Deformation Transfer for Detail-Preserving Surface Editing

Mario Botsch[1] Robert W. Sumner[2] Mark Pauly[2] Markus Gross[1]

[1]Computer Graphics Laboratory, ETH Zurich
[2]Applied Geometry Group, ETH Zurich

Abstract

Recent detail-preserving shape deformation techniques are either based on a combination of multiresolution decomposition and variational bending energy minimization, or they manipulate differential coordinates and solve a Poisson system to obtain the deformed surface. We establish an explicit connection between the two approaches and discuss their inherent limitations, such as local self-intersections for the former and translation insensitivity of the latter. Based on these new insights we combine both methods into a novel shape editing technique that does not suffer from previous limitations, while retaining editing flexibility and efficiency.

1 Introduction

Natural and intuitive shape deformations — like the ones we experience every day in real life — require a physically *plausible* simulation of the mechanical behavior of the surface. Physically *accurate* surface deformations can be obtained by minimizing stretching and bending energies of thin shells, measured as surface integrals of the change of first and second fundamental forms [20]. Since this nonlinear deformation energy is computationally demanding, it usually is simplified in the context of geometric shape editing systems with the goal of retaining as much physical realism as possible, while allowing for interactive editing performance.

A common approach is to first linearize the shell energy by replacing fundamental forms by partial derivatives, and then to apply variational calculus to derive a corresponding Euler-Lagrange PDE that characterizes the surface of minimal deformation energy [6]. Discretizing the resulting bi-Laplacian PDE using finite differences finally leads to a linear system whose solution is the deformed surface [9, 3]. Other conceptually similar methods [8, 4] fall into this category, but minimize slightly different energies. We refer to this class of variational minimization approaches as VARMIN.

However, intuitive detail preservation inherently requires some nonlinear computation to ensure that small-scale geometric details rotate in a natural way even when the user imposes purely translational changes on constrained vertices [5] (cf. Fig. 1). As a result, the VARMIN approach is complemented by a multiresolution decomposition, which splits the surface $\mathcal{S}$ into a smooth low-frequency base surface $\mathcal{B}$ and high-frequency geometric details $\mathcal{D} = \mathcal{S} \ominus \mathcal{B}$. The details typically are encoded as displacement vectors parallel to the normal field of $\mathcal{B}$ [10]. A deformation is then applied to the base surface, $\mathcal{B} \mapsto \mathcal{B}'$, and the modified fine-scale surface is reconstructed from $\mathcal{B}'$ and the original normal displacements as $\mathcal{S}' = \mathcal{B}' \oplus \mathcal{D}$.

One inherent limitation of this approach is that the difference between $\mathcal{S}$ and $\mathcal{B}$ must be sufficiently small, such that $\mathcal{S}$ can be represented as a height field over $\mathcal{B}$. If this criterion cannot be achieved for a simple two-level hierarchy, more levels $\mathcal{S} = \mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k = \mathcal{B}$ are necessary.

Moreover, since neighboring displacement vectors are not coupled, the resulting displaced surface may have local self-intersections whenever the curvature of the deformed base surface $\mathcal{B}'$ is too high in relation to the displacement length. Displacement volumes [2] avoid these local self-intersections by representing geometric details $\mathcal{D}$ by prism elements of constant volume, but in turn require an expensive non-linear detail reconstruction.

Motivated by these limitations, many recent deformation approaches, which we denote DIFFCOORD, are based on the modification of differential coordinates and do not require a multiresolution hierarchy [11, 17, 22, 23, 12]. Instead of explicitly changing spatial coordinates, these methods modify gradients or Laplacians of the surface and solve a Poisson PDE to find the deformed sur-

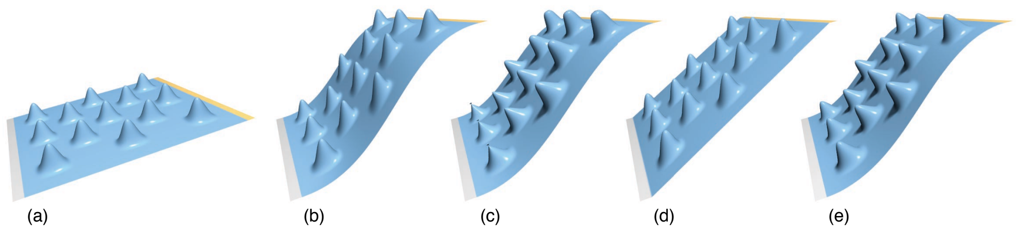

Figure 1: The right side of a bumpy plane (a) is lifted. Without a multiresolution decomposition geometric details are not rotated, yielding an unrealistic deformation (b). A multiresolution deformation using normal displacements rotates details, but also distorts their shapes (c). Due to translation-insensitivity gradient-based techniques fail for this kind of deformation (d). Our new method avoids these problems (e).

face which complies with the deformed gradients or Laplacians, respectively.

