CS Technical Report #378, September 16, 2002

Multiresolution Modeling of Point-Sampled Geometry

Mark Pauly, Leif Kobbelt, Markus Gross*

Computer Science Department
ETH Zurich, Switzerland
e-mail: {pauly, grossm}@inf.ethz.ch
http://graphics.ethz.ch/

*Computer Science Department
RWTH Aachen, Germany
e-mail: kobbelt@cs.rwth-aachen.de
http://www.informatik.rwth-aachen.de/I8/

Multiresolution Modeling of Point-Sampled Geometry

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

Abstract. We present a framework and algorithms for multiresolution modeling of point-sampled geometry. Most of our techniques extend the key ingredients of recent multiresolution editing concepts, including geometry smoothing, simplification, and offset computation, to point clouds. Using these building blocks, we designed a powerful and versatile hierarchical representation for point-based modeling. An intuitive editing metaphor along with dynamic resampling techniques facilitates modeling operations at different scales and enables us to effectively handle very large model deformations. In our approach all processing is performed *directly* on point-sampled geometry without the need for intermediate local triangulation or meshing.

Keywords: Multiresolution modeling, point-sampled geometry, deformation, dynamic resampling.

1 INTRODUCTION

In spite of a long tradition, it is only in recent years that points have received a growing attention in computer graphics. Various researchers proposed the point sample as a simple and versatile graphics primitive complementing conventional geometric elements, such as triangles or splines. Most recent efforts have been devoted to the acquisition, processing [Pauly and Gross 2001], and rendering [Rusinkiewicz and Levoy 2000], [Pfister et al. 2000], [Zwicker et al. 2001] of point-sampled models. Earlier work [Szeliski and Tonnesen 1992], [Witkin and Heckbert 1994] on physics-based frameworks proposed particles to represent and edit surfaces. While particle modeling provides an expressive and intuitive approach to surface design, it is prohibitively expensive for the interactive creation and efficient modeling of large scale point-sampled geometry. In our work, we complement prior research on point-based methods by introducing a framework for multiresolution modeling of point-sampled geometry. To this end, we are transferring key components of the well-established multiresolution decomposition and editing framework from classical manifold-based geometry representations (e.g. splines or polygon meshes) to unstructured point clouds. We adapt basic techniques such as smoothing, normal displacement, or decimation to the point-based setting. Furthermore, we present a powerful and intuitive metaphor for interactive modeling operations and we introduce a resampling method to dynamically adjust the model's sampling rate.

The concept of multiresolution surface editing was successfully applied to triangle meshes in [Zorin et al. 1997]. While this approach is confined to meshes with subdivision connectivity, [Kobbelt et al. 1998] presented a more general framework for arbitrary meshes using variational fairing. More recently, [Guskov et al. 1999] proposed a scheme targeted at mesh signal processing, but also achieved remarkable results in multiresolution surface modifications. In the context of surface editing, multiresolution representations serve two main purposes: Complexity reduction through the use of a model hierarchy and powerful editing semantics at different geometric frequency bands. We found that independent of the underlying geometry representation, each multiresolution editing framework has to provide a set of kernel functions - basic building blocks for higher level editing operations. These core components can be summarized as follows:

- *Refinement and simplification* operators to construct the hierarchy levels of the model
- *Lowpass filtering* to compute smooth base domains
- *Offset computation* to encode geometric details for decomposition and reconstruction
- *Editing metaphors* to interactively manipulate the model surface
- *Dynamic resampling* to adjust the discretization of the model during editing

For polygonal meshes the *up- and downsampling* operations are applied to switch between different resolution levels, where "resolution" refers to both vertex density *and* amount of geometric detail. In the case of point clouds both concepts of resolution have to be treated separately, since a point-sampled surface representing a low geometric detail still has to provide a sufficiently high sampling density. Hence, we use hierarchies of differently dense point clouds *only* to increase the efficiency of filtering operations. Otherwise, the vertex density is kept constant on all resolution levels. For simplification we extend quadratic error metrics [Garland and Heckbert 1997] to the point cloud setting.

Low-pass filtering is employed to remove high geometric frequencies from a given surface for the robust computation of the base domains. Most smoothing techniques for polygonal meshes use recursive Gaussian filtering by iteratively updating vertex positions. Discretization of the Laplacian is usually accomplished by weighted averaging of adjacent vertices [Taubin 1995], [Desbrun et al. 1999]. We extend this concept to point clouds by replacing the vertex neighborhood relation with the k-nearest neighbors of a sample. To accelerate the filtering, we designed a v-cycle multilevel smoother [Kobbelt et al. 1998] that alternates the relaxation with sub- and upsampling operations. Additional local volume constraints [Hubeli and Gross 2001] are used to locally bound the curvature of the base domain.

Offset computation is used to find an explicit representation of the difference between a surface P and its low-pass filtered version Q. We recall that in conventional multiresolution settings the detail $D=P-Q$ is typically stored with respect to local frames spanned by the normals and tangents of Q [Forsey and Bartels 1988]. Since P contains less geometric detail, a smaller number of basis functions and hence, samples, can be used to represent it. Thus, when editing the surface on a lower resolution level Q', the final result P' is reconstructed by first upsampling Q' and then adding the detail D back with respect to the modified local frames of Q'. By contrast, our framework keeps the number of samples intact through the hierarchy levels. Again, the low-frequency version Q is derived by applying the smoothing filter. For offset computation, we modify the sample positions in Q (not the sampling rate!) such that each sample in P can be obtained by normal displacement of the corresponding sample in the newly created point set Q'.

The modification of the point cloud is accomplished by an *interactive editing metaphor* which is similar in spirit to free-form deformation. The user both paints an active region and a handle onto the point sampled surface. Then we generate a continuous tensor field assigning transformations, such as rotations and translations, to each point in space. The samples are subsequently transformed by evaluating the tensor field at corresponding locations in space. We use this metaphor because it does not assume the underlying geometry to be a manifold surface.

