On Linear Variational Surface Deformation Methods

Mario Botsch and Olga Sorkine

Abstract—This survey reviews the recent advances in linear variational mesh deformation techniques. These methods were developed for editing detailed high-resolution meshes like those produced by scanning real-world objects. The challenge of manipulating such complex surfaces is threefold: The deformation technique has to be sufficiently fast, robust, intuitive, and easy to control to be useful for interactive applications. An intuitive and, thus, predictable deformation tool should provide physically plausible and aesthetically pleasing surface deformations, which, in particular, requires its geometric details to be preserved. The methods that we survey generally formulate surface deformation as a global variational optimization problem that addresses the differential properties of the edited surface. Efficiency and robustness are achieved by linearizing the underlying objective functional such that the global optimization amounts to solving a sparse linear system of equations. We review the different deformation energies and detail preservation techniques that were proposed in recent years, together with the various techniques to rectify the linearization artifacts. Our goal is to provide the reader with a systematic classification and comparative description of the different techniques, revealing the strengths and weaknesses of each approach in common editing scenarios.

Index Terms—Mesh editing, linear optimization, discrete differential operators.

1 Introduction

This paper presents the recent advances in mesh deformation and editing techniques. Shape deformation methods have been an active area of research in geometric modeling due to their ever widening range of applications in industrial and artistic design. Surfaces originating from 3D scans of real-world objects have become commonly affordable, which, in turn, requires the development of new tools to deal with such kind of surfaces: They are usually densely sampled and not smooth, in the sense that they contain abundant geometric detail at various scales. More traditional surface editing machinery developed, for example, for parametric surfaces or subdivision surfaces, is difficult to apply to such data; therefore, various deformation techniques have evolved that work directly with the irregular triangle mesh representation.

This survey focuses on linear surface-based algorithms for mesh deformation. We address surface-based techniques as opposed to space deformations or free-form deformations since, currently, there is no comprehensive survey that reviews the former, whereas space deformations are well exposed in the literature (for example, see [6] and [46]). A concise summary of Laplacian-based mesh processing techniques appeared in [56]. In this survey, we expand all recently proposed differential surface representations linked with linear mesh editing techniques and describe them comparatively. By "linear," we mean that the main ingredient of the deformation algorithm is a global quadratic variational minimization problem whose solution, given certain modeling constraints derived from user interaction, is the desired modified surface. The variational minimization of a quadratic functional is achieved by solving a linear system of equations; hence, we call such methods linear.

Linear methods are attractive for several reasons. First, they are fast, especially when the associated linear system is sparse, as is the case with all the techniques that we shall discuss. The availability of highly optimized sparse linear solvers makes linear techniques very efficient and, at the same time, simple to implement (basically, one only needs to set up the required linear system and then let the solver library do the rest). In addition, linear methods are robust: When appropriate boundary conditions are used, the quadratic energy has a unique global minimum. Moreover, most methods are formulated in such a way that the resulting deformed surface is a smooth function of the modeling constraints; thus, a slight perturbation of the constraints changes the resulting surface only a little.

The advantages of linear deformation methods, however, come with a price: As we shall see, in the most intuitive setting, the surface deformation problem is inherently nonlinear because it requires deducing local rotations of the surface based on position displacements. Therefore, a linear method can only provide an approximate result, or a compromise must be made in terms of the problem setup, for example, requiring more complex interactive input from the user. These trade-offs have spawned a large variety of deformation techniques, each one attacking the problem from a different angle. The impressive amount of literature on the subject that appeared in the past few years may confuse and overwhelm the casual reader. This survey is aimed at clarifying the various available methods by a systematic description of their assumptions on the problem setup and the underlying deformation mechanisms. Our

• *M. Botsch is with the Computer Graphis Laboratory, ETH Zurich, IFW D 25.1, Haldeneggsteig 4, 8092 Zurich, Switzerland. E-mail: botsch@inf.ethz.ch.*
• *O. Sorkine is with the Computer Graphics Group, Technische Universität Berlin, Sekretariat EN 7-1, Fakultät Elektrotechnik & Informatik Einsteinufer 17, 10587 Berlin, Germany. E-mail: sorkine@cs.tu-berlin.de.*

Manuscript received 29 Nov. 2006; accepted 13 Mar. 2007; published online 3 Apr. 2007.
Recommended for acceptance by H.-P. Seidel.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0214-1106. Digital Object Identifier no. 10.1109/TVCG.2007.1054.

