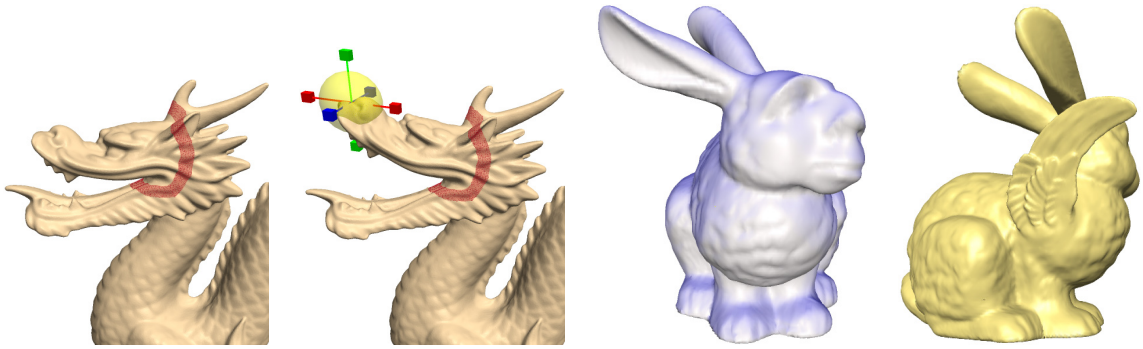
Laplacian Surface Editing

Olga Sorkine Daniel Cohen-Or Yaron Lipman
School of Computer Science
Tel Aviv University

Marc Alexa
Discrete Geometric Modeling Group
Darmstadt University of Technology

Christian Rössl Hans-Peter Seidel
Max-Planck Institut für Informatik



(a) (b) (c) (d)

Figure 1: Advanced mesh editing operations using Laplacian coordinates: free-form deformations (a-b), coating transfer (c) and mesh transplanting (d). Representing the geometry using the Laplacian coordinates enables preservation of detail.

Abstract

Surface editing operations commonly require geometric details of the surface to be preserved as much as possible. We argue that geometric detail is an intrinsic property of a surface and that, consequently, surface editing is best performed by operating over an intrinsic surface representation. We provide such a representation of a surface, based on the Laplacian of the mesh, by encoding each vertex relative to its neighborhood. The Laplacian of the mesh is enhanced to be invariant to locally linearized rigid transformations and scaling. Based on this Laplacian representation, we develop useful editing operations: interactive free-form deformation in a region of interest based on the transformation of a handle, transfer and mixing of geometric details between two surfaces, and transplanting of a partial surface mesh onto another surface. The main computation involved in all operations is the solution of a sparse linear system, which can be done at interactive rates. We demonstrate the effectiveness of our approach in several examples, showing that the editing operations change the shape while respecting the structural geometric detail. [1]

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—curve, surface, solid and object representations

Keywords: mesh editing, detail preservation

1 Introduction

Surfaces in computer graphics are mostly represented in global coordinate systems: explicit representations are based on points, vertices, or nodes that are typically described using absolute Euclidean coordinates. Implicit representations describe the shape as the level set of a function defined in Euclidean space. A global coordinate system is the natural choice for all operations involving other objects such as rendering, intersection testing and computation, transformations, or CSG modeling. On the other hand, for local surface modeling, it would be desirable that the representation captures the local shape (i.e. the intrinsic geometry of the surface) rather than the absolute position or orientation in Euclidean space.

Manipulating and modifying a surface while preserving the geometric details is important for various surface editing operations, including free-form deformations [Sederberg and Parry 1986; Coquillart 1990], cut and paste [Ranta et al. 1993; Kuriyama and Kaneko 1999; Biermann et al. 2002], fusion [Kanai et al. 1999], morphing [Alexa 2003], and others. Note that the absolute position of the vertices in a mesh is not important for these operations, which calls for an intrinsic surface representation.

A partially intrinsic surface mesh representation is multi-resolution decompositions [Forsey and Bartels 1988; Zorin et al. 1997; Kobbelt et al. 1998] [Kobbelt et al. 1999; Guskov et al. 1999; Capell et al. 2002]. In a multi-resolution mesh, the geometry is encoded as a base mesh and several levels of refinement. The refinement is typically described locally, so that the geometric details are mostly captured in a discrete set of intrinsic coordinates. Using this representation, several modeling operations can be performed on an appropriate user-specified level-of-detail.

Our approach to encoding geometric details is to use differential coordinates for the vertices. This provides an intrinsic representation of the surface mesh, where the reconstruction of global coordinates from this representation always preserves the local geometry as much as possible given the modeling constraints. Using a differential representation for editing operations has been shown to be quite effective in image domain [Fattal et al. 2002; Pérez et al. 2003]. Image domain has a natural regular parameterization and a

[1] The conference version of this paper will be presented at the Eurographics Symposium on Geometry Processing 2004.

resulting inherent definition of a gradient, which allows modeling many editing tasks as a discrete Poisson equation. However, this approach cannot be directly applied or adapted to discrete (as well as continuous) surfaces.

We rather realize an approach to surface mesh editing based on encoding each vertex relative to the centroid of its topological neighbors. The difference of a vertex position from the centroid of its neighbors is known as a Laplacian coordinate [Alexa 2003; Karni and Gotsman 2000; Sorkine et al. 2003; Lipman et al. 2004]. Laplacian coordinates are a linear function of the global mesh geometry, which allows efficient converting between absolute and intrinsic representations by solving a sparse linear system. Laplacian coordinates are invariant under translation (of absolute geometry), but they are not invariant to scaling and rotation, which poses the main practical problem.

We provide a technique that makes Laplacian coordinates invariant to rotation and isotropic scaling. Using this technique, we develop useful surface editing operations, which preserve the intrinsic geometry of the surface as much as possible given the constraints of the modeling operations. The major contributions of this work are:

Rotation and scale invariant (RSI) Laplacian coordinates: We reformulate the process of least squares fitting of the Euclidean geometry to the given Laplacian coordinates. In our fitting process, we implicitly compute an appropriate transformation per vertex, which is applied to the respective Laplacian coordinate. This leads to Laplacian coordinates that are almost insensitive to rotation and scaling.

Interactive detail-preserving surface editing: Based on the RSI Laplacian coordinates, we develop an interactive editing system. The user interactively deforms a region of the surface by manipulating a handle. The transformation of the handle induces a global deformation that resembles the outcome of manipulating an object made of some physical soft material.

Transfer of geometric detail (*coating*): Since the detail is captured in the Laplacian coordinates, we are able to "peel" high-frequency details from one surface and transfer them to another. The method can be applied to arbitrary homeomorphically parameterized surface patches.