It is important to distinguish the simplification and refinement operators of the multiresolution framework from *dynamic* adaptations of the model's sampling rate such as being necessary for excessive local stretches or squeezes during interactive editing. It is useful to compare such *resampling* with a remeshing operation of a heavily deformed polygon mesh [Kobbelt et al. 2000]. In order to guarantee sufficient local sampling density we estimate local stretches dynamically during editing. Once the sampling density becomes too high or too low, we adjust the sampling rate of the representation. Sampling positions are equalized by efficient local relaxations, which guarantees sufficient sampling rates for large model deformations. In addition, we have to preserve and interpolate the detail information D to eventually guarantee proper reconstruction.

The following sections elaborate on the technical details of our framework. Section 3 addresses the core multiresolution representation we use including smoothing, simplification and detail encoding. Section 4 is devoted to the editing metaphor and section 5 discusses the dynamic resampling. In section 6, we present a series of modeling operations to demonstrate the key features of our method.

2 POINT-BASED MODELS

This section briefly summarizes some prerequisites, which are fundamental to our framework. Our input data consists of an unstructured point cloud $P = \{p_i | 1 \le i \le N\}$ that describes some underlying manifold surface S. Each sample $p \in P$ stores a geometric position $\mathbf{p} = (x, y, z)$ as well as a set of scalar attributes, such as color c or texture coordinates (u, v).

2.1 Neighborhood

Contrary to polygonal representations we base all required local computations on *spatial* proximity between samples instead of *geodesic* proximity between mesh vertices. This approach is justified by the fact that both distance measures are similar, if the underlying smooth surface is sampled sufficiently dense [Amenta et al. 1998]. In [Floater and Reimers 2001] Floater investigated different approaches for defining local neighborhoods for point clouds. We found that for our purposes the set of k-nearest neighbors of a sample point $\mathbf{p}$, denoted by the index set N, is most suitable. We define the radius r of the enclosing sphere of the k-nearest neighbors of $\mathbf{p}$ as $r = \max_{i \in N} \|\mathbf{p} - \mathbf{p}_i\|$.

2.2 Normal Estimation

To evaluate the surface normal we need some method for estimating the tangent plane T at a point $\mathbf{p}$. For this purpose, we use a weighted least-squares optimization. The goal is to find the plane

$$T : \mathbf{p}_i \cdot \mathbf{n} - d = 0 \tag{1}$$

that minimizes the sum of weighted squared distances

$$\sum_{i \in N} (\mathbf{p}_i \cdot \mathbf{n} - d)^2 \phi_i \tag{2}$$

subject to $\|\mathbf{n}\| = 1$. The weights ϕ_i are based on Euclidean distance, i.e. $\phi_i = \phi(\|\mathbf{p}_i - \mathbf{p}\|)$ with ϕ being a smooth, positive and monotonously decreasing function. We use the Gaussian

$$\phi(x) = e^{-x^2/(\sigma^2 r^2)}, \tag{3}$$

where σ is an additional parameter controlling the locality of the approximation. Note that the weight function depends on the radius r of the neighborhood sphere and thus dynamically adapts to the local sampling density.

Using the method of Lagrange multipliers [Sagan 1969] we solve the constrained least-squares problem (2) by reducing it to the eigenvector problem

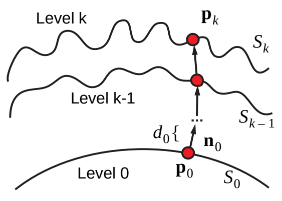
$$\mathbf{C} \cdot \mathbf{v} = \lambda \cdot \mathbf{v}, \qquad \mathbf{C} = \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \dots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \dots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}, i_j \in N, \tag{4}$$

where $\mathbf{C}$ is the 3×3 covariance matrix and $\bar{\mathbf{p}}$ the weighted average of the $\mathbf{p}_i$. The normal vector $\mathbf{n}$ is then equal to the eigenvector $\mathbf{v}_i$ with smallest associated eigenvalue λ_i.

3 MULTIRESOLUTION REPRESENTATION

In this section we introduce multiresolution point clouds as a new geometry representation for two-manifold surfaces. Each sample p is represented by a point $\mathbf{p}_0$ plus a sequence of *normal displacement offsets* $d_0, \dots, d_{n-1}$. To reconstruct the position of p at a certain level k we recursively displace $\mathbf{p}_0$ in normal direction, i.e.

$$\mathbf{p}_k = \mathbf{p}_0 + d_0\mathbf{n}_0 + d_1\mathbf{n}_1 + \dots + d_{k-1}\mathbf{n}_{k-1}, \tag{5}$$



where $\mathbf{n}_j$ denotes the surface normal estimate at $\mathbf{p}_j$ on level j, which is based on $\mathbf{p}_j$ and its neighboring samples from the same resolution level. Hence, every sample point p is active on all levels of resolution and the polygon $\mathbf{p}_0, \dots, \mathbf{p}_n$ defines its normal trajectory, since each polygon edge $\overline{\mathbf{p}_j\mathbf{p}_{j+1}}$ is aligned to the normal vector $\mathbf{n}_j$.

At each level k the point set defines the model surface S_k at the corresponding scale, with level 0 being the smoothest approximation. Note again that resolution refers to the amount of geometric detail, *not* to the sampling rate. This also explains why we do not subsample the lower resolution levels. To unambiguously reconstruct the normal trajectory each intermediate point $\mathbf{p}_j$ is needed. However, starting from $\mathbf{p}_0$ all $\mathbf{p}_j$ can be represented by one scalar coordinate d_j. This amounts to only moderate storage overhead for typical decomposition levels of up to 5 in practice.

