where $\bar{d}_i$ is a displacement coefficient given in the brush function. As illustrated in Figure 9c, this type of editing operation is particularly suitable for embossing or engraving. On the other hand, the *carving* operation is motivated by the way artists work when sculpting with clay or stone. It implements a "chisel stroke" that removes parts of the surface in the fashion of a CSG-type intersection. The editing tool is defined with respect to a reference plane that is specified by the surface normal of the touching point and an intrusion depth. The new surfel position is then given by

$$\mathbf{x}_i' = \begin{cases} \bar{\mathbf{b}}_i + \bar{d}_i \cdot \bar{\mathbf{n}} & |\mathbf{x}_i - \bar{\mathbf{b}}_i| < \bar{d}_i \\ \mathbf{x}_i & \text{otherwise} \end{cases}, \qquad (13)$$

where $\bar{\mathbf{b}}_i$ is the base point on the reference plane and $\bar{\mathbf{n}}$ the plane normal (see Figure 8). Carving operations can also be applied to rough surfaces (see Figure 9d), where normal displacements fail due to the strong variations of the surface normals.

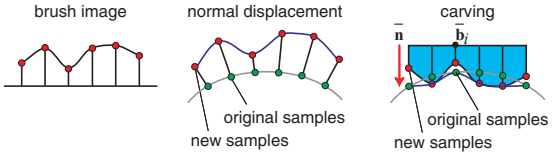


Figure 8: Normal displacement vs. carving.

Filtering. Filtering is a special kind of editing operation that modifies the samples of the original model using a user-specified filter function f. First we apply the filter function to S_Ψ yielding $S_\Psi^f = f(S_\Psi)$, then we combine filtered and original attributes using the brush function for alpha blending. As an example, consider texture filtering, i.e., $\mathbf{c}_i' = \bar{\alpha} \cdot \mathbf{c}_i^f + (1-\bar{\alpha}) \cdot \mathbf{c}_i$, where $\mathbf{c}_i^f$ is the filtered color value (illustrated in Figure 9b). The filter function is usually implemented as a discrete convolution. We can therefore implement arbitrary discrete linear filters by simply choosing the appropriate kernel grid. Filters can be applied to any attribute associated with a surfel, e.g. color, normal or distance from the reference plane for geometric offset filtering. Note that filtering with large kernels can be implemented efficiently in the spectral domain, similar to [Pauly and Gross 2001].

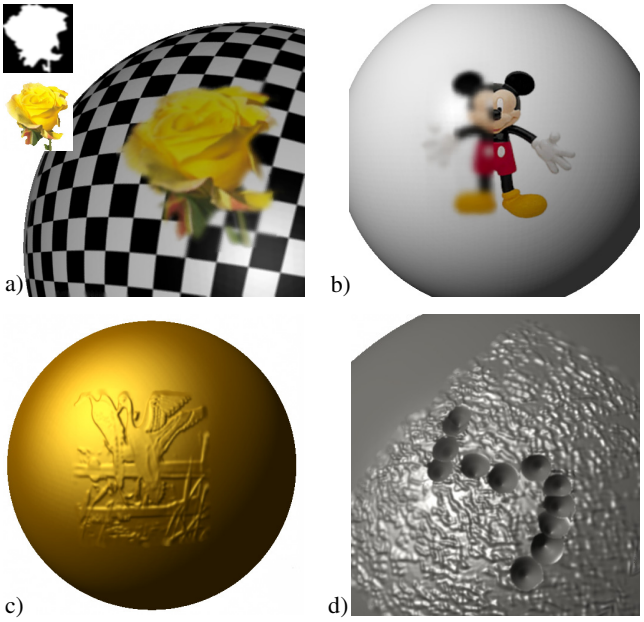


Figure 9: Editing operations: a) Texturing with alpha blending. b) Texture filtering. c) Normal displacement. d) Carving on a rough surface.

6 RESULTS

We have implemented a point-based surface editing system featuring the techniques described in the previous sections. Figure 10 depicts an example of a constrained texture mapping operation with 30 feature points on a model with 218k points. In Table 2 we summarize timings of the multilevel solver for different patches with varying sizes on this object, recorded on a Pentium IV at 2 GHz. The data for the textured patch depicted in Figure 10 is shown in the third row of Table 2.

Table 2: Timings of the multilevel solver: Number of unknowns, time to setup the least squares system, time to compute the initial solution, time to update when one feature point is modified.