Transplanting surface patches with homeomorphic boundaries: Our transplanting technique only requires that the surfaces have matching topology at the boundaries; the surface patches within the boundaries need not match. A seamless transition with gradual change of detail from one part to another results from blending the Laplacian representations of the parts.

2 Related work

Editing three-dimensional shapes has been an important research area in geometric modeling and computer graphics. The dominating approach for (free-form) designing of a surface from scratch is based on parametric surfaces (see e.g., [Farin 1992; Hoschek and Lasser 1993]), which can be generalized to non-regular base domains using subdivision techniques [Schröder and Zorin 2000].

However, we are interested in editing an existing surface, probably acquired with scanning devices. If the surface is smooth, modifications should remain smooth [Welch and Witkin 1994; Taubin 1995; Le Veuvre 2003]. If the surface contains geometric details (e.g. a sharp feature), these details should be preserved. The editing operation should naturally change the shape and simultaneously respect the structural detail. The standard approach to detail-preserving modeling operations uses a multi-resolution representation of the mesh. The geometric details are usually expressed relative to a local coordinate frame [Forsey and Bartels 1988; Zorin et al. 1997; Kobbelt et al. 1998; Kobbelt et al. 1999; Guskov et al. 1999]. The different levels can be considered as frequencies of the geometry. The coarsest level refers to the smoothest surface and adding finer levels introduces smaller details. Editing operations can be performed on coarse levels, and the so modified shape is computed by "adding" the displacements in their local coordinate frames.

The problem of basis elements with large support in multi-resolution representations has motivated differential representations for image editing [Pérez et al. 2003]. Note that the completely local and intrinsic differential representation comes at the expense of a global reconstruction computation (e.g. the solution of a global PDE), while the generation of absolute coordinates from multi-resolution representations is restricted to the modified bases.

Our motivation is similar to image editing methods based on PDEs. We propose a local differential representation, at the expense of a global reconstruction from differential to absolute geometry. Specifically, the modified surface is reconstructed by solving a sparse linear system. Using state-of-the-art solvers this turns out to be very fast and adequate for interactive systems, even for editing operations on large meshes.

The potential of differential coordinates for free-form modeling is briefly discussed by Alexa [Alexa 2003]. He specifically discusses the difficulty of deriving affine-invariant coordinates for mesh representation as the vertex neighborhood may be degenerated (i.e. planar) and, even more difficult, near-degenerate situations make the reconstruction numerically intractable.

In a recent work, Yu et al. [Yu et al. 2004] 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. Lipman et al. [Lipman et al. 2004] reconstruct the surface from discrete Laplacians of the mesh functions and spatial boundary conditions by solving a very similar least-squares system. Both works 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. Lipman et al. [Lipman et al. 2004] estimate the local rotations of the frames on the underlying smooth surface, and Yu et al. [Yu et al. 2004] propagate the rotation of the editing handle, defined by the user, to all the vertices of the region of interest. In contrast to these explicit solutions, in this paper we introduce a method that implicitly transforms the differential coordinates based on finding an *optimal* transform for each vertex. The transform is defined by a linear expression of local coordinates and a sparse set of control points. The solution of this linear system strives to preserve the size and the orientation of the differential coordinates and consequently of the surface details.

We focus our work on meshes as they are the dominating representation of surfaces these days. Other surface representations are advantageous for certain modeling operations. Implicit surfaces allow easy blending, space warping, and CSG modeling [Rockwood 1989; Guy and Wyvil 1995; Pasko et al. 1995; Wyvill et al. 1999]. The recently popular level-set approach yields a particularly simple formulation and implementation of these operations [Museth et al. 2002] based on the discrete and regular representation of a distance field. Adaptively sampled distance fields [Frisken et al. 2000] provide a discrete surface representation with controlled error. All of these essentially implicit representations allow changing the topology of the surface during modeling. Point-sampled surfaces are related to meshes; however, explicit information about the topology is missing. This has advantages for some operations [Pauly et al. 2003], though sometimes requires surface reconstruction steps to add more points to the representation.

3 Fitting transformed Laplacian coordinates

Let the mesh $\mathscr{M}$ be described by a pair (K,V), where K describes the connectivity and $V = \{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ describes the geometric positions of the vertices in $\mathbb{R}^3$. We use the following terminology: the *neighborhood ring* of a vertex i is the set of adjacent vertices $\mathscr{N}_i = \{j|(i,j) \in K\}$ and the *degree* d_i of this vertex is the number of elements in $\mathscr{N}_i$. We assume that the mesh is connected.

Instead of using absolute coordinates V, we would like to describe the mesh geometry using a set of differentials $\Delta = \{\delta_i\}$. Specifically, coordinate i will be represented by the difference between $\mathbf{v}_i$ and the average of its neighbors:

$$\delta_i = \mathscr{L}(\mathbf{v}_i). \tag{1}$$

For simplicity, we define $\mathscr{L}$ with uniform weights:

$$\mathscr{L}(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i}\sum_{j\in\mathscr{N}_i}\mathbf{v}_j. \tag{2}$$

These weights proved to be sufficient in all our experiments. However, our approach does not depend on the particular choice of $\mathscr{L}$. For instance, the cotangent weights (see, e.g. [Desbrun et al. 1999]) would accommodate extremely non-uniform tessellations, and their application is straightforward. The transformation between V and Δ can be described in matrix algebra. Let A be the mesh adjacency matrix and $D = \mathrm{diag}(d_1,\ldots,d_n)$ be the degree matrix. Then $\Delta = LV$, where $L = I - D^{-1}A$ for the uniform weights. The matrix L is commonly considered as the Laplacian operator of the mesh with connectivity A [Taubin 1995; Karni and Gotsman 2000], which is why we call δ_i the *Laplacian coordinate* of vertex i. Laplacian coordinates are invariant under translation, but sensitive to linear transforms. L has rank $n-1$, which means V can be recovered from Δ by fixing one vertex and solving a linear system.

The approach to performing modeling operations using Laplacian coordinates Δ is to fix the absolute position of several vertices (see [Alexa 2003]), i.e.,

$$\mathbf{v}'_i = \mathbf{u}_i, \quad i \in \{m,\ldots,n\},\ m < n \tag{3}$$

and solve for the remaining vertices $\{\mathbf{v}'_i\},\ i \in \{1,\ldots,m-1\}$ by fitting the Laplacian coordinates of the geometry V' to the given Laplacians Δ. It has been observed that the solution behaves better if the constraints $\{\mathbf{u}_i\}$ are satisfied in a least squares sense rather than exactly [Sorkine et al. 2003; Lipman et al. 2004]. This results in the following error functional:

$$E(V') = \sum_{i=1}^{n}\left\|\delta_i - \mathscr{L}(\mathbf{v}'_i)\right\|^2 + \sum_{i=m}^{n}\|\mathbf{v}'_i - \mathbf{u}_i\|^2, \tag{4}$$

which has to be minimized to find a suitable set of coordinates V'. Solving this quadratic minimization problem results in a sparse linear system of equations.