To eventually create this multilevel hierarchy, we require the following kernel functions: A smoothing operator (low-pass filter) is needed to compute the different levels of geometric detail (Section 3.1). A fast point cloud simplification method (Section 3.2) reduces the number of samples for efficient v-cycle computation (Section 3.3). A decomposition algorithm encodes the detailed surface with respect to its smooth approximant. By applying low-pass filtering and decomposition recursively, we extent this two-band decomposition to a multilevel scheme.

3.1 Smoothing Operators

One way to describe surface smoothing is by the diffusion equation [Desbrun et al. 1999]

$$\frac{\partial S}{\partial t} = \lambda \Delta S\,, \tag{6}$$

where Δ denotes the Laplace-Beltrami operator on the surface. Using explicit Euler integration, this leads to the common iterative formula for Gaussian smoothing

$$\mathbf{p}' = \mathbf{p} + \lambda \Delta \mathbf{p}\,, \tag{7}$$

where $\Delta\mathbf{p}$ is some discrete approximation of the Laplacian at a point $\mathbf{p}$. A standard method for computing such a discretization is the use of umbrella operators, which can be transferred to point sets by

$$\Delta\mathbf{p} = \frac{1}{\Omega}\sum_{i \in N} \omega_i(\mathbf{p}_i - \mathbf{p}), \qquad \Omega = \sum_{i \in N} \omega_i\,, \tag{8}$$

where $\{\omega_i\}$ defines a set of weights. The uniform umbrella is obtained for $\omega_i = 1, \forall i$ while a scale-dependent umbrella operator results from $\omega_i = 1/\|\mathbf{p}_i - \mathbf{p}\|$. The degree of smoothness can be controlled by the scale parameter λ, the number of iterations and the size of the local neighborhoods of the sample points. These neighborhoods are computed at the beginning of the smoothing operation and cached during the iteration. This improves the efficiency of the computations and also increases the stability of the smoothing filter, since it prevents clustering effects due to the tangential drift of sample points.

Volumetric constraints. A common problem of iterative smoothing operators is shrinkage. Taubin addresses this problem by alternating between smoothing steps with positive and negative scale parameters [Taubin 1995]. Desbrun et al. [Desbrun et al. 1999] measure the total object volume before and after the smoothing operation and rescale the model uniformly. Shrinkage is a non-uniform problem, however, and should thus be handled by a locally adaptive technique. Therefore we approximate the local change in volume in the neighborhood of a point $\mathbf{p}$ caused by a smoothing step as the cylindrical volume

$$\Delta V = \pi r^2 \cdot \|\mathbf{p}' - \mathbf{p}\| \tag{9}$$

and compensate the shrinkage by displacing the neighbors of $\mathbf{p}$ by $(\mathbf{p} - \mathbf{p}')/k$ (see Figure 1). This approach is similar to the local volume preservation strategy introduced by Hubeli et al. [Hubeli and Gross 2001] for triangle meshes. They observed, however, that applying the local anti-shrinkage technique decreases the impact of the smoothing operator.

3.2 Point Set Simplification

To apply efficient multilevel smoothing algorithms to point-sampled geometry, we need operators for sub- and supersampling. A subsampling scheme coarsifies the representation and we derive the corresponding operator by adapting a mesh decimation scheme. Supersampling will then be achieved by inverting this operator.

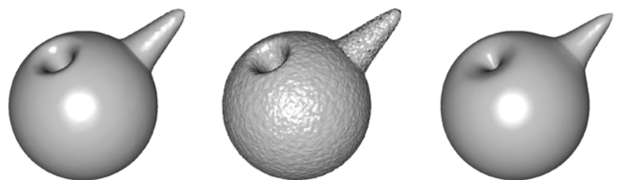


Figure 1: Local vs. global volume preservation: The middle image shows the original noisy surface. On the left 100 smoothing iterations with the local strategy, on the right 100 iterations with global rescaling.

Mesh simplification algorithms are usually based on an atomic decimation operator that decreases the number of vertices in the mesh by one. The most common operator is the edge collapse or vertex contraction where two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ are replaced by a new one $\bar{\mathbf{v}}$. This new vertex inherits the neighborhood relation from both its ancestors. In [Garland and Heckbert 1997] the decimation of triangle meshes is controlled by a quadric error metric. The idea is to approximate the surface locally by a set of tangent planes and to estimate the geometric deviation of a vertex $\bar{\mathbf{v}}$ from the surface by the sum of the squared distances to these planes. The error quadrics for each mesh vertex $\mathbf{v}$ are initialized with a set of planes defined by the triangles around that vertex and can be represented by a symmetric 4×4 matrix $Q_{\mathbf{v}}$. The quality of the collapse $(\mathbf{v}_1, \mathbf{v}_2) \to \bar{\mathbf{v}}$ is then rated according to the minimum of the error functional $Q_{\bar{\mathbf{v}}} = Q_{\mathbf{v}_1} + Q_{\mathbf{v}_2}$. In order to adapt this technique to the decimation of unstructured point clouds we have to use the k-nearest neighbor relation, since manifold surface connectivity is not available. To initialize the error quadrics for every point sample $\mathbf{p}$, we estimate a tangent plane E_i for every neighbor $\mathbf{p}_i$ in the k-neighborhood of $\mathbf{p}$. This tangent plane is spanned by the vector $\mathbf{e}_i = \mathbf{p} - \mathbf{p}_i$ and $\mathbf{b}_i = \mathbf{e}_i \times \mathbf{n}$, where $\mathbf{n}$ is the estimated normal vector at $\mathbf{p}$.

After this initialization the point cloud decimation works exactly like mesh decimation with the point $\bar{\mathbf{p}}$ inheriting the neighborhoods of its ancestors $\mathbf{p}_1$ and $\mathbf{p}_2$ and being assigned the error functional $Q_{\bar{\mathbf{v}}} = Q_{\mathbf{v}_1} + Q_{\mathbf{v}_2}$. Figure 2 shows the effectiveness of the algorithm.