These approaches also require a non-linear component in order to successfully preserve local surface details. In this case, the non-linear step is the way in which the differential coordinates are modified. A typical method uses the gradient of the desired deformation, i.e., its rotation and scale/shear components, to update the surface gradient field [22, 23]. However, since a translation does not cause a change in surface gradients, these techniques fail to yield intuitive results for translational deformations (cf. Fig. 1d). This *translation-insensitivity* is an inherent limitation of most approaches based on differential coordinates [5].

Another related technique is *deformation transfer* (DEFTRANS), which employs deformation gradients in order to transfer the deformation of a source mesh $\mathcal{S} \mapsto \mathcal{S}'$ to a target mesh $\mathcal{T}$, leading to a similar deformation $\mathcal{T} \mapsto \mathcal{T}'$ [18]. This method is applied in the MESHIK editing system [19], which allows complex shape deformations based on a set of example poses.

In this paper, we provide a detailed description of both gradient-based mesh editing (Section 2) and deformation transfer (Section 3). Although these methods were previously thought to be disparate, we derive an equivalence between them for volumetric tetrahedral meshes in Section 4. Based on this insight, we develop a surface-based formulation of deformation transfer that is equivalent to surface-based gradient editing and improves the performance of deformation transfer compared to its original formulation (Section 5). Since deformation transfer is a basic component of MeshIK [19], the latter method is also improved by our new formulation.

Building on our analysis of previous methods, we propose a new deformation technique that improves both the VARMIN and DIFFCOORD approaches, since it avoids local self-intersections and does not suffer from translation-insensitivity (cf. Fig. 1e). This method is surface-based and designed for interactive editing, since the most expensive computation involves solving sparse linear systems that can be pre-factored for computational efficiency.

2 Gradient-Based Deformation

Instead of deforming the spatial coordinates of a surface mesh, gradient-based editing [22] manipulates the mesh gradient field and derives the surface matching the deformed gradient field by solving a linear Poisson system.

Consider a triangle mesh $\mathcal{M}$ with n vertices $\{v_1, \ldots, v_n\}$ and m triangles $\{t_1, \ldots, t_m\}$, and its piecewise linear coordinate function $\mathbf{p}(\cdot)$ defined by barycentric interpolation of vertex coordinates $\mathbf{p}_i$:

$$\mathbf{p}(x) = \sum_{i=1}^{n} \phi_i(x) \cdot \mathbf{p}_i \,.$$

Here, $\phi_i(\cdot)$ are the piecewise linear "hat" basis functions associated with the vertices, i.e., $\phi_i(v_k) = \delta_{ik}$. The gradient of $\mathbf{p}(\cdot)$ is

$$\nabla \mathbf{p}(x) = \sum_{i=1}^{n} \nabla \phi_i(x)\ \mathbf{p}_i^T \tag{1}$$

and yields a constant 3×3 matrix $\mathbf{G}_j$ for each triangle t_j. The discrete divergence of this piecewise constant gradient field yields a discrete Laplace operator [21]:

$$\begin{aligned}\Delta\mathbf{p}(v_i) &= (\mathrm{div}\nabla\mathbf{p})(v_i)\\ &= \textstyle\sum_{t_j\in\mathcal{T}_i}\mathrm{area}(t_j)\left(\nabla\phi_i|_{t_j}\right)^T\mathbf{G}_j\end{aligned} \tag{2}$$

where $\mathcal{T}_i$ denotes the triangles incident to v_i.

The computation of the per-face gradients $\mathbf{G}_j$ can be written with a $3m\times n$ matrix $\mathbf{G}$ representing the gradient operator on the mesh $\mathcal{M}$

$$\begin{pmatrix}\mathbf{G}_1\\ \vdots\\ \mathbf{G}_m\end{pmatrix} = \mathbf{G}\begin{pmatrix}\mathbf{p}_1^T\\ \vdots\\ \mathbf{p}_n^T\end{pmatrix}. \tag{3}$$