The rationale of fitting given Laplacian coordinates is that details of the shape are preserved, as the relative location of vertices is encoded in Δ. As mentioned, however, these 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.

The main idea of our approach is to compute an appropriate transformation T_i for each vertex i based on the eventual new configuration of vertices V'. Thus, $T_i(V')$ is a function of V' and we formulate the error functional as

$$E(V') = \sum_{i=1}^{n}\left\|T_i(V')\delta_i - \mathscr{L}(\mathbf{v}'_i)\right\|^2 + \sum_{i=m}^{n}\|\mathbf{v}'_i - \mathbf{u}_i\|^2. \tag{5}$$

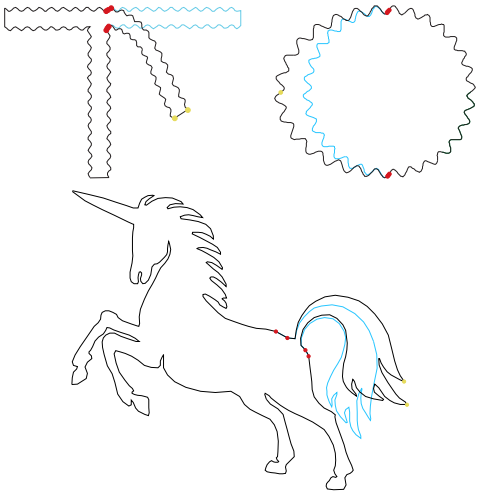


Figure 2: Editing 2D meshes using Laplacian-coordinates fitting. The red dots denote fixed anchor points and the yellow ones are the pulled-handle vertices. The original meshes are colored blue.

Note that in Eq. 5 both T_i and V' are unknown. However, if the coefficients of T_i are a linear function in V', then solving for V' implies finding T_i (though not explicitly) since $E(V')$ is simply a quadratic function in V'.

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

$$\min_{T_i}\left(\|T_i\mathbf{v}_i - \mathbf{v}'_i\|^2 + \sum_{j\in\mathscr{N}_i}\|T_i\mathbf{v}_j - \mathbf{v}'_j\|^2\right). \tag{6}$$

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

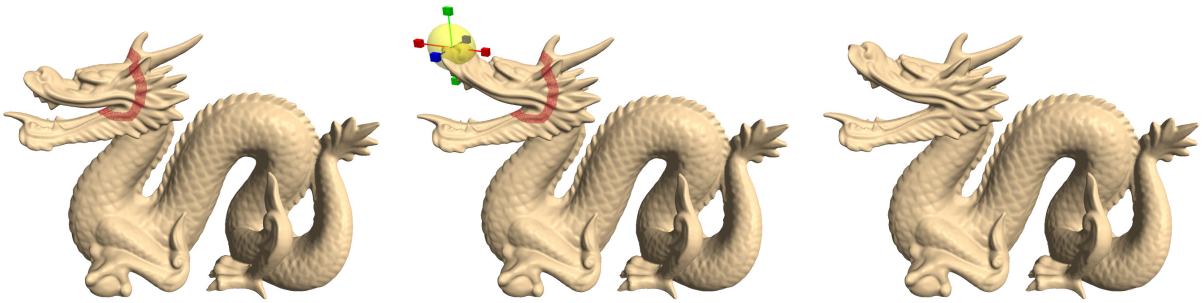
The translational part of T_i is introduced simply by using homogeneous coordinates. The linear part should satisfy the following conditions: The transformation should be a linear function in the target configuration but constrained to isotropic scales and rotations. The class of matrices representing 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. $H\mathbf{x} = \mathbf{h}\times\mathbf{x}$. Drawing upon several other properties of 3×3 skew matrices (see Appendix A), one can derive the following representation of the exponential above:

$$s\exp H = s(\alpha I + \beta H + \gamma\mathbf{h}^T\mathbf{h}). \tag{7}$$

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[2]. As a linear approximation of the class of constrained transformations we, therefore,

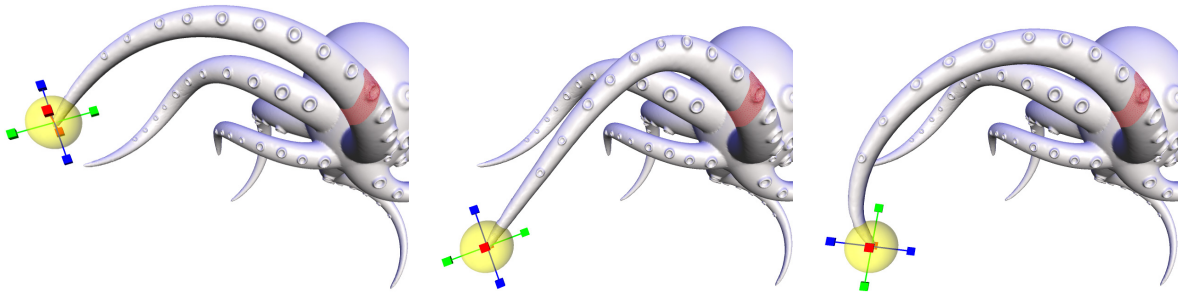
[2]Figure 2 illustrates editing of a 2D mesh. Note that in 2D the matrices of class $s\exp(H)$ can be completely characterized with the linear expression

$$T_i = \begin{pmatrix} a & w & t_x \\ -w & a & t_y \\ 0 & 0 & 1 \end{pmatrix}.$$



(a) (b) (c)

Figure 3: The editing process. (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.