3.3 Multilevel Smoothing

Since iterative Gaussian smoothing as defined by Equation (7) exhibits slow convergence on lower frequencies [Kobbelt et al. 1998], we combine the smoothing operator and simplification method described above into an efficient multilevel smoothing algorithm. Using a complete v-cycle of alternating smoothing and down- resp. upsampling steps, we obtain a smooth approximation S' of a surface S as

$$S' = \Phi \cdot \Psi^{-1} \cdot \Phi \cdot \ldots \cdot \Phi \cdot \Psi \cdot \Phi(S)\,, \tag{10}$$

where Φ is the smoothing operator and Ψ the downsampling (simplification) operator. The upsampling (refinement) operator Ψ^{-1} implements the inverse of the simplification method of Section 3.2. To re-insert a point back into the model, we store the local neighborhood of the point prior to the point pair contraction. Then we can compute the position of the inserted point as a weighted average of the points of its neighborhood.

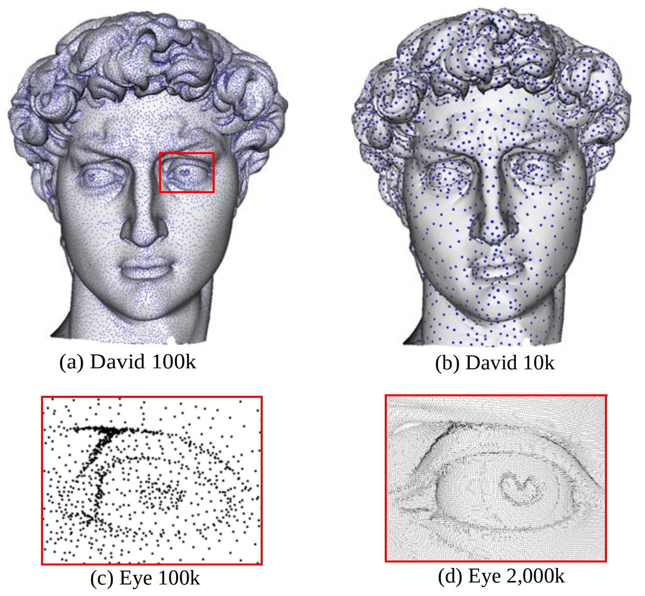


Figure 2: Simplification with quadric error metrics. The original model has been downsampled from 2,000,606 points to 100,000 in (a) in 183 seconds, and to 10,000 in (b) in 197 seconds. We show the sampling distribution of the subsampled models as blue dots on the original geometry. In (c) the region around the eye of (a) is magnified, while (d) shows the same part of the original model.

3.4 Decomposition

For a given geometric model, the decomposition operator separates global (low frequency) shape information from local (high frequency) detail. The input to our multiresolution point cloud decomposition operator usually consists of a given set of samples P and a low frequency approximation Q. The major difficulty arises from the fact that the low-pass filter operator that generates Q does not guarantee a specific sampling pattern: We cannot easily find pairs of samples $\mathbf{p} \in P$ and $\mathbf{q} \in Q$ such that the displacement $\overline{\mathbf{pq}}$ is in normal direction. Hence, the decomposition operator has to resample the surface defined by the sample set Q to a new point cloud R such that we obtain a normal base point $\mathbf{r} \in R$ for every original sample $\mathbf{p}$ without changing the actual geometric shape represented by Q. We perform this resampling by a variant of the moving least-squares surface [Alexa et al. 2001], [Levin]. The basic idea is to find weighted least-squares planes that approximate a local neighborhood of point samples. Then a local polynomial approximation is computed with respect to a parameterization over this plane. For our purposes it turns out to be sufficient to use the initial weighted least-squares plane for normal estimation. Hence we omit the second polynomial approximation step. By this we are, in fact, using moving least-squares surfaces of polynomial degree one. Finding the best fitting plane requires a non-linear optimization step since we have to determine a base point $\mathbf{r}$ and a normal direction $\mathbf{n}$. In earlier approaches the non-linear optimization has been implemented by a Powell's method. We, however, prefer a Newton-type iteration which is based on a geometric gradient estimation. Suppose we have the sample point $\mathbf{p}$ and the point cloud Q and we are looking for a base point $\mathbf{r} \notin Q$ lying on the moving least-squares surface S defined by Q such that the displacement $\overline{\mathbf{rp}}$ is perpendicular to S. We start with an initial estimate $\mathbf{r}_0$ for $\mathbf{r}$ by choosing that point from the cloud

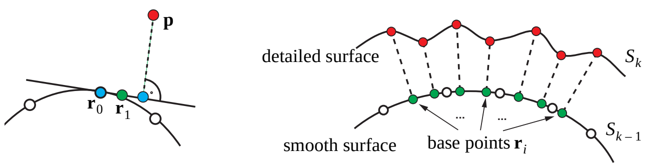


Figure 3: Multiresolution decomposition. The left image illustrates one step of the Newton iteration. On the right, the (green) base points for the (red) points on the detailed surface are shown.

Q, which lies closest to $\mathbf{p}$. Finding the best fitting plane E_0 to a local neighborhood of samples $Q \backslash \{\mathbf{r}_0\}$ that passes through $\mathbf{r}_0$ is an easy task since it only requires the solution of a linear system (see Section 2.2). By orthogonal projection of $\mathbf{p}$ onto E_0 we obtain another estimate $\mathbf{r}_1$ for $\mathbf{r}$. We iterate this procedure. In each step we find another estimate $\mathbf{r}_{i+1}$ by projecting $\mathbf{p}$ onto the plane E_i which is the weighted least-squares plane fitted to a local neighborhood of samples of Q around the previous estimate $\mathbf{r}_i$. This iterative procedure corresponds to a Newton-iteration scheme to find the point $\mathbf{r}$ on the surface S for which the displacement $\overline{\mathbf{rp}}$ is in normal direction. This becomes obvious from the observation that we replace the surface S by its linear approximation E_i in every step and then solve the base point problem for this approximation. To avoid oscillatory behavior of the iterative scheme, we introduce an additional damping factor λ, i.e. instead of fully updating from $\mathbf{r}_i$ to $\mathbf{r}_{i+1}$ we use the damped update from $\mathbf{r}_i$ to $(1-\lambda)\mathbf{r}_i + \lambda\mathbf{r}_{i+1}$. Choosing $\lambda = 0.5$ works well in most cases (see Figures 3 and 4).

