International Journal of Shape Modeling
© World Scientific Publishing Company

Laplacian Framework for Interactive Mesh Editing

Yaron Lipman Olga Sorkine
School of Computer Science, Tel-Aviv University
Ramat Aviv, Tel-Aviv, 69978, Israel.
{*lipmanya*|*sorkine*}*@post.tau.ac.il*

Marc Alexa
Discrete Geometric Modeling Group
Darmstadt University of Technology
alexa@informatik.tu-darmstadt.de

Daniel Cohen-Or David Levin
Tel-Aviv University
{*dcor*|*levin*}*@tau.ac.il*

Christian Rössl Hans-Peter Seidel
MPI Saarbrücken
{*roessl*|*hpseidel*}*@mpi-sb.mpg.de*

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Communicated by (xxxxxxxxxx)

Recent works in geometric modeling show the advantage of local differential coordinates in various surface processing applications. In this paper we advocate surface representation via differential coordinates as a basis to interactive mesh editing. One of the main challenges in editing a mesh is to retain the visual appearance of the surface after applying various modifications. The differential coordinates capture the local geometric details and therefore are a natural surface representation for editing applications. The coordinates are obtained by applying a linear operator to the mesh geometry. Given suitable deformation constraints, the mesh geometry is reconstructed from the differential representation by solving a sparse linear system. The differential coordinates are not rotation-invariant and thus their rotation must be explicitly handled in order to retain the correct orientation of the surface details. We suggest two methods for computing the local rotations: the first estimates them heuristically using a deformation which only preserves the underlying smooth surface, and the second estimates the rotations implicitly through a variational representation of the problem.

We show that the linear reconstruction system can be solved fast enough to guarantee interactive response time thanks to a precomputed factorization of the coefficient matrix. We demonstrate that our approach enables to edit complex meshes while retaining the shape of the details in their natural orientation.

Keywords: mesh editing; differential coordinates; Laplacian coordinates.

1. Introduction

Editing tools for three dimensional shapes have been an important research area in geometric modeling and computer graphics. It is a challenging problem since a good editing tool should be intuitive and easy to use, and at the same time flexible and powerful. In the following we are focusing on mesh editing, where the tool works on shapes represented by triangular meshes. There is a vast amount of tools for free-form modeling of shapes from scratch mostly based on piecewise polynomial surface representations (see e.g., [7,11]). For triangle meshes, the most popular example of such tools are subdivision techniques [20]. However, these techniques aim at the design of smooth surfaces, and they are not appropriate for editing arbitrary, existing meshes such as the complex, highly detailed shapes that emerge from digitizing real-world models.

There are a number of crucial requirements on an editing operation which make shape modeling a challenging problem: The operation should be efficient enough for *interactive* work. It should provide *local* influence and *detail* preservation. Typically, moving a handle is a local operation, where only nearby vertices are affected. In addition, a flexible tool allows the user to easily define the degree of locality and hence enables edits of different scale. When dragging the handle vertices, the deformed surface should retain the look of the original surface in a natural way. If a surface is smooth, the modified shape should remain smooth. If the surface contains some geometric details, the *shape* and *orientation* of these details should be preserved. The editing operation should naturally change the shape and simultaneously respect the structural detail. This problem becomes more pronounced with the emergence and the proliferation of three dimensional scanned models. Unlike CAD models, the surfaces of scanned models are usually not smoothed and contain high-frequency details which one would like to preserve since they contribute a lot to the appearance of the surface.

In this paper we advocate the use of differential coordinates as an alternative representation for the vertex coordinates. We show that this representation leads to efficient, interactive and intuitive shape modeling including local control and detail preservation. The differential coordinates represent the geometric details and are defined with respect to a common global coordinate system. This representation allows a direct detail-preserving reconstruction of the modified mesh by solving a linear least squares system. The differential coordinates are not rotation-invariant since they are defined in a global coordinate frame. As we show below, this can cause distortion of the orientation of the details on the reconstructed surface. We suggest two methods to rectify the local orientation. First, we rotate the differential coordinates according to the rotation of an approximated local frame. The second method finds the local transformations implicitly, by expressing them as linear functions of the unknown (deformed) mesh geometry, and then incorporating these transformations into the reconstruction optimization.

The method we present in this paper allows editing arbitrary triangle meshes. Our approach enables flexible, intuitive and interactive shape modeling. The method is conceptually simple and fairly straightforward to implement compared to common techniques. The method avoids explicit multiresolution representations of the shape to allow editing in different scale.

For the sake of speed, in this work we have restricted ourselves to express the differential coordinates in linear terms only. The reconstruction process requires solving a sparse linear least-squares system over the modified region of the surface. We show that this process is fast enough to guarantee interactivity even for detailed mesh regions.

2. Background

2.1. *Mesh Editing*

In this section we briefly overview mesh editing techniques for geometric modeling as they have evolved in the recent years. Early approaches focused on the design of smooth surfaces. Welch and Witkin [26] introduced a variational method for free-from shape design based on arbitrary triangle meshes. An edit operation imposes some geometric boundary conditions, and the modified surface is obtained by an optimization process that minimizes a fairness functional. Taubin [24] improves the efficiency of the optimization by applying Laplacian smoothing which requires only the solution of a sparse linear system. The Laplacian-based fairing operator is carefully developed from a signal processing point of view, revealing the relation to geometric frequencies. Techniques for modeling smooth surfaces are still an active research area [16].

The above work considers the design of smooth surfaces. Shapes that contain geometric details, like those acquired from real-world objects, require special editing tools to preserve the details. The standard approach to detail-preserving uses a multiresolution representation of the mesh. It enables large-scale editing on a coarse level and naturally propagates modifications to the finer levels. The geometric details are usually expressed as some kind of displacements relative to a local coordinate frame [9]. The different levels can be considered as geometric frequencies or resolution of detail, where the coarsest level refers to a smooth surface. Roughly speaking, the editing modifies a coarse level, and the modified version of the next finer level is computed by ”adding” the displacements. This is iterated over the hierarchy until the finest level of the detailed surface is reconstructed.

Zorin [28] present a framework for interactive multiresolution modeling. Their technique is based on input meshes with subdivision connectivity. Kobbelt [14] enable the interactive editing of arbitrary meshes, using a two-band decomposition to encode details between original and a smoothed mesh. The further improvement of the reconstruction leads to multi-band decompositions [10,15].