(a) (b) (c)

Figure 4: Different handle manipulations. (a) The region of interest (arm), bounded by the belt of stationary anchors, and the handle. (b) Translation of the handle. (c) Subsequent handle rotation. Note that the detail is preserved in all the manipulations.

use

$$T_i = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{8}$$

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

Given the matrix T_i as in Eq. 8, we can write down the linear dependency (cf. Eq. 6) of T_i on V', explicitly. Let $(s_i, \mathbf{h}_i, \mathbf{t}_i)^T$ be the vector of the unknowns in T_i. Then we wish to minimize

$$\|A_i(s_i, \mathbf{h}_i, \mathbf{t}_i)^T - \mathbf{b}_i\|^2, \tag{9}$$

where A_i contains the positions of $\mathbf{v}_i$ and its neighbors and $\mathbf{b}_i$ contains the position of $\mathbf{v}_i'$ and its neighbors. The structure of $(s_i, \mathbf{h}_i, \mathbf{t}_i)^T$ yields

$$A_i = \begin{pmatrix} v_{k_x} & 0 & v_{k_z} & -v_{k_y} & 1 & 0 & 0 \\ v_{k_y} & -v_{k_z} & 0 & v_{k_x} & 0 & 1 & 0 \\ v_{k_z} & v_{k_y} & -v_{k_x} & 0 & 0 & 0 & 1 \\ \vdots & & & & & & \end{pmatrix}, \; k \in \{i\} \cup \mathcal{N}_i, \tag{10}$$

and

$$\mathbf{b_i} = \begin{pmatrix} v'_{k_x} \\ v'_{k_y} \\ v'_{k_z} \\ \vdots \end{pmatrix}, \; k \in \{i\} \cup \mathcal{N}_i. \tag{11}$$

The linear least-squares problem above is solved by

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

which shows that the coefficients of T_i are linear functions of $\mathbf{b}_i$, since A_i is known from the initial mesh V. The entries of $\mathbf{b}_i$ are simply entries of V' so that $(s_i, \mathbf{h}_i, \mathbf{t}_i)$ and, thus, T_i is a linear function in V', as required.

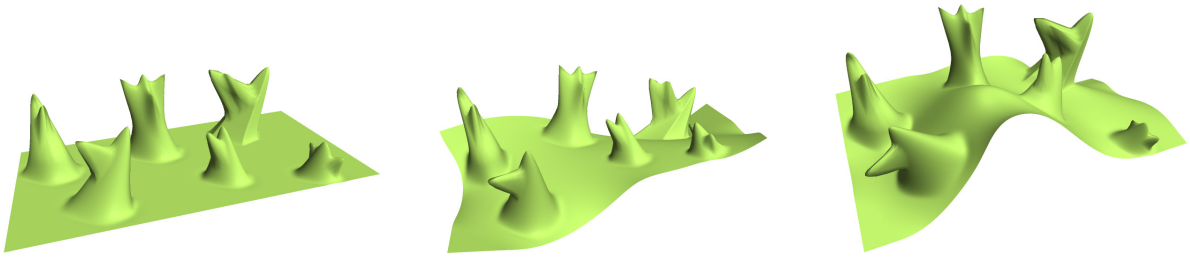
3.1 Adjusting T_i

In many modeling situations, solving for absolute coordinates in the way explained above is sufficient. However, there are two exceptions that require adjusting the transformations:

1. As mentioned, T_i does not exactly represent the class of isotropic scales and rotations. For large angles ϕ around the axis $\mathbf{h}/\|\mathbf{h}\|$ the space is scaled along $\mathbf{h}/\|\mathbf{h}\|$ with a factor of $\cos\phi$.

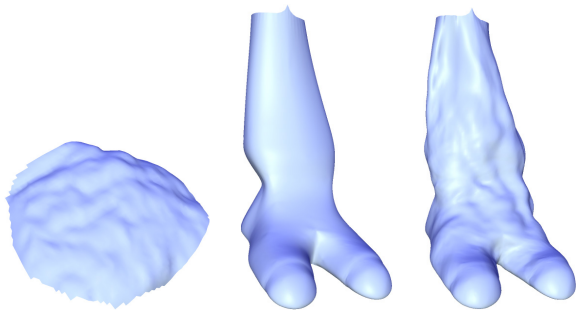
2. Sometimes anisotropic scaling is the wanted free-form deformation, e.g., the dislocation of a single vertex typically implies a stretch in only one direction.

Both situations are handled in a similar way: The current set of transformations $\{T_i\}$ is computed from V and V'. Then each T_i is inspected, the corresponding Laplacian coordinate δ_i is updated appropriately depending on the cases above, and the system is solved again. In the case of too large angles of rotations, it is possible to first apply an approximated reconstruction using the method in [Lipman et al. 2004] and then refine it with our technique, such that smaller rotations are involved. In the case of wanted anisotropic scaling, the $\{\delta_i\}$ are scaled by the inverse of the scale implied by the constraints. See Figure 4 for an example of large rotations.



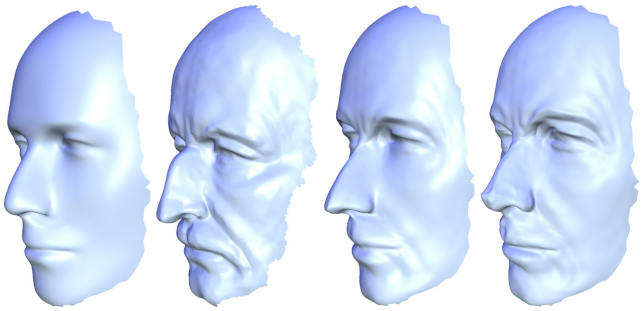
(a) (b) (c)

Figure 5: 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.



(a) (b) (c)

Figure 6: Coating transfer; The coating of the Bunny (a) is transferred onto the mammal's leg (b) to yield (c).



(a) (b) (c) (d)

Figure 7: The coating of the *Max Planck* is transferred onto the *Mannequin*. Different levels of smoothing were applied to the *Max Planck* model to peel the coating, yielding the results in (c) and (d).

4 Mesh editing

There are many different tools to manipulate an existing mesh. Perhaps the simplest form consists of manipulating a *handle*, which is a set of vertices that can be moved, rotated and scaled by the user. The manipulation of the handle is propagated to the shape such that the modification is intuitive and resembles the outcome of manipulating an object made of some physical soft material. This can be generalized to a free-form deformation tool which transforms a set of control points defining a complex of possibly weighted handles, enabling other modeling metaphors to be mimicked (see e.g., the recent work of [Bendels and Klein 2003] and the references therein).

The editing interaction is comprised of the following stages: First, the user defines the region of interest (ROI) for editing. The ROI is defined by the closed simple loop of its boundary edges. Next, the handle inside the ROI is defined. In addition, the user can optionally define the amount of "padding" of the ROI by *stationary anchors*. These stationary anchors form a *belt* that supports the transition between the ROI and the untouched part of the mesh. Then, the user manipulates the handle, and the surface is reconstructed with respect to the relocation of the handle and displayed.