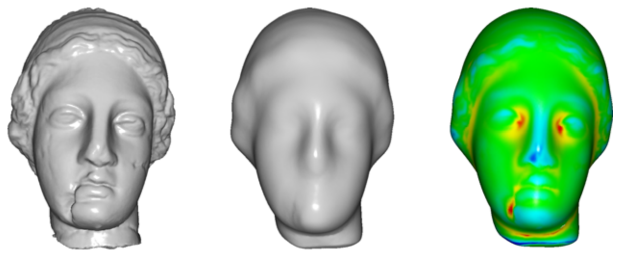


Figure 4: Decomposition: Original (left), smooth base domain (middle) and color-mapped detail coefficients, ranging from maximum negative (blue) to maximum positive (red) displacement.

In this context, two issues deserve further discussion: Since we are using a Newton-type optimization with damping, the iteration is guaranteed to find the correct base point $\mathbf{r}$, if the initial estimate $\mathbf{r}_0$ lies sufficiently close. In our experiments we never experienced any difficulties with the convergence of the process, since the point cloud Q represents a smooth low-frequency surface. A further issue relates to the low-pass filtering we use for the computation of the base domain. It is important to note that the moving least-squares surface intrinsically includes a low-pass filtering of the such-defined surface. In principle we could apply the decomposition operation directly to the point cloud P such that we find base points $\mathbf{r}$ lying on the moving least-squares surface defined by P. However, there are two drawbacks. The first is that using a second independently generated point cloud Q provides more flexible control (including local volume preservation) on the amount of geometric detail that is removed when going from P to Q. Moreover, the v-cycle smoothing algorithm (see Section 3.3) turns out to be much more efficient for aggressive smoothing. Another aspect is that the moving least-squares surface tends to instabilities and oscillations, if the point cloud P is noisy. Hence a pre-smoothing step would be necessary anyway.

4 EDITING

Given a point-based multiresolution representation of a surface S we now need some modeling technique to manipulate the point cloud P_k that defines the surface S_k of a certain level k. For this purpose we have implemented a point-based free-form deformation algorithm. Deformations can be controlled by a continuous tensor-field, which is created using a simple and flexible user interaction metaphor.

The tensor-field defines a transformation (translation and rotation) for each point in space based on a continuously varying scale parameter $t \in [0, 1]$. This scale parameter is computed using spatial distance functions that are defined by specifying two regions in space, a *zero-region* $\chi_0 \in \mathbf{R}^3$ and a *one-region* $\chi_1 \in \mathbf{R}^3$ subject to $\chi_0 \cap \chi_1 = \varnothing$. Points within χ_0 will not be displaced at all ($t = 0$), while points within χ_1 will experience the maximum displacement ($t = 1$). All other points are displaced in such a way that we create a smooth transition between zero- and maximum displacement.

This is achieved as follows: For each point $\mathbf{p}$, we compute the distance $d_0(\mathbf{p})$ to the zero-region and the distance $d_1(\mathbf{p})$ to the one-region defined as

$$d_j(\mathbf{p}) = \begin{cases} 0 & \mathbf{p} \in \chi_j \\ \min_{\mathbf{q} \in \chi_j}(\|\mathbf{p} - \mathbf{q}\|) & \mathbf{p} \notin \chi_j \end{cases} \tag{11}$$

for $j = 0, 1$. The new position for $\mathbf{p}$ is then determined as $\mathbf{p}' = F(\mathbf{p}, t)$. F is a deformation function and

$$t = \beta(d_0(\mathbf{p})/(d_0(\mathbf{p}) + d_1(\mathbf{p}))) \tag{12}$$

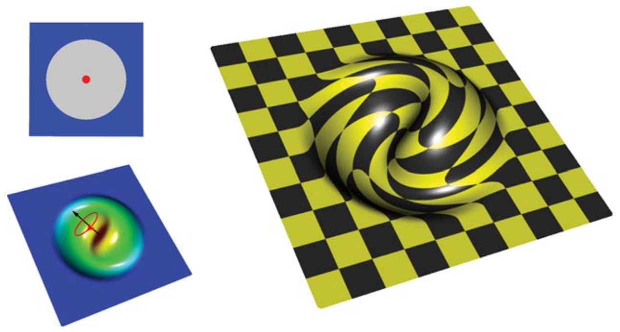
the scale parameter for $\mathbf{p}$ with blending function β. In our implementation we use the C^1 continuous blending function $\beta(x) = (x^2 - 1)^2$, satisfying $\beta(0) = 1$ and $\beta(1) = 0$ and $\beta'(0) = \beta'(1) = 0$. This ensures a smooth transition of displacements at the boundaries of χ_0 and χ_1.