The original gradient field is then manipulated by applying a local rotation/scaling matrix $\mathbf{S}_j$ to each gradient $\mathbf{G}_j$, which yields $\mathbf{G}'_j := \mathbf{G}_j\mathbf{S}_j^T$. Finally, computing new vertex positions $\mathbf{p}'_i$, such that the resulting mesh complies with the new gradients $\mathbf{G}'_j$, leads to a weighted least-squares problem:

$$\underbrace{\mathbf{G}^T\mathbf{D}\,\mathbf{G}}_{\Delta}\begin{pmatrix}\mathbf{p}'^{\,T}_1\\ \vdots\\ \mathbf{p}'^{\,T}_n\end{pmatrix} = \underbrace{\mathbf{G}^T\mathbf{D}}_{\mathrm{div}}\begin{pmatrix}\mathbf{G}'_1\\ \vdots\\ \mathbf{G}'_m\end{pmatrix}, \tag{4}$$

where $\mathbf{D}$ is a diagonal weighting matrix containing the triangle areas. In this equation, $\mathbf{G}^T\mathbf{D}$ is the matrix formulation of the divergence operator of (2), and therefore $\mathbf{G}^T\mathbf{D}\mathbf{G}$ is the divergence of the gradient, i.e., the discrete Laplace operator. Hence, the weighted least-squares problem (4) is a simple Poisson equation.

Notice that this derivation is valid both for two-manifold triangle meshes and volumetric solids represented by tetrahedra. In the latter case, t_j simply denotes tetrahedra instead of triangles and triangle area is replaced by tetrahedron volume. In the former case, (2) leads to the standard Laplace discretization for triangle meshes [14, 13]

$$\Delta\mathbf{p}_i = \sum_{v_k\in\mathcal{V}_i}\frac{1}{2}\left(\cot\alpha_{ik}+\cot\beta_{ik}\right)(\mathbf{p}_i-\mathbf{p}_k)\ , \tag{5}$$

where $\mathcal{V}_i$ denotes the one-ring neighbors of v_i, and α_{ik} and β_{ik} are the two angles opposite to the edge (v_i, v_k). Hence, the matrix $\mathbf{G}^T\,\mathbf{D}\,\mathbf{G}$ can also be built directly from the cotangent weights in (5).

3 Deformation Transfer

Sumner and colleagues presented a framework for transferring deformations from one triangle mesh to another [18], and extended it for their mesh-based inverse kinematics technique [19].

For a source mesh given in an original state $\mathcal{S}$ and deformed state $\mathcal{S}'$, an affine *source deformation* $\mathbf{q}\mapsto\mathbf{S}_j\mathbf{q}+\mathbf{t}_j$ is derived for each triangle $t_j\in\mathcal{S}$. Its deformation gradient $\mathbf{S}_j\in\mathbb{R}^{3\times3}$ consists of the rotational, stretch, and shear parts of the deformation. Given a target mesh $\mathcal{T}$, the goal is to find new vertex positions $\mathbf{p}'_i$ for $v_i\in\mathcal{T}$ such that the triangles' deformation gradients $\mathbf{T}_j$ for $\mathcal{T}$ match the given source deformation gradients $\mathbf{S}_j$ for $\mathcal{S}$.

Since a triangle with original vertex positions $(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)$ and (unknown) deformed positions $(\mathbf{p}'_1,\mathbf{p}'_2,\mathbf{p}'_3)$ does not fully define a linear mapping $\mathbf{T}$, Sumner et al. [18] add a fourth point $\mathbf{p}_4$ (respectively $\mathbf{p}'_4$), thereby turning the triangle into a tetrahedron. This extension yields

$$\begin{aligned}\mathbf{T} &= \left(\mathbf{p}'_1-\mathbf{p}'_4,\mathbf{p}'_2-\mathbf{p}'_4,\mathbf{p}'_3-\mathbf{p}'_4\right)\cdot\\ &\qquad\underbrace{(\mathbf{p}_1-\mathbf{p}_4,\mathbf{p}_2-\mathbf{p}_4,\mathbf{p}_3-\mathbf{p}_4)^{-1}}_{=:\mathbf{V}^{-1}=(\mathbf{a},\mathbf{b},\mathbf{c})^T}\ ,\end{aligned} \tag{6}$$

which maps the initial tetrahedron's orientation to the new configuration, i.e.,

$$\mathbf{T}(\mathbf{p}_i-\mathbf{p}_4) = \left(\mathbf{p}'_i-\mathbf{p}'_4\right)\ ,\quad i\in\{1,2,3\}\ . \tag{7}$$

In this formulation, the deformation gradient $\mathbf{T}$ is linear in the positions $\mathbf{p}'_i$. Letting $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ denote the rows of $\mathbf{V}^{-1}$ gives

$$\mathbf{T}^T = (\mathbf{a},\mathbf{b},\mathbf{c},-\mathbf{a}-\mathbf{b}-\mathbf{c})\cdot\left(\mathbf{p}'_1,\mathbf{p}'_2,\mathbf{p}'_3,\mathbf{p}'_4\right)^T\ . \tag{8}$$