The submesh of the ROI is the only part considered during the editing process. The positions of the handle vertices and the stationary anchors constrain the reconstruction and hence the shape of the resulting surface. The handle is the means of user control, and therefore, its constraints are constantly updated. The unconstrained vertices of the submesh are repeatedly reconstructed to follow the user interaction. The stationary anchors are responsible for the transition from the ROI to the fixed untouched part of the mesh, resulting in a soft transition between the submesh and stationary part of the mesh. Selecting the amount of padding by anchor vertices depends on the user's requirements, as mentioned above. We have observed in all our experiments that setting the radius of the "padding ring" to be up to 10% of the ROI radius gives satisfying results.

The reconstruction of the submesh requires solving the linear least-squares system as described in Section 3. The method of building the system matrix (Eq. 12), including the computation of a sparse factorization, is relatively slow, but constructed only once when the ROI is selected. The user interaction with the handle requires solely updating the positions of the handle vertices in the right-hand-side vector, and solving.

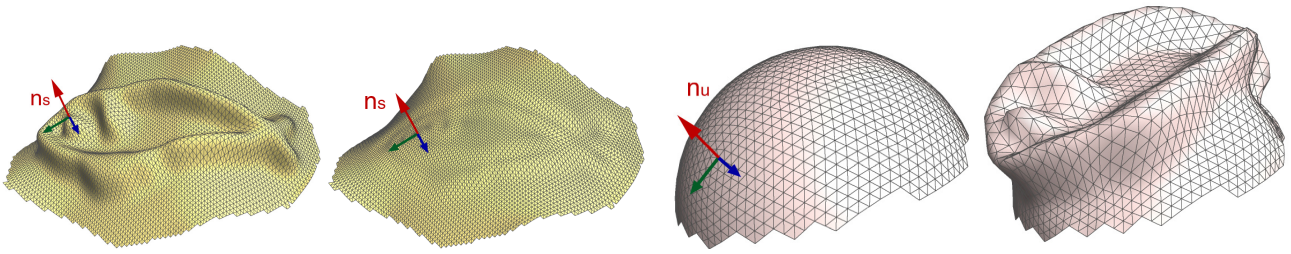
Figures 3 and 4 illustrate the editing process. Note that the details on the surface are preserved, as one would intuitively expect. Figure 5 demonstrates deformation of a model with large extruding features which cannot be represented by a height field.

5 Coating transfer

Coating transfer is the process of peeling the coating of a *source* surface and transferring it onto a *target* surface. See Figure 6 for an example of such an operation. We use the term *coating* to refer to the high-frequency surface details. More precisely, the coating is defined as the difference between the original surface and a low-frequency band of the surface. Let S be the source surface from which we would like to extract the coating, and let $\tilde{S}$ be a smooth version of S. The surface $\tilde{S}$ is a low-frequency surface associated with S, which can be generated by filtering (see, e.g., [Desbrun et al. 1999]). The amount of smoothing is a user-defined parameter, and depends on the range of detail that the user wishes to transfer.

We encode the coating of a surface based on the Laplacian coordinates. Let δ_i and $\tilde{\delta}_i$ be the Laplacian coordinates of the vertex i in S and $\tilde{S}$, respectively. We define ξ_i to be the encoding of the coating at vertex i defined by

$$\xi_i = \delta_i - \tilde{\delta}_i \,. \tag{13}$$



(a) (b) (c) (d)

Figure 8: Coating transfer. The orientation of a coating detail (a) is defined by the normal at the corresponding vertex in the low frequency surface in (b). The transferred coating vector needs to be rotated to match the orientation of the corresponding point in (c) to reconstruct (d).

The values of ξ_j encode the coating of S, since given the bare surface $\tilde{S}$ we can recover the original coating simply by adding ξ_j to $\tilde{\delta}_i$ and reconstructing S with the inverse Laplacian transform L^{-1}. That is,

$$S = L^{-1}(\tilde{\delta} + \xi)\,. \tag{14}$$

In this case of a coating transfer of S onto itself, S is faithfully reconstructed. However, in general, instead of coating $\tilde{S}$ with ξ, we would like to add the coating ξ onto an arbitrary surface U. If the target surface U is not smooth, it can be smoothed first, and then the coating transfer applied. Before we move on, we should note that the coating transfer from S onto $\tilde{S}$ is simple, since the neighborhoods of the corresponding vertices i have the same *orientation*. We define the orientation of a vertex i in a surface S by the local frame of i on $\tilde{S}$. Loosely speaking, the orientation of a point reflects the general orientation of its neighborhood, without respecting the high frequencies of the surface.

When applying a coating transfer between two surfaces, the coating ξ should first be aligned, or rotated with respect to the target. This compensates for the different local surface orientations of corresponding points in the source and target surfaces.

The following is an important property of the Laplacian coordinates:

$$R \cdot L^{-1}(\delta_j) = L^{-1}(R \cdot \delta_j)\,, \tag{15}$$

where L^{-1} is the transformation from Laplacian coordinates to absolute coordinates, and R a *global* rotation applied to the entire mesh. The mapping between corresponding points in S and U defines different local orientations across the surfaces. Thus, our key idea is to use the above property of the Laplacian coordinates locally, assuming that, locally, the rotations are close to each other (in induced norm).

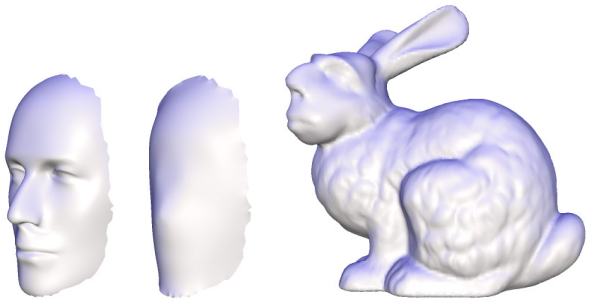
5.1 Coating

Assume that the source surface S and the target surface U share the same connectivity, but have different geometries, and that the correspondence between their vertices is given. In the following we generalize this to arbitrary surfaces.

The local rotation R_i at each vertex i in S and U is taken to be the local rotation between their corresponding frames (see Figure 8). The frame of vertex i in S is defined by its normal $\mathbf{n}_s$ and the normalized projection of some edge e_s emanating from i onto the tangent plane defined by $\mathbf{n}_s$ (the third vector is determined by the right-hand product of the first two). The corresponding frame in U is established by $\mathbf{n}_u$ (the normal of i in U) and the projection of the edge e_u which corresponds to e_s. Denote the rotated coating encoding of vertex i by $\xi_i' = R_i(\xi_i)$. Having all the R_i associated with the ξ_i, the coating transfer from S onto U is expressed as follows:

$$U' = L^{-1}(\Delta + \xi') \tag{16}$$

where Δ denotes the Laplacian coordinates of the vertices of U. Now the new surface U' has the coating of U.