Currently we use two deformation functions:

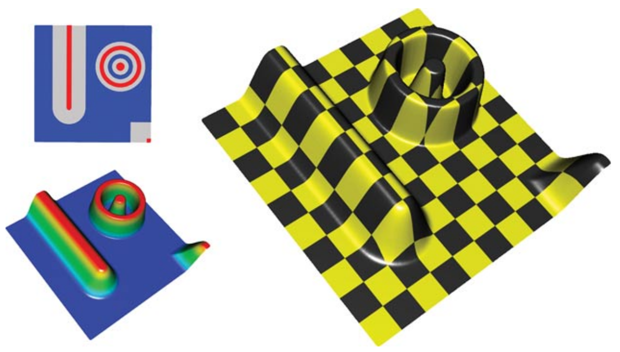
- Translation: $F_T(\mathbf{p}, t) = \mathbf{p} + t \cdot \mathbf{v}$, with $\mathbf{v}$ being the translation vector. Many choices for $\mathbf{v}$ are conceivable, e.g. a constant vector, the surface normal at $\mathbf{p}$ or a vector pointing to a user-specified point $\hat{\mathbf{p}}$
- Rotation: $F_R(\mathbf{p}, t) = R(\mathbf{r}, t \cdot \alpha) \cdot \mathbf{p}$, where $R(\mathbf{r}, \alpha)$ is the matrix that specifies a rotation around axis $\mathbf{r}$ with angle α.

The user specifies χ_0 and χ_1 by marking points on the point cloud P. This is achieved with a simple paint tool that supports drawing of points, lines, circles and other primitives, as well as flood filling and erasure. This provides very flexible and powerful editing functionality as illustrated in Figure 5. Yet the user interface and handling of the deformation is simple and intuitive. In principle, there is no need to restrict χ_0 and χ_1 to be part of the model surface. Instead the zero- and one-regions can also be specified on a different model that can be arbitrarily aligned with P.

Our deformation method shows some similarities to the *wires* scheme introduced by Singh and Fiume [Singh and Fiume 1998]. We use the same blending function and our zero- and one-regions are similar to the wires and domain curves. However, we believe that our technique provides more flexibility, since χ_0 and χ_1 are defined as volumetric entities, not one-dimensional curves.



(a) 90 degree rotation around an axis through the plane center that spans an angle of 45 degrees with the plane normal



(b) Translation in the direction of the plane normal

Figure 5: Two examples of elementary free-form deformations. The upper left image shows the zero-region (blue) and the one-region (red). The scale parameter is visualized below using a rainbow color map. On the right the final textured surface.

5 DYNAMIC RESAMPLING

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. detail coefficients or color values (Section 5.2).

5.1 Measuring surface stretch

During the deformation we have to measure the local stretching to decide when new sample points need to be inserted to preserve the overall sampling density. We use the first fundamental form known from differential geometry to measure the local distortion at each sample point [DoCarmo 1976]. Let $\mathbf{u}$ and $\mathbf{v}$ be two tangent vectors at a sample point $\mathbf{p}$. We initialize $\mathbf{u}$ and $\mathbf{v}$ such that they are of unit length and orthogonal to each other. The first fundamental form at $\mathbf{p}$ is now defined by the 2×2 matrix

$$\begin{bmatrix} \mathbf{u}^2 & \mathbf{u}\cdot\mathbf{v} \\ \mathbf{u}\cdot\mathbf{v} & \mathbf{v}^2 \end{bmatrix} \qquad (13)$$

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. Since we initialized the two tangent vectors $\mathbf{u}$ and $\mathbf{v}$ to an orthonormal system, the resulting first fundamental form is the identity matrix indicating that initially there is no tension on the surface.

When we apply a deformation to the point samples, the point $\mathbf{p}$ is shifted to a new position $\mathbf{p}'$ and the 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 computing the eigenvalues of 13. We can rate the distortion by taking the ratio of the two eigenvalues (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 13 defines an ellipse in the tangent plane centered at $\mathbf{p}$ with the principal axes defined by the eigenvectors and eigenvalue, 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 6).

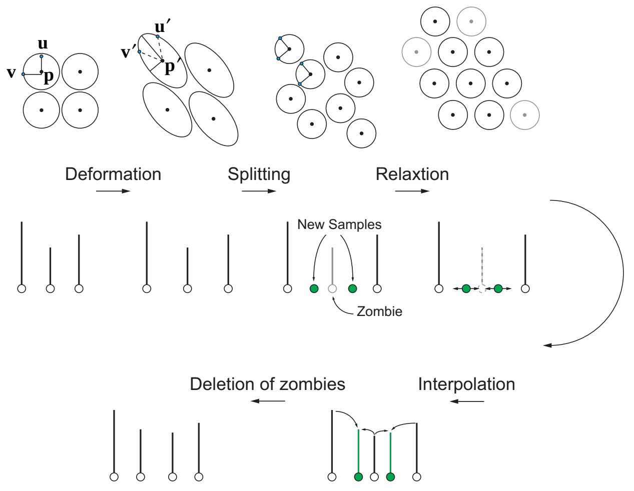


Figure 6: Dynamic resampling: The top row shows the sampling positions, while the bottom row illustrates the scalar function values (indicated by lines). After deforming the model, splitting creates new samples and zombies. The latter are ignored when computing sample positions during relaxation, but are used when computing function values during interpolation.

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 filter operator to the sample position we call it a *relaxation filter*, while we call it an *interpolation filter*, if we apply it to the function values. This definition makes sense if we look at the way how we use these filters in the context of our resampling framework:

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. 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.