Let us denote with $\tilde{n}$ the number of vertices n plus the additional fourth vertices for each triangle, i.e., $\tilde{n} = n+m\approx 3n$. Then a global $3m\times\tilde{n}$ matrix $\tilde{\mathbf{G}}$ can be composed from equations (8) for each triangle, such that the m deformation gradients are computed by

$$\begin{pmatrix}\mathbf{T}_1^T\\ \vdots\\ \mathbf{T}_m^T\end{pmatrix} = \tilde{\mathbf{G}}\begin{pmatrix}\mathbf{p}'^{\,T}_1\\ \vdots\\ \mathbf{p}'^{\,T}_{\tilde{n}}\end{pmatrix}. \tag{9}$$

Finally, the new vertex positions $\mathbf{p}'_i$ satisfying the given per-triangle deformations $\mathbf{S}_j$ in the least squares sense are given by

$$\tilde{\mathbf{G}}^T\,\tilde{\mathbf{G}}\begin{pmatrix}\mathbf{p}'^{\,T}_1\\ \vdots\\ \mathbf{p}'^{\,T}_{\tilde{n}}\end{pmatrix} = \tilde{\mathbf{G}}^T\begin{pmatrix}\mathbf{S}_1^T\\ \vdots\\ \mathbf{S}_m^T\end{pmatrix}. \tag{10}$$

4 Equivalence for Tetrahedral Meshes

In this section we show that the matrix $\tilde{\mathbf{G}}$ of (9) and (10) actually corresponds to the matrix $\mathbf{G}$ for the gradient operator of tetrahedral meshes in (3).

Consider a tetrahedron with vertices $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$. In order to compute the gradient of a linear function within the tetrahedron, we need the basis functions' gradients $(\nabla\phi_1, \nabla\phi_2, \nabla\phi_3, \nabla\phi_4)$ as defined in (1). The gradient $\nabla\phi_i$ must be perpendicular to the opposite face $(\mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l)$ of the tetrahedron, and its length has to be $1/h$ with h being the height of $\mathbf{p}_i$ above the opposite face. The first condition requires $\nabla\phi_i^T(\mathbf{p}_j - \mathbf{p}_l) = 0$ and $\nabla\phi_i^T(\mathbf{p}_k - \mathbf{p}_l) = 0$, and the latter condition simplifies to (with $\mathbf{p}_{ij} := \mathbf{p}_i - \mathbf{p}_j$)

$$\begin{aligned} \nabla\phi_i^T \mathbf{p}_{ij} &= \cos\left(\angle\left(\nabla\phi_i^T, \mathbf{p}_{ij}\right)\right) \|\mathbf{p}_{ij}\| \left\|\nabla\phi_i^T\right\| \\ &= h \cdot \tfrac{1}{h} = 1. \end{aligned}$$

These three conditions fully define the gradient $\nabla\phi_i$. With analogous conditions for the other gradients, we get the linear system

$$\begin{pmatrix} (\mathbf{p}_1 - \mathbf{p}_4)^T \\ (\mathbf{p}_2 - \mathbf{p}_4)^T \\ (\mathbf{p}_3 - \mathbf{p}_4)^T \end{pmatrix} (\nabla\phi_1, \ldots, \nabla\phi_4) = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}. \tag{11}$$

Notice that the matrix on the left hand side of (11) is the transpose of $\mathbf{V}$ from (6). As a consequence, computing the gradients $\nabla\phi_i$ as (combinations of) columns of $\mathbf{V}^{-T}$ as in (11) is equivalent to choosing rows of $\mathbf{V}^{-1}$ as in (8):

$$\begin{aligned} (\nabla\phi_1, \ldots, \nabla\phi_4) &= \mathbf{V}^{-T} \cdot \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \\ &= (\mathbf{a}, \mathbf{b}, \mathbf{c}, -\mathbf{a} - \mathbf{b} - \mathbf{c}). \end{aligned}$$

With this (8) becomes

$$\mathbf{T}^T = (\nabla\phi_1, \ldots, \nabla\phi_4) \cdot \left(\mathbf{p}'_1, \mathbf{p}'_2, \mathbf{p}'_3, \mathbf{p}'_4\right)^T, \tag{12}$$

such that the (transposed) deformation gradient $\mathbf{T}$ equals the gradient of the (deformed) coordinate function $\mathbf{p}'(\cdot)$ within that tetrahedron.