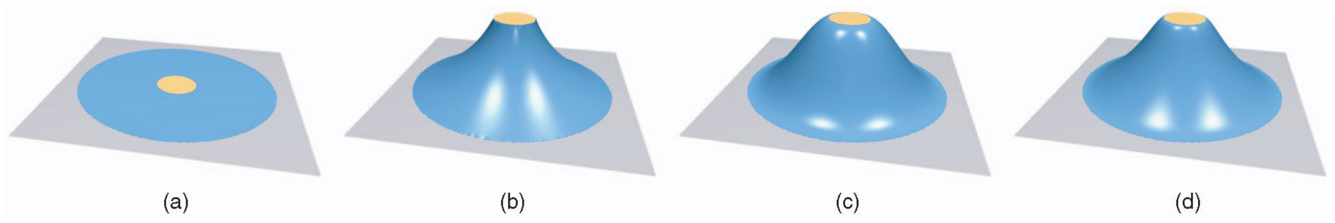

Fig. 1. The original surface $\mathcal{S}$ (a) is edited by minimizing its deformation energy, subject to user-defined constraints that fix the gray part $\mathcal{F}$ of the surface and prescribe the transformation of the yellow handle region $\mathcal{H}$. The linearized energy (2) consists of stretching and bending terms, and the examples show pure stretching with $k_s = 1$, and $k_b = 0$ (b), pure bending with $k_s = 0$, and $k_b = 1$ (c), and a weighted combination with $k_s = 1$, and $k_b = 10$ (d).

goal is to help the reader make practical conclusions about the recited techniques; that is, we wish to answer the following questions: Under which circumstances is each method useful, and how can we choose the appropriate algorithm for a particular scenario?

Deformation tools allow the user to interact with a 3D surface and modify its shape. The quality of a deformation method is measured by its flexibility, the quality of the shapes that it produces, and its intuitive results. An ultimately flexible editing tool would permit arbitrary changes in the surface shape, provided with the right modeling constraints. Such tools may be, however, too general and difficult to control. An intuitive tool allows the user to easily edit the shape, where the manipulation of the controls gives "natural," intuitively expected results. What is a natural deformation? The simplest answer would be something that behaves like a real-world object, which is made of physical material. Physically based deformation techniques are abundant in computer graphics, especially in computer animation [48]. However, when aiming at surface *design* rather than simulation, a merely physically *plausible* result is usually sufficient. Moreover, it is often the case that the desired deformation only looks natural but, in fact, is not possible or difficult to perform in a real-world setting with physical materials. Therefore, one is usually after a deformation that gives aesthetically pleasing results, which might be physically plausible, but the way to achieve them is not necessarily physically correct. An aesthetically pleasing deformation result would preserve the local appearance of the surface under deformation, and in addition, it is generally desired to make smooth or piecewise smooth deformations.

In the following, we review the different sides of the mathematical machinery behind recent surface deformation methods and classify the particular approaches by their specific selections of that machinery (Sections 2 and 3). We then perform a practical comparison between representative methods (Section 4), which visualizes the main aspects of various deformation setups and methods. We dedicate a special section to questions and answers, where we shed light on several obscure points of the deformation approaches (Section 5), and we conclude in Section 6.

2 Multiresolution Editing

As mentioned above, intuitive and, hence, predictable modeling results can be achieved by emulating a physical surface deformation process. Motivated by this, one category of modeling approaches starts from a physically accurate formulation of surface deformation and successively simplifies the computational model in order to achieve higher efficiency and increased numerical stability.

In the following, we therefore start with the introduction of the physically accurate nonlinear thin-shell deformation model for continuous surfaces and show the typically employed simplifications and linearizations (Section 2.1). After that, we discuss its discretization to triangle meshes (Section 2.2), which then leads to merely solving a sparse linear system (Section 2.3). The linearization will be shown to distort fine-scale geometric details, which is taken care of by multiresolution deformations (Section 2.4). In Section 2.5, we describe approaches related to the presented techniques and classify them with respect to the methods used for surface deformation and multiresolution detail preservation.