(a) (b) (c)

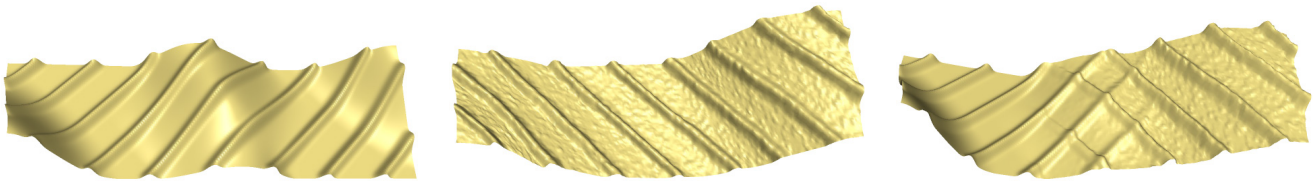
Figure 9: Transferring the coating of the *Mannequin* onto the face of the *Bunny*. (a) The source surface S. It is significantly smoothed to peel the coating. (b) The smoothed surface $\tilde{S}$. (c) The result of coating transfer onto the *Bunny*.

5.2 Mapping and resampling

So far we have assumed that the source and target meshes (S and U) share the same connectivity, and hence the correspondence is readily given. However, the coating transfer between arbitrary surfaces is more involved. To sample the Laplacian coordinates, we need to define a mapping between the two surfaces.

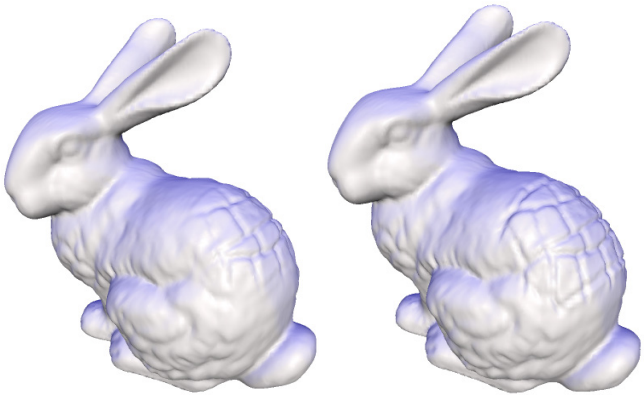
This mapping is established by parameterizing the meshes over a common domain. Both patches are assumed to be homeomorphic to a disk, so we may choose either the unit circle or the unit square as a common domain. We apply the mean-value coordinate parameterization [Floater 2003], as it efficiently produces a quasi-conformal mapping, which is guaranteed to be valid for convex domains. We fix the boundary conditions for the parameterization such that a correspondence between the source and target surfaces is achieved, i.e. we identify corresponding boundary vertices and fix them at the same domain points. In practice, this is a single vertex in S and in U that constrains rotation for the unit circle domain, or four boundary vertices for the unit square domain.

Some applications require a more careful correspondence than what can be achieved from choosing boundary conditions. For example, the mapping between two faces (see Figure 7) should link relevant details like facial features (e.g., the brow wrinkles of the *Max Planck*). In this case the user provides a few additional (inner) point-to-point constraints which define a warp of the mean-



(a) (b) (c)

Figure 10: Mixing details using Laplacian coordinates. The Laplacian coordinates of surfaces in (a) and (b) are linearly blended in the middle to yield the shape in (c).



(a) (b)

Figure 11: Transplanting the *Armadillo*'s coating onto the *Bunny*'s back with a soft transition (a) and a sharp transition (b) between the two types of details. The size of the transition area in which the Laplacians are blended is large in (a) and small in (b).

value parameterization. In our implementation we used a radial basis function elastic warp; see e.g., [Lévy 2001; Praun et al. 2001; Kraevoy et al. 2003] for advanced constrained mapping techniques.

In general, a vertex $i \in U$ is mapped to some arbitrary point inside a triangle $\tau \in S$. We experimented with several methods of sampling the Laplacian for a vertex. The best results are obtained by first mapping the 1-ring of i onto S using the parameterization, and then computing the Laplacian from this mapped 1-ring. Note that this approach assumes a locally similar distortion in the mapping. This is usually the case for the coating transfer; we used the 1-ring sampling in all the respective examples. We obtain similar results by linear interpolation of the three Laplacian coordinates sampled at the vertices of the triangle τ. While this approach leads to some "blurring" compared to the first one, it is even simpler and does not suffer from extremely different parametric distortion. In addition, no special treatment is required at the boundary of the domain when it is chosen to be a disk.

To enable faithful resampling of the Laplacian coordinates, the tessellations of the surfaces U and S need to be "compatible", i.e. each mesh must be (locally) fine enough to accommodate the detail of the other mesh. This can be achieved by a local, isotropic remeshing (see e.g., [Alliez et al. 2003; Surazhsky and Gotsman 2003; Vorsatz et al. 2003]) of the sampled regions of U and S.

After the mapping between U and S has been established and the Laplacians have been sampled, the coating transfer proceeds as explained before. Note that now the corresponding ξ_i is the difference between the *sampled* Laplacian coordinates in S and $\tilde{S}$. See the examples in Figures 6, 7 and 9.

5.3 Mixing details

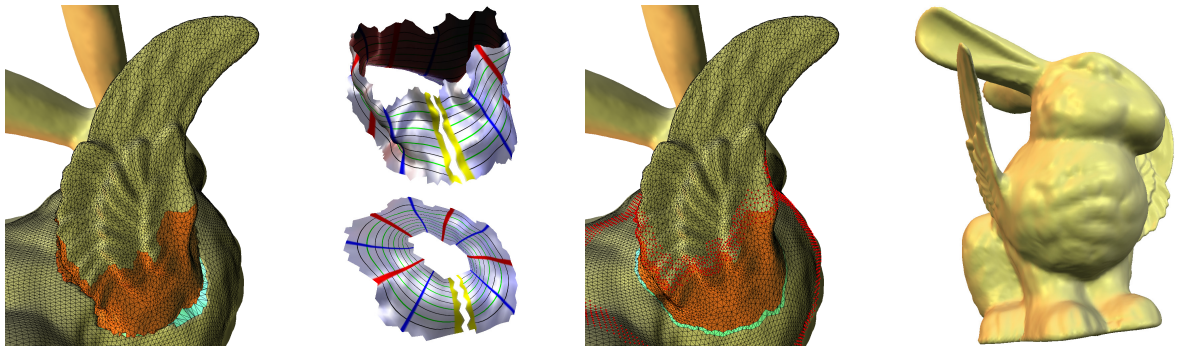
Given two meshes with different detail, a variant of the above transfer mechanism can be applied on a third target mesh from the two sources. Each vertex in the transitional region of the target mesh receives the linear interpolation of the corresponding Laplacian coordinates of the source meshes. Figure 10 illustrates the effect of mixing the details. This example emphasizes the gradual transition of geometric structure, as the details of the two source meshes differ in smoothness, form and orientation. Note that the the global shape of the target mesh is deformed respectively. By adding anchor points over the target, its shape can be further deformed. Figure 11 shows the application of this mechanism to transplant the *Armadillo*'s back onto the *Bunny*'s back with a soft transition. In the next section we further discuss this transplanting operation.