Since the matrix $\tilde{\mathbf{G}}$ is composed from the per-triangle equations (8) or (12), respectively, it equals the matrix $\mathbf{G}$ from (4) in the case of a volumetric tetrahedral mesh. Including the weighting matrix $\mathbf{D}$ of (4) to (10) thus turns the deformation transfer formulation into a volumetric Poisson problem, with tetrahedra constructed by adding a fourth point to each triangle of the mesh.

5 Equivalence for Triangle Meshes

The motivation for adding a fourth vertex to each triangle is to get an expression for deformation gradients $\mathbf{T}_j$ that is linear in the unknown vertex positions $\mathbf{p}'_i$. Even though these additional points are unknowns in the linear system (10), they are discarded after solving this system. In this section, we show that the additional points are not required for solving the deformation transfer problem, which thus decreases the size of the linear system (10) from $\tilde{n} \times \tilde{n}$ to $n \times n$, i.e., by a factor of 3.

As shown in the last section, the "volumetric" formulation of deformation transfer is equivalent to a volumetric Poisson problem. Hence, a natural surface-based formulation should be equivalent to the surface-based Poisson approach as described in Section 2. *Surface-based* deformation gradients $\mathbf{T}_j$ as in (8) or (12) should therefore equal the per-triangle gradients $\mathbf{G}'_j$ of (4) for the piecewise linear coordinate function $\mathbf{p}'(\cdot)$ of the mesh.

Similar to Section 4, the basis functions' gradients $\nabla\phi_i$ within a triangle $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ have to be perpendicular to the opposite edge $(\mathbf{p}_j, \mathbf{p}_k)$ and their length must be one over the distance to this edge. In addition to these two constraints, the gradients have to lie within the triangle and thus must be perpendicular to the triangle's normal $\mathbf{n}$, leading to the linear system

$$\begin{pmatrix} (\mathbf{p}_1 - \mathbf{p}_3)^T \\ (\mathbf{p}_2 - \mathbf{p}_3)^T \\ \mathbf{n}^T \end{pmatrix} (\nabla\phi_1, \ldots, \nabla\phi_3) = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Using this system to compute $\nabla\phi_i$ from the original points then turns (12) into

$$\mathbf{T}^T = (\nabla\phi_1, \nabla\phi_2, \nabla\phi_3) \cdot \left(\mathbf{p}'_1, \mathbf{p}'_2, \mathbf{p}'_3\right)^T, \tag{13}$$

which is still linear in the unknown positions $\mathbf{p}'_i$, but does not require an additional point per triangle. Similar to (7), this surface-based deformation gradient transforms the tangent directions, but it simply discards the normal component:

$$\begin{aligned} \mathbf{T}(\mathbf{p}_i - \mathbf{p}_2) &= \left(\mathbf{p}'_i - \mathbf{p}'_2\right) \quad i \in \{1, 2\} \\ \mathbf{Tn} &= \mathbf{0}. \end{aligned}$$

Composing the matrix $\tilde{\mathbf{G}}$ of (10) from the surface-based deformation gradients (13) then leads to the standard gradient matrix $\mathbf{G}$ of (3), and by including the diagonal weighting matrix $\mathbf{D}$ of triangle areas, the deformation transfer problem (10)

		Volumetric		Surface	
Example	Triangles	Factorization	Backsubst.	Factorization	Backsubst.
Thorn	87,038	19.001	0.228	3.282	0.161
Bumps	80,000	9.718	0.172	3.510	0.135
Cylinder	9,900	0.437	0.021	0.151	0.015

Table 1: Timing statistics for the two versions of deformation transfer. The volumetric approach with an added fourth vertex per triangle is compared with the surface-based deformation transfer with no extra vertex. All timing data is measured in seconds.

turns into a standard Poisson problem (4) for triangle meshes:

$$\underbrace{\mathbf{G}^T\mathbf{D}\,\mathbf{G}}_{\Delta}\begin{pmatrix}\mathbf{p}_1'^{\,T}\\ \vdots\\ \mathbf{p}_n'^{\,T}\end{pmatrix} = \underbrace{\mathbf{G}^T\mathbf{D}}_{\text{div}}\begin{pmatrix}\mathbf{S}_1^T\\ \vdots\\ \mathbf{S}_m^T\end{pmatrix} \qquad (14)$$

However, while the matrices of (14) and (4) are identical, the right-hand sides differ: Deformation transfer uses div $\left(\mathbf{S}_j^T\right)$, whereas gradient-based approaches use div $\left(\mathbf{G}_j\mathbf{S}_j^T\right)$. Notice that $\mathbf{p}\left(\cdot\right)$ corresponds to the identity on the original mesh, such that its gradients $\mathbf{G}_j$ are the identity within t_j's (tangent) plane ($\mathbf{G}_j\mathbf{t} = \mathbf{t}$), but discard the normal component ($\mathbf{G}_j\mathbf{n} = \mathbf{0}$). Hence, $\mathbf{G}_j\mathbf{S}_j^T$ and $\mathbf{S}_j^T$ differ only in the normal component, which corresponds to the orthogonal complement of the image of the gradient matrix $\mathbf{G}$ of (3). As a consequence, both right-hand sides lead to the same least squares results [7].