The encoding scheme of the local detail is critical. Zorin [28] and Guskov [10] use local frames attached to vertices and normal displacements that pierce the original surface and thus lead to resampling. Kobbelt [14] use face-based frames, and the local encoding optimizes the base point of the displacement vector. The encoding was further improved in [15] to avoid artifacts in the reconstruction. In a recent work [5], displacement volumes are applied to prevent local self-intersection in the reconstructed surface. This method requires an iterative, non-linear optimization process. Other works aim at adjusting the vertex density [13] and remeshing the modified surface on the fly, trading computation time for a regular vertex distribution.

A different approach, introduced by Lee [17] parameterizes the region of interest over a planar domain and fits a multiresolution B-spline to the relocated handles. The modified surface is reconstructed from displacements to the spline. This can be interpreted as a kind of simple constraint deformation (*scodef* [3]), well-known for FFD. In this context, Bendels and Klein [2] recently introduced an inherent parameterization by geodesic distances to improve on the constraint fitting and interpolation.

2.2. *Differential Coordinates*

The simplest form of differential coordinates is the Laplacian coordinates. The powerful properties of Laplacian coordinates for mesh representation are not new and have been exploited in various ways. Taubin [24] derives a discrete mesh fairing operator that is applied to model smooth surfaces. Karni and Gotsman [12] take advantage of this extension of spectral theory to arbitrary 3D mesh structures for progressive and compressed geometry coding. Based on Laplacian coordinates, Sorkine [22] derive a geometry compression algorithm that benefits from strong quantization.

Alexa [1] shows that Laplacian coordinates can be effective for morphing and briefly discusses their potential for free-form modeling. He proposes to use differential coordinates to perform local morphing and deformation of the mesh, suggesting differential coordinates as a *local* mesh description, which would be more suitable to constrain under a global deformation of the mesh. This work also mentions the difficulty in using affine-invariant coordinates for mesh representation: the vertex neighborhood cannot always define a local frame (due to linear dependency), and thus the problem is numerically unstable.

In a recent work, Yu et al. [27] introduce an editing technique, formulated by manipulation of the gradients of the coordinate functions (x, y, z) defined on the mesh. The surface is reconstructed by solving the least-squares system resulting from discretizing the Poisson equation $\Delta f = g$ with Dirichlet boundary conditions. As in [18], Yu et al. [27] point out the main problem of this approach: the need to rotate the local frames that define the gradients, or the Laplacians, to preserve the orientation of the local details. They propose to remedy this problem by explicit assignment of the local rotations by propagating the rotation of the editing handle, defined by the user, to all the vertices of the region of interest. If, however, the transformation of the handle consists only of translation, the result might distort the surface details. Another recent work [21] proposes a rotation-invariant local representation of geometry, which avoids the need to assign rotations altogether. However, the reconstruction of the surface geometry is not linear, which hinders interactive applications.

3. Fundamentals

Let $G = (V, E)$ be a 3D triangular mesh, where V denotes the set of vertices of the mesh and E denotes the set of edges. Denote by $\mathbf{p}_j$ the spatial position of vertex j. Let S be a scheme approximating vertices $\mathbf{p}_j \in V$ by linear combination of some other vertices:

$$\mathbf{p}_j \approx S(\mathbf{p}_j) = \sum_{i \in \mathrm{supp}(j), i \neq j} \alpha_{ji} \mathbf{p}_i \tag{1}$$

where $\mathrm{supp}(j)$ denotes the set of vertex indices that scheme S uses to approximate vertex j.

Now, the linear transformation $D(\mathbf{p}_j) = \mathbf{p}_j - S(\mathbf{p}_j)$ is defined as linear differential mesh operator created by scheme S, and $D(V) = V - S(V)$ is defined as differential representation of the mesh created by scheme S. In the next sections we will use such representations for our 3D mesh as a point of departure to our mesh editing algorithm.

A basic example of a linear differential mesh operator created by scheme S is the mesh Laplacian operator:

$$D(\mathbf{p}_j) = L(\mathbf{p}_j) = \mathbf{p}_j - \frac{1}{d_j} \sum_{i:(j,i)\in E} \mathbf{p}_i, \tag{2}$$

where d_j is the valency of vertex j and $S(\mathbf{p}_j) = \frac{1}{d_j}\sum_{(j,i)\in E} \mathbf{p}_i$ is the approximation scheme S.

In general, the operator D can be viewed as a filter of high-frequency detail, i.e. the detail that is missed out by the approximation scheme S. In the case of the Laplacian scheme, D measures the deviation of a vertex from the centroid of its neighbors and thus captures local detail properties of the surface. These are the kind of details that we would like to preserve during an editing operation.

The operator D is linear and can be represented by an $(n \times n)$ matrix M, where $n = |V|$:

$$M_{ij} = \begin{cases} 1 & i = j \\ -\alpha_{ij} & j \in \mathrm{supp}(i) \\ 0 & \text{otherwise} \end{cases}$$

Thus, $(\delta^{(x)}, \delta^{(y)}, \delta^{(z)}) = M(\mathbf{p}^{(x)}, \mathbf{p}^{(y)}, \mathbf{p}^{(z)})$, where $\delta^{(x)}$ is the n-vector of x components of $D(\mathbf{p})$. We call the vector $D(\mathbf{p}_j)$ the *differential coordinates* of vertex j. If $|supp(j)|$ is small then M is a sparse matrix, and the differential coordinates can be efficiently computed.

Given the differential coordinates $\delta^{(x)}, \delta^{(y)}, \delta^{(z)}$ of the mesh, the absolute coordinates of the mesh geometry can be reconstructed by solving the system $M\mathbf{x} = \delta^{(x)}$ (the same goes for $\mathbf{y}$ and $\mathbf{z}$). The matrix M can be singular. For example, in the case of the simple Laplacian (2), $\mathrm{rank}(M) = n - k$ where k is the number of connected components in the mesh [8]. We add spatial constraints to the system to obtain a unique least-squares reconstruction and to control the shape of the surface. To put a (soft) constraint on the position of vertex i, we add the equation $w_i x_i = w_i u_i$ to the system (u_i is the desired location and $w_i > 0$ is the weight that we assign to the constraint). We then solve the resulting system $A\mathbf{x} = \mathbf{b}$ in the least-squares sense.