6 Transplanting surface patches

In the previous sections we showed how the Laplacian coordinates allow us to transfer the details of a surface onto another and how to gradually mix the details of two surfaces. These techniques are refined to allow a seamless transplanting of one shape onto another. The transplanting operation consists of two apparently independent classes of operations: topological and geometric. The topological operation creates one consistent triangulation from the connectivities of the two submeshes. The geometric operation creates a gradual change of the geometric structure of one shape into another. The latter operation is based on the Laplacian coordinates and the reconstruction mechanism.

Let S denote the mesh that is transplanted onto a surface U. See Figure 12, where the right wing (S) of the *Feline* is transplanted onto the *Bunny* (U). The transplanting requires the registration of the two parts in world coordinates. This defines the desired location and orientation of the transplanted shape, as well as its scale. If required, the meshes are locally remeshed in order to make the scale of the Laplacian coordinates compatible (cf. Section 5.2). The user selects a region $U_\circ$ of U onto which S will be transplanted. Hence the boundary of $U_\circ$ is assumed to be homeomorphic to the boundary of S. After cutting $U_\circ$ off U, the two boundary loops are trivially zipped. This creates the connectivity of the target mesh D (Figure 12(a)).

The remaining transplanting algorithm is similar to details mixing. The *transitional regions* for resampling, S' on S and U' on $U_\circ$, are selected, e.g., by offsetting from the respective cut seams. Since D includes a zipped "copy" of S, its transitional region D' is implicitly defined by S' along with a trivial mapping between vertices of the two regions. For sampling, we require a correspondence between the patches S' and U'. We parameterize both meshes over a common domain, e.g., the unit square. If the patches have to be cut to match the topology of the domain, the cuts are used to align the mappings for correspondence between the patches. In our experiments no further warping was necessary to improve the correspondence (cf. Section 5.2).



(a) (b) (c) (d)

Figure 12: Transplanting of *Feline*'s wings onto the *Bunny*. (a) After cutting the parts and fixing the desired pose, the zipping (in green) defines the target connectivity D. The transitional region D' is marked red. (b) D' is sampled over the respective regions $U' \subset U_\circ$ ($U_\circ$ is the cut part of the *Bunny*'s back) and S' (the bottom of the wing). The texture with *uv*-isolines visualizes the mapping over the unit square. The cut (in yellow) aligns the two maps. (c) The result of reconstruction. The reconstructed submesh is padded by a belt of anchors (red dots). Note the change of the zipping seam triangles (in green) and the gradual change and preservation details within the transition region (red). (d) The flying *Bunny* (see also Figure 1(d)).

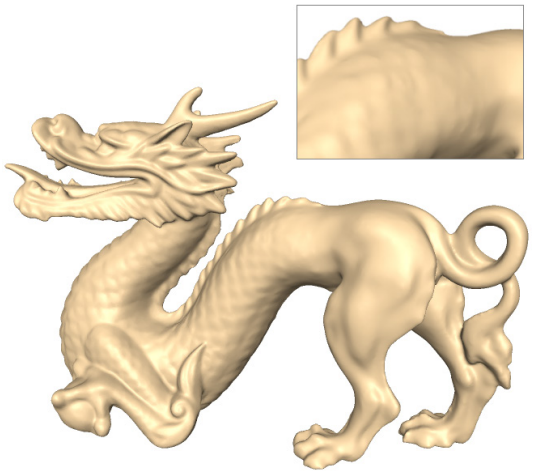


Figure 13: Transplanting part of the *Feline* onto the *Dragon*.

Once the transitional regions and the mappings are defined, the transplanting procedure is ready to sample the Laplacian coordinates over the target region D'. In order to mix details, the corresponding samples from S' and U' are linearly interpolated with weights defined by their relative position in the unit square parameter domain. More precisely, if $v \in [0,1]$ defines the coordinate along the "height" axis (the blue and red lines in Figure 12(b), then the weights are $(1-v)$ and v, respectively. Since the length distortion of the maps may significantly differ, we linearly interpolate the Laplacian coordinates for sampling (cf. Section 5.2). The transition region is padded with some additional free vertices and a belt of anchors (similar to the editing ROI). These vertices are supposed to stay in place as much as possible, and their Laplacian coordinates are sampled from D. The remaining vertices are fixed and hence not required for the reconstruction. Figures 12(c)-(d), 13 show the results.

7 Implementation details

All the techniques presented in this paper are implemented and tested on a 2.0 GHz Pentium 4 computer. The main computational core of the surface reconstruction algorithm is solving a sparse linear least-squares problem. We use a direct solver [Toledo 2003] which first computes a sparse triangular factorization of the normal equations (employing fill-reducing reordering) and then finds the minimizer by back-substitution. The system is well-conditioned thanks to the anchors. As mentioned in Section 4, constructing the matrix of the least-squares system and factorizing it takes the bulk of the computation time. This might seem a heavy operation for such an application as interactive mesh editing; however, it is done only once per ROI selection. Solving by back-substitution is quite fast and enables us to reconstruct the surface interactively, following the user's manipulations of the handle. It should be noted that the system is comprised of only the vertices that fall into the ROI; thus the complexity is not directly dependent on the size of the entire mesh, but rather on the size of the ROI. We experimented with various ROIs of sizes in the order of tens of thousands of vertices. The "intermediate preprocess" times observed were a few seconds, while the actual editing process runs at interactive frame rates. For example, the construction of the system matrix for a ROI on the arm of the *Octopus* (about 10K vertices) took 1.5 seconds and the factorization 1.9 seconds. The solve (when moving the handle) took 0.07 seconds. The construction of the normal equations can probably be further optimized. Some short editing sessions are demonstrated in the accompanying video.