Notice that only the *target* deformation gradients $\mathbf{T}_j$ are computed using the formulation (13), since for them a linear expression in the $\mathbf{p}_i'$ is required. The *source* deformation gradients $\mathbf{S}_j$, are either computed using the fourth points as in (6), or by employing normal vectors. If we denote by $\mathbf{q}_i$, $\mathbf{q}_i'$, and $\mathbf{n}$, $\mathbf{n}'$ the vertex positions and normal of a triangle t_j in $\mathcal{S}$ or $\mathcal{S}'$, respectively, the source deformation gradient can be computed as

$$\begin{aligned}\mathbf{S}_j \;&=\; \left(\mathbf{q}_1' - \mathbf{q}_3', \mathbf{q}_2' - \mathbf{q}_3', \mathbf{n}'\right)\cdot \\ &\qquad \left(\mathbf{q}_1 - \mathbf{q}_3, \mathbf{q}_2 - \mathbf{q}_3, \mathbf{n}\right)^{-1}\;.\end{aligned} \qquad (15)$$

Compared to the original approach [18], our new surface-based formulation reduces the dimension of the matrix $\mathbf{G}^T\mathbf{D}\,\mathbf{G}$ from $\tilde{n}$ to n, and the average number of non-zeros in this sparse matrix from $23n$ to $7n$. This reduction by a factor of 3 allows our method to handle larger models and, since the computational complexity of sparse solvers is linear in the number of non-zeros [1], it also improves the performance of deformation transfer (see Table 1).

6 Deformation Transfer for Detail-Preserving Mesh Editing

As discussed in Section 1, VARMIN and DIFFCOORD suffer from inherent limitations, such as local self-intersections for VARMIN and translation insensitivity for DIFFCOORD. In this section we use the new insight gained from the equivalence between deformation transfer and gradient-based editing to derive a new method that overcomes these limitations.

We propose to improve the VARMIN approach by replacing the normal displacement step by a deformation transfer process. After changing the base surface from $\mathcal{B}$ to $\mathcal{B}'$ we compute the corresponding deformation gradients $\mathbf{S}_j$ for each triangle $t_j \in \mathcal{B}$. Using the surface-based formulation of Section 5, we transfer the deformation $\mathcal{B} \mapsto \mathcal{B}'$ to the fine-scale surface $\mathcal{S}$, which yields the desired result $\mathcal{S}'$. The following paragraphs and Fig. 2 describe our multiresolution framework step by step.

Base Surface Computation Implementing the detail reconstruction operator as a deformation transfer process removes the restriction for $\mathcal{S}$ to be a height field over $\mathcal{B}$, therefore allowing simple two-level hierarchies even for complex geometries (cf. Fig. 3). For instance, we can compute the base surface $\mathcal{B}$ by removing *all* high frequencies from the user-defined deformable region. This can be achieved by minimizing the thin-plate energy, which only requires solving a bi-Laplacian system $\Delta^2(\mathbf{q}_i) = \mathbf{0}$ with boundary constraints [3].

Base Surface Deformation A base surface deformation $\mathcal{B} \mapsto \mathcal{B}'$ of minimal (linearized) bending energy can be computed by solving the bi-Laplacian system $\Delta^2(\delta\mathbf{q}_i) = \mathbf{0}$ for a smooth displacement field $\delta\mathbf{q}_i := \mathbf{q}_i' - \mathbf{q}_i$. However, any other technique for deforming $\mathcal{B}$ to $\mathcal{B}'$ can be applied as well.