2.1 Continuous Formulation

The main requirement for physically based surface deformations is an *elastic energy* that measures how much the object has been deformed from its initial configuration. Although, for solid objects, this energy basically considers local stretching within the object, for two-manifold surfaces (so-called *thin shells*), an additional bending term is required.

Let us denote by $\mathcal{S} \subset \mathbb{R}^3$ a two-manifold surface that is parameterized by a function $\mathbf{p} : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$. This surface is to be deformed to $\mathcal{S}'$ by adding to each point $\mathbf{p}(u, v)$ a displacement vector $\mathbf{d}(u, v)$ such that $\mathcal{S}' = \mathbf{p}'(\Omega)$ with $\mathbf{p}' = \mathbf{p} + \mathbf{d}$.

It is known from differential geometry [19] that the first and second fundamental forms $\mathbf{I}(u, v)$ and $\mathbf{II}(u, v) \in \mathbb{R}^{2\times 2}$ can be used to measure geometrically intrinsic (that is, parameterization-independent) properties of $\mathcal{S}$, such as lengths, areas, and curvatures. The change of fundamental forms therefore yields a measure of stretching and bending [64]:

$$E_{\text{shell}}(\mathcal{S}') = \int_\Omega k_s \|\mathbf{I}' - \mathbf{I}\|_F^2 + k_b \|\mathbf{II}' - \mathbf{II}\|_F^2 \,\mathrm{d}u\mathrm{d}v, \qquad (1)$$

where $\mathbf{I}'$ and $\mathbf{II}'$ are the fundamental forms of $\mathcal{S}'$, $\|\cdot\|_F$ denotes a (weighted) Frobenius norm, and the stiffness parameters k_s and k_b are used to control the resistance to stretching and bending. In addition, the energy (1) is invariant under rigid motions (rotation plus translation), which is a geometrically intuitive requirement.

In a modeling application, one is typically not interested in a dynamic time-dependent simulation but, instead, directly solves for the rest state of the deformation process. This requires the minimization of the elastic energy (1) subject to user-defined boundary constraints. As shown in Fig. 1, this typically means fixing certain surface parts $\mathcal{F} \subset \mathcal{S}$ and prescribing displacements for the so-called *handle* region(s) $\mathcal{H} \subset \mathcal{S}$. In an interactive application, $\mathcal{S}'$ has to be recomputed by minimizing E_{shell} each time the user manipulates the boundary constraints, for instance, by moving the handle region $\mathcal{H}$.

However, this nonlinear minimization is computationally too expensive for interactive applications. Hence, it is simplified by replacing the change of the first and second fundamental forms by the first-order and second-order partial derivatives of the displacement function $\mathbf{d}$ [15], [68]:

$$\begin{aligned}\tilde{E}_{\text{shell}}(\mathbf{d}) \;=\; \int_\Omega & k_s\Big(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2\Big) \;+ \\ & k_b\Big(\|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2\Big)\,\mathrm{d}u\mathrm{d}v,\end{aligned} \tag{2}$$

where we use the notation $\mathbf{d}_x = \frac{\partial}{\partial x}\mathbf{d}$ and $\mathbf{d}_{xy} = \frac{\partial^2}{\partial x \partial y}\mathbf{d}$.

The minimization of (2) can be performed efficiently by applying variational calculus, which yields its so-called *Euler-Lagrange* partial differential equation (PDE):

$$-\,k_s\,\Delta\mathbf{d} \;+\; k_b\,\Delta^2\mathbf{d} = \mathbf{0}, \tag{3}$$

which characterizes the minimizer of (2). Δ and Δ^2 represent the Laplacian and the bi-Laplacian operator, respectively,

$$\begin{aligned}\Delta\mathbf{d} = \operatorname{div}\nabla\,\mathbf{d} \;&=\; \mathbf{d}_{uu} + \mathbf{d}_{vv},\\ \Delta^2\mathbf{d} = \Delta(\Delta\mathbf{d}) \;&=\; \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv}\,.\end{aligned}$$

$\mathcal{S}'$ can therefore be found directly by solving (3), again subject to suitable boundary constraints.

In order for the change of the second derivatives in (2) to closely approximate the change of surface curvatures (that is, bending), the parameterization $\mathbf{p}$ should be as close to isometric as possible. Because of that, Ω is typically chosen to equal the initial surface $\mathcal{S}$ such that $\mathbf{d} : \mathcal{S} \to \mathbb{R}^3$ is defined on the manifold $\mathcal{S}$ itself. This is conceptually similar to the data-dependent functionals of Greiner and Loos [26].