8 Conclusions

We have developed an intrinsic geometry representation for meshes that fosters several local surface editing operations. Geometry is essentially encoded using differential properties of the surface, so that the local shape (or, surface detail) is preserved as much as possible given the constraints posed by the user. We show how to use this representation for interactive free-form deformations, detail transfer or mixing, and transplanting partial surface meshes.

It is interesting to compare the Laplacian-based approach to multi-resolution approaches: since each vertex is represented individually as a Laplacian coordinate, the user can freely choose the editing region and model arbitrary boundary constraints, however, computing absolute coordinates requires the solution of a linear system. On the other hand, the non-local bases in multi-resolution representations limit the choice of the editing region and the boundary constraints, but absolute coordinates are computed simpler, by summing displacements through the hierarchy.

Global modeling operations naturally require global surface representations. We would like to adapt our approach to implicit shapes, possibly to the level-set framework. When working with meshes, explicit handling of connectivity is required. In [Biermann et al. 2002], this problem is dealt with by using regular remeshing. Here, we tried to preserve the original connectivities as much as possible, modifying only the transition area. A similar approach is taken in [Lévy 2003]. However, the reconstruction of the transitional region respects only smoothness conditions, while in our case the transitional surface includes a gradual change of shape and details inherited from the original surfaces.

In general, modeling geometry should be coupled to modeling other surface properties, such as textures. The machinery of discrete Poisson equations has already shown to be effective for image editing, so that editing textured surfaces should possibly be performed on a combined differential geometry/texture representation.

Acknowledgments

The *Bunny*, *Dragon*, *Armadillo* and *Feline* models are courtesy of Stanford University and the *Octopus* is courtesy of Mark Pauly. 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) and the Israeli Ministry of Science.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

ALLIEZ, P., DE VERDIÈRE, É. C., DEVILLERS, O., AND ISENBURG, M. 2003. Isotropic surface remeshing. In *Proceedings of Shape Modeling International*, 49–58.

BENDELS, G. H., AND KLEIN, R. 2003. Mesh forging: editing of 3d-meshes using implicitly defined occluders. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 207–217.

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. In *Proceedings of ACM SIGGRAPH 2002*, 312–321.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, 41–47.

COQUILLART, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Proceedings of SIGGRAPH 90*, 187–196.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH 99*, 317–324.

FARIN, G. 1992. *Curves and surfaces for computer aided geometric design: a practical guide*. Academic Press.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. In *Proceedings of ACM SIGGRAPH 2002*, 249–256.

FLOATER, M. S. 2003. Mean-value coordinates. *Computer Aided Geometric Design 20*, 19–27.

FORSEY, D., AND BARTELS, R. 1988. Hierarchical b-spline refinement. In *Proceedings of ACM SIGGRAPH 88*, 205–212.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH 2000*, 249–254.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *Proceedings of ACM SIGGRAPH 99*, 325–334.

GUY, A., AND WYVIL, B. 1995. Controlled blending for implicit surfaces. In *Implicit Surfaces '95*.

HOSCHEK, J., AND LASSER, D. 1993. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters.

KANAI, T., SUZUKI, H., MITANI, J., AND KIMURA, F. 1999. Interactive mesh fusion based on local 3D metamorphosis. In *Graphics Interface '99*, 148–156.

KARNI, Z., AND GOTSMAN, C. 2000. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, 279–286.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH 98*, 105–114.

KOBBELT, L., VORSATZ, J., AND SEIDEL, H.-P. 1999. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry: Theory and Applications 14*, 5–24.

KRAEVOY, V., SHEFFER, A., AND GOTSMAN, C. 2003. Matchmaker: constructing constrained texture maps. In *Proceedings ACM SIGGRAPH 2003*, 326–333.

KURIYAMA, S., AND KANEKO, T. 1999. Discrete parametrization for deforming arbitrary meshes. In *Graphics Interface*, 132–139.

LE VEUVRE, L. 2003. Modelling and deformation of surfaces defined over finite elements. In *Proceedings of Shape Modeling International*, 175–183.

LÉVY, B. 2001. Constrained texture mapping for polygonal meshes. In *Proceedings ACM SIGGRAPH 2001*, 417–424.

LÉVY, B. 2003. Dual domain extrapolation. In *Proceedings of ACM SIGGRAPH 2003*, 364–369.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press. To appear.

MUSETH, K., BREEN, D. E., WHITAKER, R. T., AND BARR, A. H. 2002. Level set surface editing operators. In *Proceedings of ACM SIGGRAPH 2002*, 330–338.

PASKO, A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. 1995. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8, 429–446.

PAULY, M., KEISER, R., KOBBELT, L. P., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGGRAPH 2003*, 641–650.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *Proceedings of ACM SIGGRAPH 2003*, 313–318.

PRAUN, E., SWELDENS, W., AND SCHRDER, P. 2001. Consistent mesh parameterizations. In *Proceedings of ACM SIGGRAPH 2001*, 179–184.

RANTA, M., INUI, M., KIMURA, F., AND MÄNTYLÄ, M. 1993. Cut and paste based modeling with boundary features. In *SMA '93: Proceedings of the Second Symposium on Solid Modeling and Applications*, 303–312.

ROCKWOOD, A. P. 1989. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics 8*, 4 (Oct.), 279–297.

SCHRÖDER, P., AND ZORIN, D. 2000. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH 86*, 151–160.

SORKINE, O., COHEN-OR, D., AND TOLEDO, S. 2003. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 42–51.

SURAZHSKY, V., AND GOTSMAN, C. 2003. Explicit surface remeshing. In *Proceedings of EG Symposium on Geometry Processing*, 17–28.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH 95*, 351–358.

TOLEDO, S. 2003. *TAUCS: A Library of Sparse Linear Solvers, version 2.2*. Tel-Aviv University, Available online at *http://www.tau.ac.il/ stoledo/taucs/*, Sept.

VORSATZ, J., RÖSSL, C., AND SEIDEL, H.-P. 2003. Dynamic remeshing and applications. In *Proceedings Solid Modeling*, 167–175.

WELCH, W., AND WITKIN, A. 1994. Free–Form shape design using triangulated surfaces. In *Proceedings of ACM SIGGRAPH 94*, 247–256.

WYVILL, B., GUY, A., AND GALIN, E. 1999. Extending the CSG tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2 (June), 149–158.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. In *Proceedings of ACM SIGGRAPH 2004*. to appear.

ZORIN, D., SCHRDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proceedings of ACM SIGGRAPH 97*, 259–268.

A Exponential of a 3×3 skew-symmetric matrix

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

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

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

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

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.