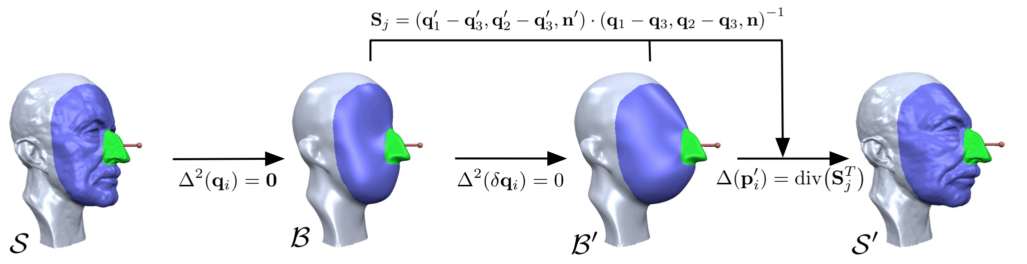
$\mathbf{S}_j = (\mathbf{q}_1' - \mathbf{q}_3', \mathbf{q}_2' - \mathbf{q}_3', \mathbf{n}') \cdot (\mathbf{q}_1 - \mathbf{q}_3, \mathbf{q}_2 - \mathbf{q}_3, \mathbf{n})^{-1}$
$\Delta^2(\mathbf{q}_i) = \mathbf{0}$
$\Delta^2(\delta\mathbf{q}_i) = 0$
$\Delta(\mathbf{p}_i') = \mathrm{div}(\mathbf{S}_j^T)$
$\mathcal{S}$
$\mathcal{B}$
$\mathcal{B}'$
$\mathcal{S}'$

Figure 2: Schematic of our multiresolution deformation pipeline based on deformation transfer.

Deformation Transfer From $\mathcal{B}$ and $\mathcal{B}'$ we extract the source deformation gradients $\mathbf{S}_j$ as in (15). Depending on the application, only the rotational component of $\mathbf{S}_j$ can be kept by discarding scaling and shearing components based on a polar decomposition [16], which then better preserves surface area. To transfer the deformation $\mathcal{B} \mapsto \mathcal{B}'$ to the fine-scale surface $\mathcal{S}$ we solve the Poisson system $\Delta(\mathbf{p}_i') = \mathrm{div}(\mathbf{S}_j^T)$ as in (14).

Since the Laplacian matrix for the deformation transfer as well as the bi-Laplacian matrix for the base surface deformation only depend on the initial vertex positions $\mathbf{p}_i \in \mathcal{S}$ and $\mathbf{q}_i \in \mathcal{B}$, and hence do not change during deformation, we can prefactor these matrices and perform efficient back-substitutions in each frame [1]. Even higher efficiency can be achieved by employing the precomputed basis functions of [3] for the deformation of the base surface.

Employing deformation transfer as a multiresolution representation not only enables a simple two-level decomposition, it moreover avoids local self-intersections much better than normal displacements. Local self-intersections would cause high *local* errors in the resulting gradient field, which are highly unlikely, since the least squares solution of (14) distributes the errors evenly over the whole mesh. Fig. 4 shows a comparison with normal displacements and the non-linear displacement volumes of [2]. Note that while displacement volumes better preserve global volume, computation time is orders of magnitude higher than for our method. Fig. 5 illustrates how normal displacements distort local details, whereas our technique leads to more intuitive results.

Because deformation transfer is equivalent to gradient-based Poisson editing, our new approach can also be considered as an improvement of the latter. Instead of heuristically propagating user-specified deformation gradients (rotations and scalings) over the mesh — which was shown in [5] to fail for translations — now the change of the base surface $\mathcal{B}$ is used to infer per-triangle rotations.

Since the base surface deformation $\mathcal{B} \mapsto \mathcal{B}'$ is derived by minimizing global bending energies, an optimally smooth deformation is used to derive the deformation gradients. In comparison to the heuristic propagation of local rotations this yields a higher deformation quality. Moreover, while rotations are handled equally well in both methods (cf. Fig. 5), the main benefit of our approach over DIFFCOORD is the more intuitive surface behavior under translational deformations (cf. Fig. 1e). Similar to the Poisson methods, our detail transfer technique requires the solution of the same kind of linear system, i.e., both approaches are roughly equivalent in terms of computational complexity.

Compared to recent non-linear approaches [15, 5], our method is less suited for large-scale deformations. Due to our linear formulation, large deformations need to be split up into a sequence of smaller ones to obtain physically plausible results. However, this drawback is partly compensated by faster computations as compared to [5].

7 Conclusion

We have presented a thorough analysis of gradient-based deformations and deformation transfer, showing their equivalence for both surface and volume meshes. This insight leads to performance improvements for methods based on deformation transfer. In addition it allows the formulation of a new mesh editing method that combines the advantages of differential coordinates and explicit multiresolution decomposition.

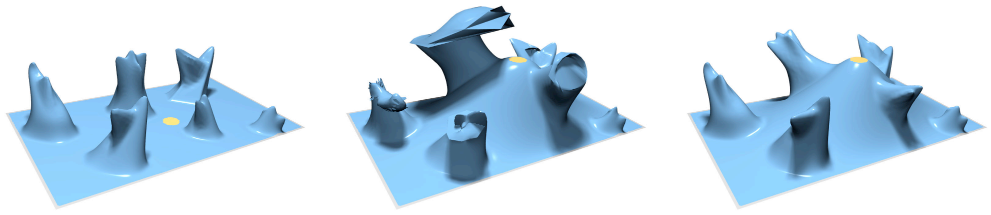

Figure 3: This thorny model (left) cannot be represented as a height field over the base surface, which is why the deformation based on normal displacements fails (center). Our new approach employs deformation transfer for the multiresolution reconstruction and therefore overcomes this limitation (right).