4. Preserving the orientation of the details

Ideally, relative coordinates should be rotation-invariant, represented in a local coordinate system with respect to some local reference frame. However, the differential coordinates, as defined above, are represented in the global coordinate system, since they are merely an image of a linear transformation of $\mathbb{R}^{3n}$. Therefore they are not rotation-invariant. The transformation of the differential coordinates to local frames (defined at each vertex differently, based on some neighborhood) is not a linear invertible mapping of $\mathbb{R}^{3n}$. While staying in a linear framework has efficiency and simplicity advantages, it brings up the following problem: As a result of an editing operation, certain deformation of the surface is introduced, which typically involves some local rotations. However, the reconstruction of

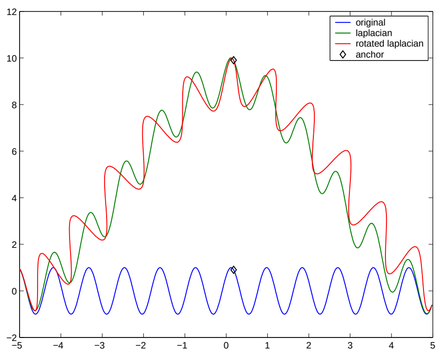

Fig. 1. The differential coordinates are not rotation-invariant. Therefore, when we edit the mesh (lower curve), the orientation of the details with respect to the low-frequency surface is not preserved. To rectify this, we explicitly rotate the differential coordinates (see the dark curve).

the surface from the differential coordinates does not respect the local rotations and therefore the orientation of the reconstructed details will not be preserved and not rotated with the deformed surface. This is demonstrated in Figures 1 and 6.

Recall that the editing of the surface is meant to modify large features of the surface, while keeping the small details locally unchanged. More precisely, we would like to preserve the orientation of the details with respect to the surroundings. To compensate for rotations, we *explicitly* rotate the vectors representing the differential coordinates, while continuing to represent them in the global coordinate system. The rotation is taken to be the local estimation of the transformation applied to the low frequency surface.

More formally, let us consider two meshes M and M', where M' is the mesh obtained from M by an arbitrary editing transformation T. M and M' share the same connectivity and have different geometry. Denote by $\mathbf{p}_j$ and $\mathbf{p}'_j$ the spatial locations of vertex j in M and M', respectively. Let us also define $\mathbf{n}_j$ and $\mathbf{n}'_j$ as the estimates of the normals at vertex j in M and M', computed as an average of the face normals in some neighborhood of the vertex.

The following is an important property of the differential coordinates:

$$R \cdot D(\mathbf{p}_j) = D(R \cdot \mathbf{p}_j), \tag{3}$$

where D is the transformation from absolute to differential coordinates and R a *global* rotation applied to the entire mesh.

The editing transformation T introduces different local rotations across the surface (in addition to stretch, of course). Thus, our key idea is to use the above property of the differential coordinates locally, assuming that locally the rotations are similar.

The local rotation at vertex j is approximated by observing the rotation of an orthogonal frame consisting of $\{\mathbf{n}_j, \mathbf{u}_{ji}, \mathbf{n}_j \times \mathbf{u}_{ji}\}$, where $\mathbf{u}_{ji}$ is a unit vector obtained by projecting some edge (j,i) onto the plane orthogonal to $\mathbf{n}_j$. In other words,

$$\mathbf{u}_{ji} = \frac{\mathbf{v}}{\|\mathbf{v}\|}, \text{ where } \mathbf{v} = (\mathbf{p}_i - \mathbf{p}_j) - \langle \mathbf{p}_i - \mathbf{p}_j, \mathbf{n}_j \rangle \mathbf{n}_j.$$

The rotation of the normal component is defined by $\mathbf{n}_j \leftrightarrow \mathbf{n}'_j$. The rotation of the tangential component is estimated by observing the transformation of the chosen edge (j,i). Among all edges emerging from j, it is best to choose the one whose direction is the closest to being orthogonal to $\mathbf{n}_j$. We can write $D(\mathbf{p}_j)$ in this frame:

$$D(\mathbf{p}_j) = \alpha\mathbf{n}_j + \beta\mathbf{u}_{ij} + \gamma(\mathbf{n}_j \times \mathbf{u}_{ji}).$$

After applying transformation T, the above frame transforms to $\{\mathbf{n}'_j, \mathbf{u}'_{ji}, \mathbf{n}'_j \times \mathbf{u}'_{ji}\}$, where $\mathbf{u}'_{ji}$ is the direction of edge (j,i) in the transformed mesh M', projected onto the plane orthogonal to $\mathbf{n}'_j$. The rotated differential coordinates of vertex j are:

$$D'(\mathbf{p}'_j) = \alpha\mathbf{n}'_j + \beta\mathbf{u}'_{ji} + \gamma(\mathbf{n}'_j \times \mathbf{u}'_{ji}).$$

We define R_1 and R_2 to be *similar rotations* if

$$\|R_1 - R_2\| \approx 0 \tag{4}$$

using some norm induced by a vector norm on $\mathbb{R}^3$. Since property (3) is correct globally, we expect it to be correct for locally similar rotations. Denote by R_j the rotation associated with the vertex j. The normal directions of nearby points over a low-frequency surface do not deviate rapidly (in Section 7 we describe how to achieve ”smooth” normal estimation). Local tangential rotation is also a slow changing parameter for reasonable transformations T. Since R_j is defined using the estimation of normal and tangential rotations of the low-frequency surface, we expect $\|R_i - R_j\|$ to be small for vertices i in the neighborhood of j. Thus, we can expect the property (3) to be valid locally, or in other words, that the reconstructed transformed surface retains the orientation of the details with respect to the underlying low-frequency surface.

In summary, the reconstruction from the rotated differential coordinates consists of the following four steps:

1. Apply a rough deformation T to the mesh.
2. Approximate local rotations R_j.
3. Rotate each differential coordinate $D(\mathbf{p}_j)$ by R_j.
4. Solve the system of $R_j(D(\mathbf{p}_j))$ to reconstruct the edited surface.

5. Implicit derivation of the Laplacian rotations

In this section we suggest an alternative for calculating the rotation R_j, as suggested in [23]. The reconstruction of the geometry of the mesh from its differential coordinates, as described in Section 3, could also be seen as the following variational problem:

$$E(V') = \sum_{i=1}^{n} \left\| \delta_i - \left(\mathbf{v}'_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \mathbf{v}'_j \right) \right\|^2 + \sum_{i=m}^{n} \|\mathbf{v}'_i - \mathbf{u}_i\|^2, \tag{5}$$

which has to be minimized to find a suitable set of coordinates V', where $\delta_i = (\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)})$ is the Laplacian vector of vertex i, and $\mathbf{u}_i$ are the spatial constraints.