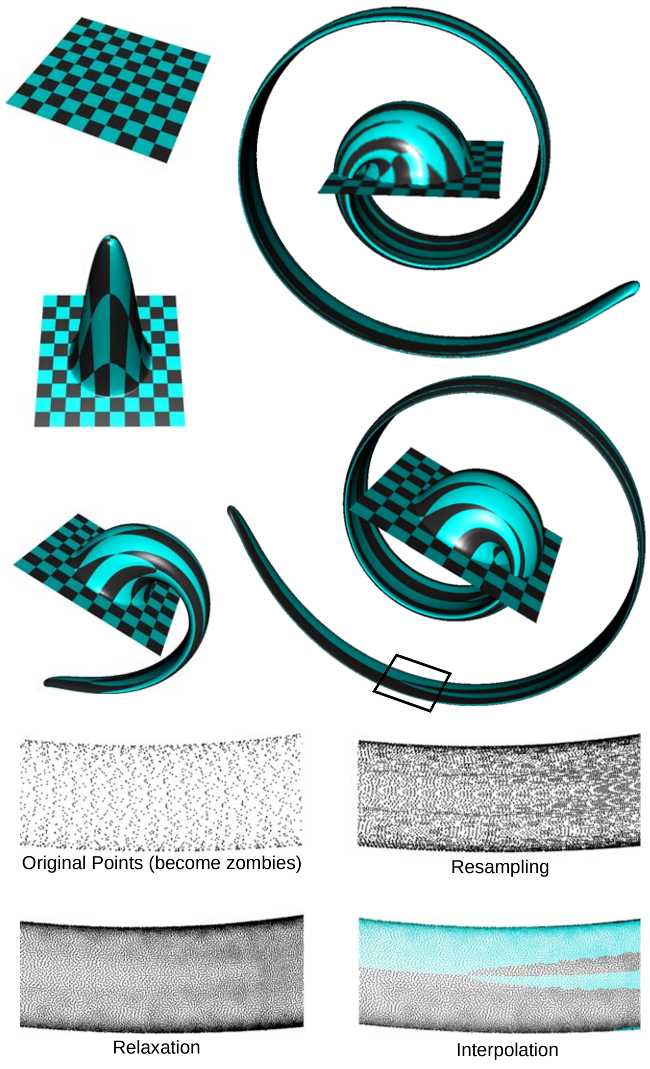


Figure 7: A very large deformation of a plane. The left column illustrates intermediate steps of the deformation. The right column shows the final deformations in two resolutions: The top image contains 69,346 points (original plane 10,000), while the bottom image contains 621,384 points (original plane 160,000). The zooms show the four stages of the dynamic resampling (cf. Figure 6). Note that we interpolate texture coordinates, not color, in which case the image would be much more blurry.

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 $\mathbf{p}$. 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 7 illustrates our dynamic resampling method for a very large deformation that leads to a substantial increase in the number of sample points. It clearly demonstrates the robustness and scalability of our method, even 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 detail coefficients or color will be lost, which leads to blurring when the surface is re-stretched. Also, from an implementation point of view, data structures can be held much more simple if only dynamic insertions, but no deletions are allowed. Thus instead of dynamic sample deletion we do an optional "garbage collection" at the end of the editing operation, using the simplification method of Section 3.2 to reduce the sampling rate.

6 RESULTS & DISCUSSION

We have implemented the algorithms described above and tested our method on a number of point-sampled models. Figure 8 shows an example of a multiresolution edit of a human body using four elementary deformations, 3 rotations (head, arm, leg) and one translation (belly). Note how the geometric detail is preserved in an intuitive manner, e.g. the crinkles on the shirt. The same effect is demonstrated more noticeably in Figure 9, where a laser-range scan of the ball joint of a human leg is deformed using a translation and a rotation. This model contains a lot of high-frequency detail that is accurately reconstructed on the deformed surface.

In Figure 10 we created an embossing effect by texture-mapping the zero- and one-regions onto a plane and applying a translatory deformation in a fixed direction. The zooms of the further-deformed plate illustrate the difference between editing on single resolution (left) versus multiresolution (right). Obviously, if the displacements are not in normal direction, the result is counter-intuitive. This example also shows how complex geometry can be created easily using our deformation framework in connection with conventional texture mapping.

Another advantage of our editing metaphor is scalability. Figure 11 shows an editing operation on a scan of Michelangelo's David. This model contains more than 2 million points and is clearly beyond the capabilities of our system for interactive modeling. However, the user can interactively edit a subsampled version of the surface until she is satisfied with the result. Then all the parameters of the deformation (e.g. zero- and one-region) can easily be transferred to the high-resolution model and the deformation can be computed offline. This is possible because our system does not rely on intrinsic features of the surface, such as control points.

Implementation Details. We use a kd-tree for computing the set of k-nearest neighbors. We found that for all our models $8 \leq k \leq 12$ is a suitable choice for stable computations. The kd-tree features fast creation time and fast nearest neighbor queries. In our current implementation, a tree of 500,000 points is build in 1.33 seconds and a query for 10 nearest neighbors in such a tree takes approx. 9 microseconds (on a Pentium 4, 1.8GHz). However, the kd-tree is a static structure and thus not suitable for dynamically changing point positions. Therefore, we cache neighborhood information during deformations and rebuild the complete tree at the end of the editing session (where we also do carbage collection).

Performance. For the examples of Figures 8, 9 and 10 we achieve for all operations interactive framerates of between 0.3 and 4 secs. Since our code is not yet optimized for performance, we believe that a substantial increase in speed is still possible. In particular, we believe that the efficiency of the local computations can be improved considerably by exploiting coherence.

Stability. An important issue concerning the robustness of our multiresolution decomposition is the stability of the normal estimation. In this context it is important that the smoothing operator guarantees bounded curvature of the resulting base domain. This is the major motivation for using a local volume preservation method, as it prevents spikes in the smoothed surface that would cause the normal estimation of the decomposition operator to fail.

7 CONCLUSIONS & FUTURE WORK

In this paper we introduce a new multiresolution modeling framework for point-sampled geometry. We have implemented multilevel smoothing, fast simplification and a multiresolution decomposition operator for unstructured point clouds. In connection with a flexible and powerful editing metaphor and a dynamic resampling strategy, our system handles large deformations with intuitive detail reconstruction. Future work will be directed towards adding more semantics to the editing operations including multiresolution feature detection and extraction. We also believe that our framework is well suited for morphing and animation.

References

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

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH 1998*.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH 1999*.

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

FLOATER, M. AND REIMERS, M. 2001. Meshless parameterization and surface reconstruction. In *Comp. Aided Geom. Design 18*.