UNKNOWNS	SETUP	INIT	UPDATE
58170	6.9 sec.	2.2 sec.	1.8 sec.
107394	14.9 sec.	8.3 sec.	6.6 sec.
215628	26.3 sec.	12.1 sec.	7.6 sec.

After texturing, we additionally embossed a displacement map on the model as shown in Figure 11a. To produce Figure 11b, we started with a sphere with 114k points and then applied the texture from the moon surface shown in Figure 11c. We resampled the sphere at the resolution of the texture, which is 700×700 pixels. Finally, we applied the displacement map depicted in Figure 11d. Our system includes a splat renderer similar to [Zwicker et al. 2001]. On a Pentium IV at 2.0 GHz, it renders approximately 500k antialiased splats per second at an output resolution of 512×512 pixels.

7 CONCLUSIONS AND FUTURE WORK

We presented a versatile system for efficient 3D appearance and shape editing of point-based models. The key ingredients of our editor comprise a flexible and powerful point cloud parameterization and a dynamic resampling scheme based on a continuous

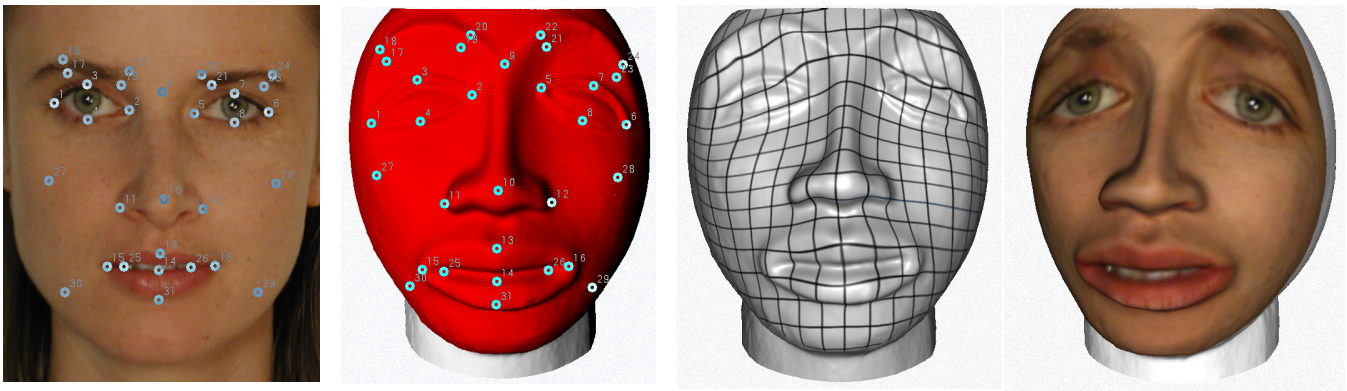


Figure 10: Interactive texture mapping (from left to right): Input texture with feature points, original model with corresponding markers, visualization of the parameter mapping, rendering of the resulting surface.