As mentioned, the Laplacian coordinates are sensitive to linear transformations. Thus, the detail structure of the shape can be translated, but not rotated or scaled. If the constraints $\mathbf{u}_i$ imply a linear transform, the details are not transformed accordingly.

As was done heuristically in Section 6, our approach is to compute an appropriate transformation R_i for each vertex i based on the eventual configuration of vertices V'. Thus, $R_i(V')$ is a function of V', and we formulate the error functional as

$$E(V') = \sum_{i=1}^{n} \left\| R_i(V')\delta_i - \left(\mathbf{v}'_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \mathbf{v}'_j \right) \right\|^2 + \sum_{i=m}^{n} \|\mathbf{v}'_i - \mathbf{u}_i\|^2. \tag{6}$$

Note that in Eq. 6 both R_i and V' are unknown. However, if the coefficients of R_i are a linear function in V', then solving for V' implies finding R_i (though not explicitly) since $E(V')$ is simply a quadratic function in V'.

The basic idea for a definition of R_i is to derive it from the transformation of $\mathbf{v}_i$ and its neighbors into $\mathbf{v}'_i$ and its neighbors:

$$\min_{R_i} \left(\|R_i \mathbf{v}_i - \mathbf{v}'_i\|^2 + \sum_{j \in \mathcal{N}_i} \|R_i \mathbf{v}_j - \mathbf{v}'_j\|^2 \right). \tag{7}$$

Since this is a quadratic expression, the minimizer is a linear function of V', as required. However, if R_i is unconstrained, the natural minimizer for $E(V')$ is a membrane solution, and all geometric detail is lost. Thus, R_i needs to be constrained in a reasonable way. We have found that R_i should include rotations, isotropic scales, and translations. In particular, we want to disallow anisotropic scales (or shears), as they would allow removing the normal component from Laplacian coordinates.

The translational part of R_i is introduced simply by using homogeneous coordinates. The linear part should satisfy the following conditions: The transformation should be a linear function in the target configuration but constrained to isotropic scales and rotations. The class of matrices representing isotropic scales and rotation can be written as $T = s\exp(H)$, where H is a skew-symmetric matrix. In 3D, skew-symmetric matrices emulate a cross product with a vector, i.e. $H\mathbf{x} = \mathbf{h} \times \mathbf{x}$. The vector $\mathbf{h}$ represents the rotation axis.

Lemma 5.1. *For 3×3 matrices, the exponential $s\exp(H)$ can be represented as $\alpha I + \beta H + \gamma \mathbf{h}^T\mathbf{h}$, where α, β, γ are some scalars.*

Proof. *Let $\mathbf{h} \in \mathbb{R}^3$ be a vector and $H \in \mathbb{R}^{3\times3}$ be a skew-symmetric matrix so that $H\mathbf{x} = \mathbf{h} \times \mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^3$. We are interested in expressing the exponential of H in terms of the coefficients of H, i.e. the elements of $\mathbf{h}$. The matrix exponential is computed using the series expansion*

$$\exp H = I + \frac{1}{1!}H + \frac{1}{2!}H^2 + \frac{1}{3!}H^3 + \ldots$$

The powers of skew-symmetric matrices in three dimensions have particularly simple forms. For the square we find

$$H^2 = \begin{pmatrix} -h_2^2 - h_3^2 & h_1h_2 & h_1h_3 \\ h_1h_2 & -h_1^2 - h_3^2 & h_2h_3 \\ h_1h_3 & h_2h_3 & -h_1^2 - h_2^2 \end{pmatrix} = \mathbf{h}\mathbf{h}^T - \mathbf{h}^T\mathbf{h}\,I \tag{8}$$

and using this expression (together with the simple fact that $H\mathbf{h} = 0$) it follows by induction that

$$H^{2n} = (-\mathbf{h}^T\mathbf{h})^{n-1}\mathbf{h}\mathbf{h}^T + (-\mathbf{h}^T\mathbf{h})^n I$$

and

$$H^{2n-1} = (-\mathbf{h}^T\mathbf{h})^{n-1} H$$

for $n \in \mathbb{N}$. Thus, all powers of H can be expressed as linear combinations of I, H, and $\mathbf{h}\mathbf{h}^T$, and, therefore,

$$\exp H = \alpha I + \beta H + \gamma \mathbf{h}\mathbf{h}^T$$

for appropriate factors α, β, γ. □

Inspecting the terms we find that only s, I, and H are linear in the unknowns s and $\mathbf{h}$, while $\mathbf{h}^T\mathbf{h}$ is quadratic. As a linear approximation of the class of constrained transformations we, therefore, use

$$R_i = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

This matrix is a good linear approximation for rotations with small angles. The consequences for larger angles are discussed later.

Given the matrix R_i as in Eq. 9, we can write down the linear dependency (cf. Eq. 7) of R_i on V', explicitly:

$$\begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_k^{(x)} \\ v_k^{(y)} \\ v_k^{(z)} \\ 1 \end{pmatrix} = \begin{pmatrix} v_k'^{(x)} \\ v_k'^{(y)} \\ v_k'^{(z)} \\ 1 \end{pmatrix}, \quad k \in \{i\} \cup N(i) \tag{10}$$

We wish to write down $s, \mathbf{h}, \mathbf{t}$ as expressions of V' and V. Therefore we reform the above as the following system of equations, where the unknowns are $s, \mathbf{h}, \mathbf{t}$.