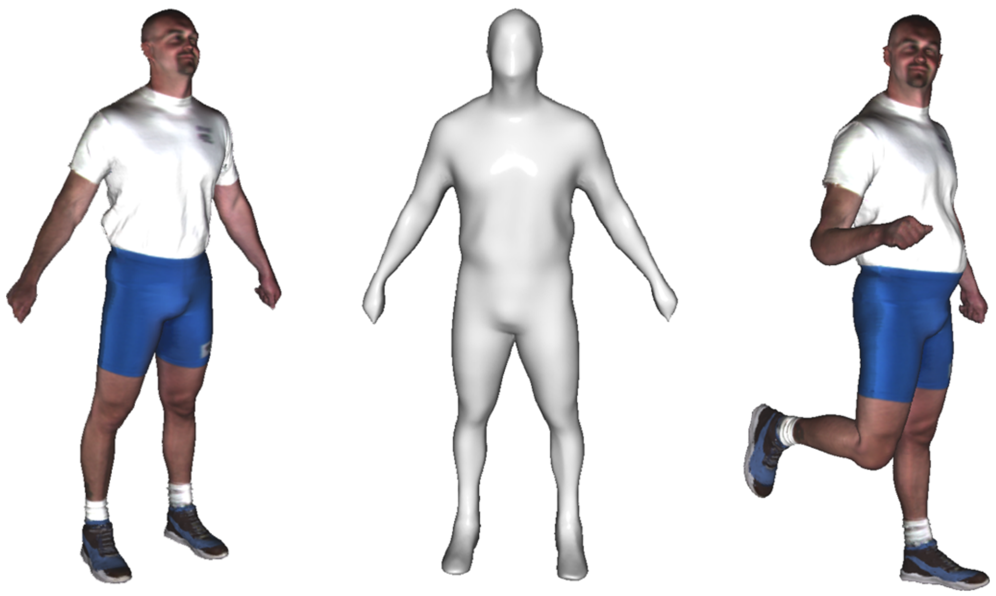


Figure 8: Multiresolution edits on a human body (148,138 points). The original model is shown on the left, the smooth base domain in the middle and the deformed surface on the right.

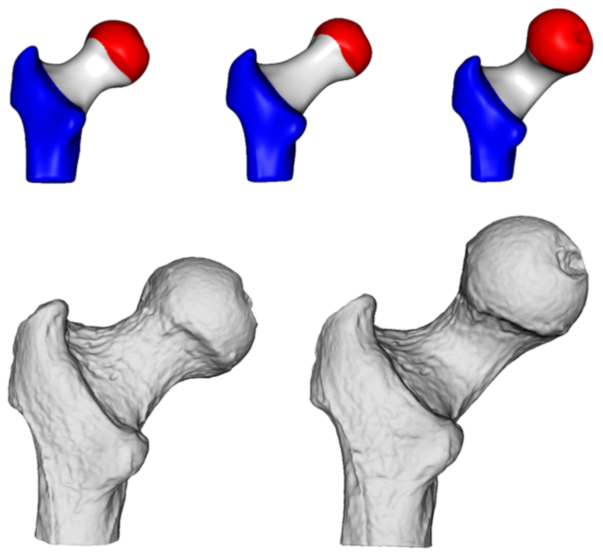


Figure 9: Edits on a ball joint (137062 points). The top row shows the smooth base domain with zero-region (blue) and one-region (red).

FORSEY, D. AND BARTELS, R. 1988. Hierarchical b-spline refinement. In *Proceedings of SIGGRAPH 1988*.

GARLAND, M. AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997*.

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

HUBELI, A. AND GROSS, M. H. 2001. Multiresolution methods for non-manifold models. In *IEEE Transaction on Visualization and Computer Graphics 2001*.

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).

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

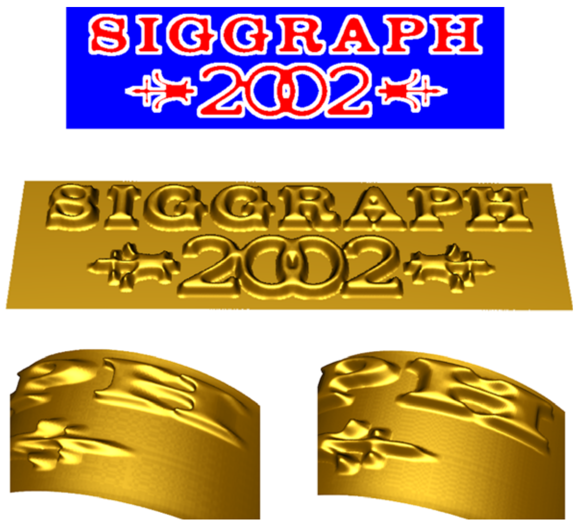


Figure 10: The Siggraph logo embossed on a plane (123464 points). The top image shows zero- and one-region (blue resp. red), the lower images show zooms of the deformed surface. On the left single-resolution deformation, on the right multiresolution deformation with proper detail reconstruction.

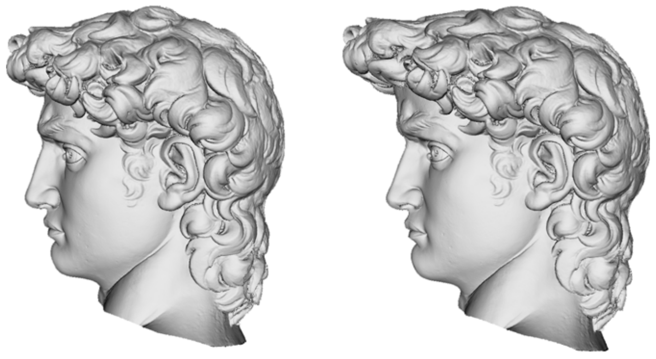


Figure 11: A gentle deformation of Michelangelo’s David (2,000,606 points).

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

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*.

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

SAGAN, H. 1969. *Introduction to the Calculus of Variations*. Dover Publications.

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*.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 1995*.

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

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

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

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