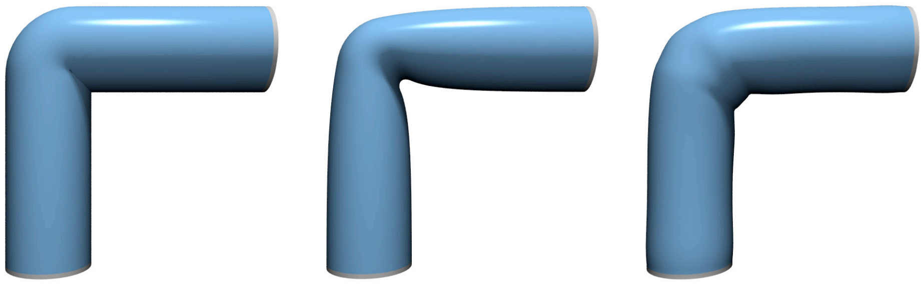

Figure 4: A cylinder is bent by explicitly deforming its base surface, which is a cylinder of half the radius. Normal displacements immediately lead to local self-intersections (left), whereas our new technique effectively avoids them (center). Compared to the non-linear displacement volumes [2] (right) our linear method does not (try to) preserve volume.

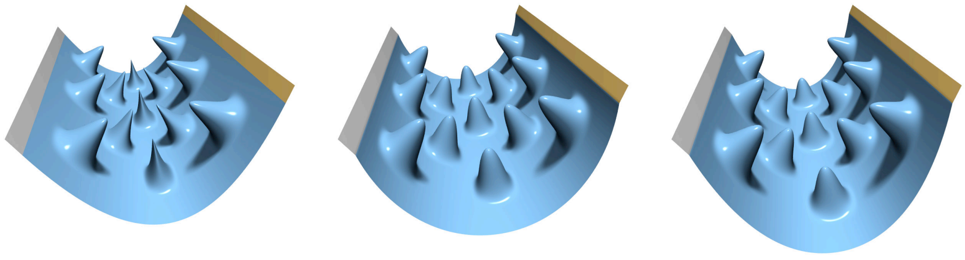

Figure 5: A concave bending of the bumpy plane of Fig. 1. Normal displacements unnaturally change the local details (left), whereas gradient-based editing (center) and our technique (right) preserve them.

References

[1] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for geometry processing. In *11th IMA conference on the Mathematics of Surfaces*, 2005.

[2] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. In *Proc. of Eurographics 03*, pages 483–491, 2003.

[3] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. In *Proc. of ACM SIGGRAPH 04*, pages 630–634, 2004.

[4] M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. In *Proc. of Eurographics 05*, pages 611–621, 2005.

[5] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Eurographics Symposium on Geometry Processing 06*, pages 11–20, 2006.

[6] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *Proc. of ACM SIGGRAPH 91*, pages 257–266, 1991.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.

[8] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proc. of ACM SIGGRAPH 99*, pages 325–334, 1999.

[9] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proc. of ACM SIGGRAPH 98*, pages 105–114, 1998.

[10] L. Kobbelt, J. Vorsatz, and H.-P. Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Comput. Geom. Theory Appl.*, 14(1-3):5–24, 1999.

[11] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proc. of Shape Modeling International 04*, pages 181–190, 2004.

[12] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proc. of ACM SIGGRAPH 05*, pages 479–487, 2005.

[13] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[14] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[15] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proc. of Symp. on 3D Data Processing, Visualization and Transmission (3DPVT) '04*, pages 68–75, 2004.

[16] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Proc. of Graphics Interface 92*, pages 258–264, 1992.

[17] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proc. of Eurographics symposium on Geometry Processing 04*, pages 179–188, 2004.

[18] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *Proc. of ACM SIGGRAPH 04*, pages 399–405, 2004.

[19] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. In *Proc. of ACM SIGGRAPH 05*, pages 488–495, 2005.

[20] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of ACM SIGGRAPH 87*, pages 205–214, 1987.

[21] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. In *Proc. of ACM SIGGRAPH 03*, 2003.

[22] 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 *Proc. of ACM SIGGRAPH 04*, pages 644–651, 2004.

[23] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. In *Proc. of Eurographics 05*, pages 601–609, 2005.