$$\begin{pmatrix} v_k^{(x)} & 0 & v_k^{(z)} & -v_k^{(y)} & 1 & 0 & 0 \\ v_k^{(y)} & -v_k^{(z)} & 0 & v_k^{(x)} & 0 & 1 & 0 \\ v_k^{(z)} & v_k^{(y)} & -v_k^{(x)} & 0 & 0 & 0 & 1 \\ \vdots & & & & & & \end{pmatrix} \begin{pmatrix} s \\ h_1 \\ h_2 \\ h_3 \\ t_x \\ t_y \\ t_z \end{pmatrix} = \begin{pmatrix} v_k'^{(x)} \\ v_k'^{(y)} \\ v_k'^{(z)} \\ \vdots \end{pmatrix}, \quad k \in \{i\} \cup N(i), \tag{11}$$

and this system is equivalent to Eq. 7. Denoting the matrix by A_i and the right-hand side vector by $\mathbf{b}_i$, we abbreviate the above as $A_i(s_i, \mathbf{h}_i, \mathbf{t}_i) = \mathbf{b}_i$. We solve this system in the least-squares sense via normal equations:

$$(s_i, \mathbf{h}_i, \mathbf{t}_i)^T = \left(A_i^T A_i\right)^{-1} A_i^T \mathbf{b}_i, \tag{12}$$

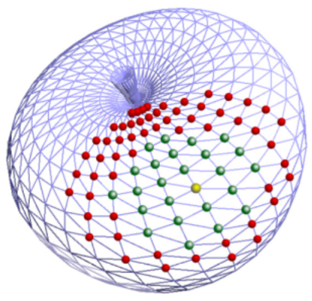

Fig. 2. Classification of the vertices of the edited region (ROI). The yellow vertex is the handle vertex which is moved by the user. The green vertices are the *free* vertices of the ROI (their position changes according to the reconstruction process). The red vertices are the stationary anchors - their position is constrained in the least-squares sense.

which shows that the coefficients of R_i are linear combinations of $\mathbf{v}'_k,\ k \in \{i\} \cup N(i)$, since A_i is known from the initial mesh V. Next, we plug the expressions for R_i into the optimization in Eq. 6 to get a linear least-squares system in unknown V' only.

6. Editing using differential coordinates

From the user's point of view, the editing process is comprised of the following stages: First, the user defines the region of interest (ROI) for editing. Next, the handle vertices are selected. In addition, the user can optionally define the amount of "padding" of the ROI by *stationary anchors*. These stationary anchor vertices support the transition between the ROI and the untouched part of the mesh. The user can also define the type of the differential operator he wishes to use. Finally, the user moves the handle, and the surface is reconstructed with respect to the relocation of the handle and displayed. The last two steps of selecting and then relocating a handle are repeated for the current ROI until the desired surface edit is achieved.

On the algorithmic side, the following steps are performed. Once the ROI, the stationary anchors within and the handle vertices are defined, the mesh vertices are logically partitioned into two groups: the modified vertices, consisting of the ROI, and the rest of the mesh, which is untouched and thus stays fixed. Only the submesh of the modified vertices is considered in the following editing process. The positions of the handle vertices and the anchors constrain the reconstruction and hence the shape of the resulting surface. The handle acts as a control, therefore its constraints are constantly updated. The unconstrained vertices of the edited mesh represent the overall shape and are forced to follow the user interaction. The stationary anchors are responsible for the transition from the ROI to the fixed part of the mesh. The least-squares solution approximates their positions (see also [22]) resulting in a soft blend between the two submeshes. To further improve on the smoothness, we choose several layers of anchors, which are weighted proportional to their geodesic distance from the handle. Selecting the amount of these padding anchor vertices depends on the user's requirements, as mentioned above. We have observed in all our ex-

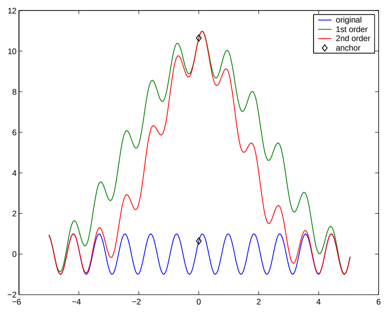

Fig. 3. The effect of editing the mesh (in blue) using different orders of the Laplacian operator. The constrained anchors are the left- and rightmost vertices of the mesh. Pulling the handle vertex in the middle results in the green curve for the 1st-order Laplacian and red curve for the 2nd-order Laplacian operator.

periments that setting the radius of the "padding ring" to be about 10% of the ROI radius gives satisfying results. Figure 2 illustrates the vertex classification.

The edited surface is reconstructed from the locally rotated differential coordinates, as described in Section 4. To approximate the rotations we have to a priori estimate the normals of the editing result. The details of this normal estimation are given in the next section. It is based on a reference shape that is a rough, approximate result of editing the input mesh. Here, we simply use the reconstruction with respect to the not yet rotated differential coordinates as reference. Then our normal estimation approximates the normals of some underlying smooth surface. This approach proved to be effective for estimation of local rotations; however, other types of deformations can be applied to obtain a reference surface, such as a simple constrained deformation [2,3].

In the last step, after applying the local rotations to the differential coordinates of the vertices in the ROI, we reconstruct the surface by solving the linear least-squares system defined in Section 3. The system is constructed from the basic differential operator matrix and the extension of the constrained vertex positions equations. The right-hand side vector contains the rotated differential coordinates together with the constrained locations of the handle and the stationary anchors. The solving procedure is efficiently implemented, as explained in Section 8. We are free to choose an appropriate (linear) differential operator D, such as different orders of the Laplacian. However, a higher-order operator has larger support, resulting in a less sparse system matrix. Figure 3 shows a 2D example of editing a mesh by employing the same constraints and handle movement, by using first- and second-order Laplacian (without applying explicit rotations to the differential coordinates). The latter operator exhibits smoother transition between the stationary vertices and the ROI.

7. Normals estimation

The detail preservation technique introduced in Section 4 requires an approximation of the normals of the underlying smooth surface. A naive estimation can be applied by averaging the normals of the detailed surface in some neighborhood W_j of radius r around the

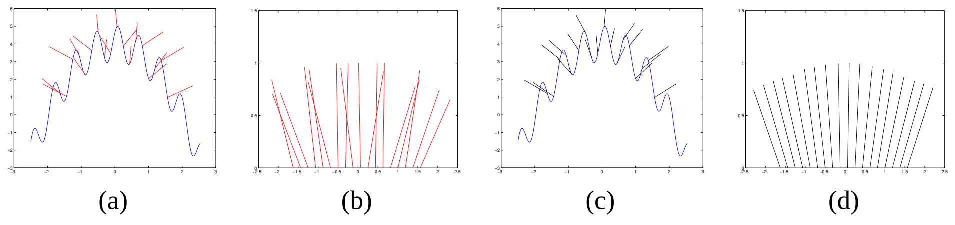

Fig. 4. A 2D example of smooth surface normals estimation. (a) and (c) show the surface with high-frequency detail and the estimated normals of the underlying smooth surface. In (a) a naive averaging of the detailed surface normals was used. (b) shows the same normal vectors as in (a), but the y coordinate of the origin point of each normal is set to zero. This visualizes the problem of the naive estimation - the resulting normals do not vary smoothly. In (c) we show the result of normals estimation using weighted average (with the same support as in (a)–(b)), as explained in Section 7. As demonstrated in (d), such estimation leads to more smoothly-varying normals which are closer to the real smooth surface normals.