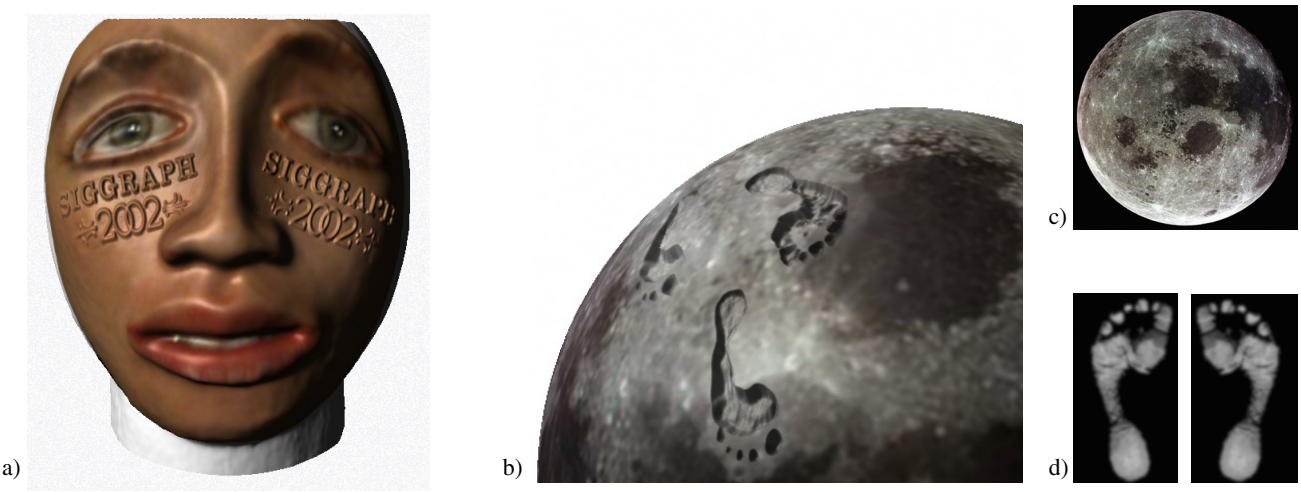


Figure 11: Texturing combined with sculpting: a) Normal displacement on the textured face. b) Footprints on the moon, carved with the displacement maps of d) on a surface textured with the image of c).

reconstruction of the model surface. Although geometry editing is limited to normal displacement, we currently support a broad range of editing operators for efficient 3D content creation. The achievable effects go well beyond the conventional 2D photo editing functionality and new, more sophisticated editing operators can be added very easily. Future work will be devoted to extending our system towards more general modeling operations. We will also investigate high quality rendering of parameterized point-based surfaces using a ray tracing approach.

Acknowledgements

This project has partly been sponsored by the European Graduate Program on Combinatorics, Geometry and Computation. A part of our research on point-based graphics has been supported by MERL, Cambridge MA. We would also like to thank Martin Roth for proofreading the paper.

References

AGRAWALA, M., BEERS, A. C., AND LEVOY, M. 1995. 3d painting on scanned surfaces. pages 145–150. ISBN 0-89791-736-7.

ALEXA, M. 2001. Point set surfaces. In *IEEE Visualization*.

ALIAS WAVEFRONT 2001. Maya. http://www.aliaswavefront.com.

ARIUS3D 2001. pointstream. http://www.pointstream.net.

ASHBY, MANTEUFFEL, AND SAYLOR 1990. A taxonomy for conjugate gradient methods. *J. Numer. Anal.*, 27:1542–1568.

CURLESS, B. AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH 96*, pages 303–312.

EYETRONICS 2001. ShapeSnatcher. http://www.eyetronics.com.

FLOATER, M. S. 1997. Parametrization and smooth approximation of surface triangulations. *Comp. Aided Geom. Design*, 14:231–250.

FLOATER, M. S. AND REIMERS, M. 2001. Meshless parameterization and surface reconstruction. *Comp. Aided Geom. Design*, 18:77–92.

LEVY, B. 2001. Constrained texture mapping for polygonal meshes. In *SIGGRAPH 2001*, pages 417–424. Los Angeles, CA, August 12-17, 2001.

LEVY, B. AND MALLET, J.-L. 1998. Non-distorted texture mapping for sheared triangulated meshes. In *SIGGRAPH 98*, pages 343–352.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *SIGGRAPH 2001*, pages 433–442.

O'NEILL, B. 1966. *Elementary Differential Geometry*. Academic Press.

PAULY, M. AND GROSS, M. 2001. Spectral processing of point-sampled geometry. In *SIGGRAPH 01*, pages 379–386.

PEDERSEN, H. K. 1995. Decorating implicit surfaces. In *SIGGRAPH 1995*, pages 291–300. Los Angeles, CA, August 6-11, 1995.

PERRY, R. N. AND FRISKEN, S. F. 2001. Kizamu: A system for sculpting digital characters. In *SIGGRAPH 2001*, pages 47–56.

PFISTER, H., ZWICKER, M., VANBAAR, J., AND GROSS, M. 2000. Surfels: Surface elements as rendering primitives. In *SIGGRAPH 2000*, pages 335–342. New Orleans, LA, July 23-28, 2000.

RIGHT HEMISPHERE 2001. DeepPaint3D. http://www.us.deeppaint3d.com.

RUSINKIEWICZ, S. AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH 2000*, pages 343–352.

SZELISKI, R. AND TONNESEN, D. 1992. Surface modeling with oriented particle systems. In *SIGGRAPH 92*, pages 185–194. July 1992.

WELCH, W. AND WITKIN, A. 1994. Free-form shape design using triangulated surfaces. In *SIGGRAPH 94*, pages 247–256.

ZWICKER, M., PFISTER, H., VANBAAR, J., AND GROSS, M. 2001. Surface splatting. In *SIGGRAPH 2001*, pages 371–378.