estimated vertex j:

$$\mathbf{n}_j = \frac{\mathbf{n}}{||\mathbf{n}||}; \quad \mathbf{n} = \sum_{i \in W_j} \mathbf{n}_i.$$

However, this simple method does not always give satisfactory results since it weighs all the normals equally (see Figure 4(a)–(b) for an example). A better alternative is to use a smooth weighting scheme, where the weights decrease with the distance from the estimated vertex: $\mathbf{n} = \sum_{i \in W_j} w_{ij}\mathbf{n}_i; \quad w_{ij} = p(\text{dist}(\mathbf{p}_j, \mathbf{p}_i))$. The radial function p should be a smooth function vanishing close to r (the estimation support radius). We have chosen to use the polynomial $p(t) = \frac{2}{r^3}t^3 - \frac{3}{r^2}t^2 + 1$. It has the desired properties: $p(0) = 1$, $p'(0) = p'(r) = p(r) = 0$, and it is smooth. The distance measure used should ideally be the geodesic distance between $\mathbf{p}_j$ and $\mathbf{p}_i$; however, it is difficult and computationally costly to compute. Therefore, we retreat to an approximation by computing the length of the weighted shortest path between $\mathbf{p}_j$ and $\mathbf{p}_i$ using Dijkstra's algorithm, where the edges of the mesh graph are assigned weights equal to the edges' length. A more detailed discussion of this choice is given in the next section.

The weighting scheme leads to a smoother approximation of the normals, as can be observed in the 2D example in Figure 4. The figure compares the naive averaging with the elaborated weighted averaging. Note that the supporting neighborhood is the same in both cases.

8. Implementation issues

An interactive editing tool must provide the user an immediate feedback. The critical part of our algorithm is the reconstruction from the differential coordinates, consequently we express them in linear terms only. Thus, the computational kernel of our editing algorithm is a sparse linear solver for the least-squares problem $\min ||A\mathbf{x} - \mathbf{b}||$ over the modified region of the surface. This problem can be solved fast enough to guarantee interactive editing. The speed is gained thanks to a pre-factorization of the coefficient matrix, which permits very fast solves. Hence it is possible to work on large, detailed meshes while maintaining interactive frame-rates.

To solve the linear least-squares system, we use a direct solver for the normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$. The coefficient matrix $A^T A$ is positive semi-definite, and its triangular factorization is computed as $A^T A = R^T R$, where R is an upper triangular matrix. The factorization is the most time-consuming operation, but it only needs to be done once per defined edited region. Once the factorization is available, the system can be solved very efficiently by back-substitution, as many times as necessary. This is required each time the position of the handle vertices is changed, which implies a change of the right-hand side vector $\mathbf{b}$.

In our implementation we use TAUCS version 2.2 [25] as linear solver. It is a direct solver which performs quite fast even on large editing regions, as shown in Table 1. The table displays factorization and solving times for the ROIs that we used in our experiments. The fast solve procedures enable interactive frame-rates when editing complex, detailed meshes. The timings were measured on a 2.4 GHz Pentium 4 computer.

Another implementation issue to address is the computation of geodesic distances needed for our smooth normals estimation and the weighting policy for the stationary anchor points. As explained in Section 6, the anchors' weights are proportional to their respective geodesic distance from the handle. In contrast to [2], these distances are applied merely to aid a smooth transition between the edited region and the fixed part of the mesh. Therefore, we observed that an inexpensive approximation to the geodesic distance is sufficient for our application. We use Dijkstra's algorithm to compute discrete shortest paths, where each edge is weighted by its length.

9. Results and discussion

We demonstrate that representing the geometric information of a triangle mesh in differential form enables detail-preserving interactive shape modeling. The absolute vertex positions are reconstructed from their relative coordinates by solving a sparse linear system. This can be done efficiently, as discussed in the previous section. In fact, we get interactive response for the reconstruction in our experiments. Table 1 provides the computation times for factorization and back-substitution for the shown examples as well as the size of the editing region. Note that the factorization is applied only once per ROI.

Figure 7 shows examples for edits on the *Octopus* model. As the user defines regions of interest of different size, the surface is edited on different scales of detail. In the examples, we padded the outer layers of the ROI with weighted stationary anchors for about 10% of its radius, as explained in Section 6.

The figures show the preservation of details and surface features like the rings of the *Octopus*. In Figures 6 and 7, the estimated local rotations (Section 4) are applied to the differential coordinates to preserve the orientation of the details. Figure 6 illustrates the effect of this operation for a simple height field and for the *Octopus* model. We compare to the reconstruction from coordinates defined with respect to the global coordinate system, which clearly suffers from unnatural distortion of the local detail (note the rings on the arm of the *Octopus*).

Model	ROI	Factor	Solve
Octopus (Figure 7, top)	4,685	0.092	0.005
Octopus (Figure 7, bottom)	12,774	0.568	0.020
Octopus (Figure 6, bottom)	16,792	0.804	0.030
Height field (Figure 6, top)	32,280	1.863	0.069

Table 1. Running times of solving the linear least-squares systems for the different editing regions. *ROI* denotes the number of vertices in the editing region. *Factor* is the time in seconds spent on the factorization of the normal equations. The factorization is performed only once, when the editing region is selected. *Solve* is the time to solve for one mesh function.

Figures 5 and 8 demonstrate some editing results when the local transformations are computed implicitly (Section 5). It can be seen that here as well, the details are well preserved. The example in Figure 8 would be difficult to treat with the heuristic estimation of rotations, because the details are large and hence a lot of smoothing would be required to correctly estimate the normals of the underlying smooth surface. On the other hand, treating large rotational deformations is difficult with this approach, due to the need to linearize rotation matrices in 3D.

All the above examples indicate that our method enables intuitive and flexible shape modeling at interactive frame rates for fairly complex models. For all edits we chose the differential mesh operator D as uniform discretization of the Laplacian also known as the umbrella operator [14]. For our purposes this simplest discretization has been proven to be good enough, but better approximations can be used as well (as in e.g., [6,10]). The order of the Laplacian affects the local support of the operator and hence the sparseness of the system. We plan to investigate the tradeoff between additional computational costs and the benefit for editing.

Our approach is conceptually simple, and its implementation is relatively straightforward. The software consists of two main components: the triangle mesh and a sparse linear solver together with a matrix package. Both components are available in standard libraries (e.g. [4,25]) and can be easily combined. Note that our technique does not require any involved method for multiresolution analysis and synthesis to provide interactive edits of different scale.

10. Conclusion

In this paper we show how a differential representation of vertex coordinates can be exploited for the editing of arbitrary triangle meshes. The use of this representation leads to a conceptually simple yet powerful method for interactive, feature-preserving shape modeling method. Thanks to local rotations of the relative coordinates the orientation of the details are preserved. Our examples show the effectiveness and efficiency of the method for fairly complex input meshes. In particular, we show that a simple and intuitive modeling tool provides results quickly, while preserving the local surface details.

As we discussed, the value of relative coordinates and in particular Laplacian coordinates, have been recently pronounced in other applications like mesh morphing and geom-

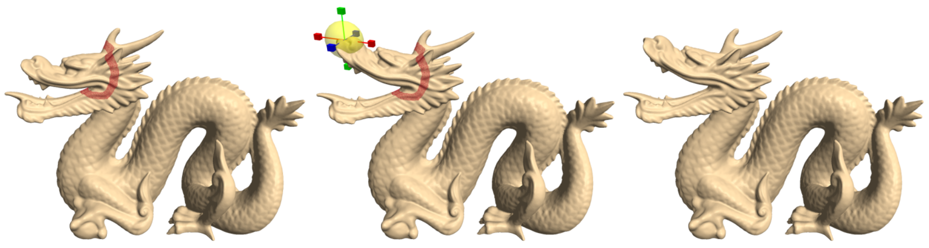

Fig. 5. Example of editing results using implicit optimization of local transformations. (a) The user selects the region of interest – the upper lip of the dragon, bounded by the belt of stationary anchors (in red). (b) The chosen handle (enclosed by the yellow sphere) is manipulated by the user: translated and rotated. (c) The editing result.

etry compression. We believe that differential coordinates have a lot more potential in digital geometry processing. For instance, this includes the extension of more digital image processing techniques that employ differential operators, like in [19], to meshes, which we plan to investigate on in the future, as well as on alternative representations of differential coordinates.

Acknowledgements

We would like to thank Sivan Toledo and Seungyong Lee for valuable discussions and Mark Pauly for providing the *Octopus* model. This work was supported in part by the German Israel Foundation (GIF) and by grants from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), the Israeli Ministry of Science and by the EU research project 'Multiresolution in Geometric Modelling (MINGLE)' under grant HPRN-CT-1999-00117.

References

1. Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003.
2. G. H. Bendels and R. Klein. Mesh forging: editing of 3D-meshes using implicitly defined occluders. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 207–217, 2003.
3. Paul Borrel and Ari Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, April 1994.
4. M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Openmesh a generic and efficient polygon mesh data structure. In *OpenSG Symposium 2002*, 2002.
5. Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum (Eurographics 2003)*, 22(3):483–491, 2003.
6. Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH 99*, pages 317–324, 1999.
7. Gerald Farin. *Curves and surfaces for computer aided geometric design: a practical guide*. Academic Press, 1992.

8. M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. Journal*, 23:298–305, 1973.
9. David Forsey and Richard Bartels. Hierarchical b-spline refinement. In *Proceedings of ACM SIGGRAPH 88*, pages 205–212, 1988.
10. Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *Proceedings of ACM SIGGRAPH 99*, pages 325–334, 1999.
11. J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, 1993.
12. Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, pages 279–286, 2000.
13. L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), pages 249–260, 2000.
14. Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH 98*, pages 105–114, 1998.
15. Leif Kobbelt, Jens Vorsatz, and Hans-Peter Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry: Theory and Applications*, 14:5–24, 1999.
16. Loïc Le Veuvre. Modelling and deformation of surfaces defined over finite elements. In *Proceedings of Shape Modeling International*, pages 175–183, 2003.
17. Seungyong Lee. Interactive multiresolution editing of arbitrary meshes. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics 1999)*, volume 18(3), pages 73–82, 1999.
18. Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.
19. Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Proceedings of ACM SIGGRAPH 2003*, pages 313–318, 2003.
20. Peter Schröder and Denis Zorin. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000.
21. A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proceedings of 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 68–75, 2004.
22. Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 42–51, 2003.
23. Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 179–188. Eurographics Association, 2004.
24. Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH 95*, pages 351–358, 1995.
25. Sivan Toledo. TAUCS*: A Library of Sparse Linear Solvers, version 2.2*. Tel-Aviv University, Available online at *http://www.tau.ac.il/ stoledo/taucs/*, September 2003.
26. William Welch and Andrew Witkin. Free–Form shape design using triangulated surfaces. In *Proceedings of ACM SIGGRAPH 94*, pages 247–256, 1994.
27. Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. In *Proceedings of ACM SIGGRAPH 2004*, pages 641–648, 2004.
28. Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *Proceedings of ACM SIGGRAPH 97*, pages 259–268, 1997.

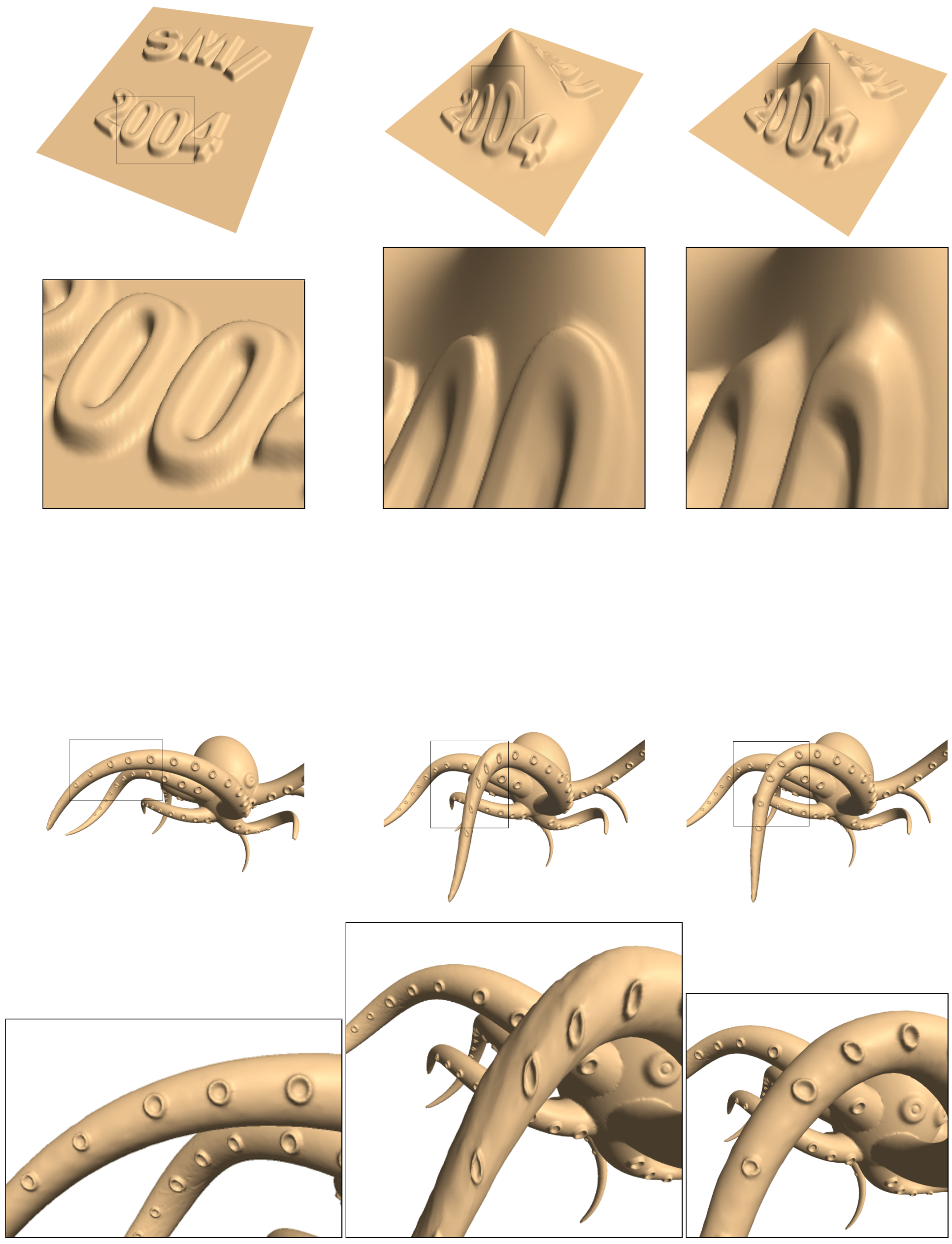

Fig. 6. The effect of applying local rotations to the differential coordinates. The left column displays the original models. Middle column shows an edit performed *without* local rotations. Note the distortion of the letters and the circle stamps. The right column shows the result of the same editing operation *with* local rotations applied. The orientation of the details is much better preserved.

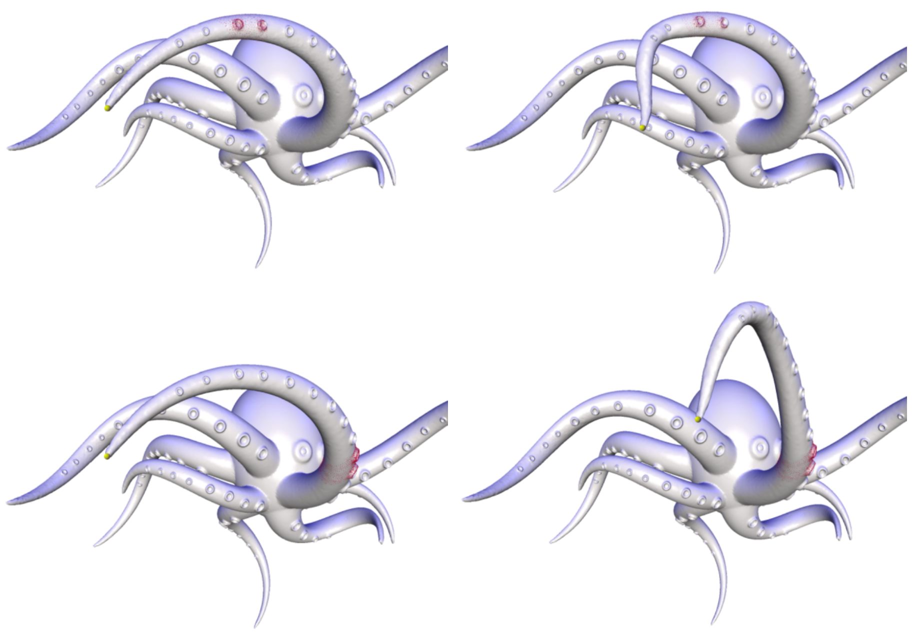

Fig. 7. Defining different ROIs and applying the editing technique. The handle vertex is located at the tip of the front arm, marked by the bright sphere. The left column displays the original model with anchor vertices shown by small dots (they mark the padded boundary of the ROI). In the right column the result of an editing operation is displayed. The small ROI in the top row results in a local change of the shape of the arm, whereas the larger ROI in the bottom row allows for a more global deformation.

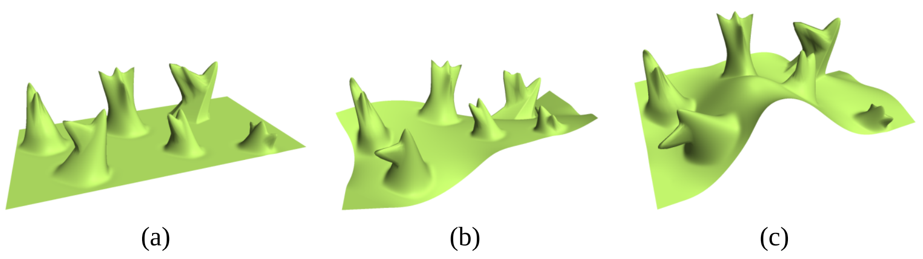

Fig. 8. Deformations of a model (a) with detail that cannot be expressed by height field. The deformation changes the global shape while respecting the structural detail as much as possible. The local transformations were optimized implicitly.