As a consequence, the Laplace operator Δ with respect to the parameterization $\mathbf{p}$ turns into the Laplace-Beltrami operator $\Delta_\mathcal{S} = \operatorname{div}_\mathcal{S}\nabla_\mathcal{S}$ with regard to the manifold $\mathcal{S}$ [19]:

$$-\,k_s\,\Delta_\mathcal{S}\mathbf{d} \;+\; k_b\,\Delta_\mathcal{S}^2\mathbf{d} = \mathbf{0}. \tag{4}$$

Notice that this variational minimization is closely related to the design of fair surfaces [47], [68], where the surface area and curvature are minimized instead of their changes, that is, stretching and bending. The linearized *membrane* and *thin-plate* energies corresponding to (2) are defined as

$$\begin{aligned}\tilde{E}_{\text{memb}}(\mathbf{p}) &= \int_\Omega \|\mathbf{p}_u\|^2 + \|\mathbf{p}_v\|^2\,\mathrm{d}u\mathrm{d}v,\\ \tilde{E}_{\text{plate}}(\mathbf{p}) &= \int_\Omega \|\mathbf{p}_{uu}\|^2 + 2\|\mathbf{p}_{uv}\|^2 + \|\mathbf{p}_{vv}\|^2\,\mathrm{d}u\mathrm{d}v\,.\end{aligned} \tag{5}$$

Analogous to (4), their corresponding Euler-Lagrange equations are $-\Delta_\mathcal{S}\mathbf{p} = 0$ and $\Delta_\mathcal{S}^2\mathbf{p} = 0$, respectively. Since the Laplacians or bi-Laplacians vanish on the resulting surfaces, those are stationary surfaces of Laplacian and bi-Laplacian flows typically used in surface smoothing [18], [63]:

$$\mathbf{p}_t = \lambda\Delta_\mathcal{S}\mathbf{p} \quad \text{and} \quad \mathbf{p}_t = -\lambda\Delta_\mathcal{S}^2\mathbf{p}\,.$$

The order k of partial derivatives in the energy or in the corresponding Euler-Lagrange equations $(-1)^k\,\Delta_\mathcal{S}^k\mathbf{d} = 0$ defines the maximum continuity C^{k-1} for interpolating displacement constraints [10]. Hence, minimizing (2) by solving (4) provides C^1 continuous surface deformations, as can also be observed in Fig. 1.

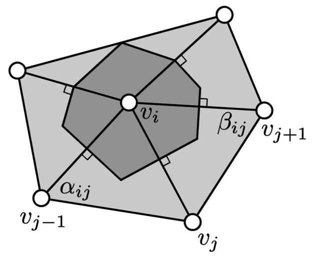

Fig. 2. The angles α_{ij} and β_{ij} and the (dark gray) Voronoi area A_i used to discretize the Laplace-Beltrami $\Delta_\mathcal{S}$ at a vertex v_i in (6) and (7).

2.2 Discretization

The energies and PDEs presented so far were formulated for continuous two-manifold parametric surfaces $\mathcal{S} = \mathbf{p}(\Omega)$. However, our final goal is to represent the surface $\mathcal{S}$ by a triangle mesh, since this allows for higher topological flexibility and computational efficiency [13]. In the following, we denote by $\mathcal{S}$ a triangle mesh whose topology is determined by n vertices $(v_1, \ldots, v_n)$ and m triangles $(t_1, \ldots, t_m)$, $t_i \in \{1, \ldots, n\}^3$, and whose piecewise linear geometric embedding is defined by the vertex positions $\mathbf{p}_i = \mathbf{p}(v_i) \in \mathbb{R}^3$.

In the discrete mesh setting, the user selects certain vertices as the fixed part $\mathcal{F}$ and the handle region $\mathcal{H}$ and typically prescribes either positions $\mathbf{p}'_i = \mathbf{c}_i \in \mathcal{S}'$ or corresponding displacements $\mathbf{d}_i = \mathbf{c}_i - \mathbf{p}_i$ for them. For the rest of the paper, let us assume without loss of generality that from the n vertices $(v_1, \ldots, v_n)$, the first n' vertices are free, whereas the last $k = n - n'$ vertices $(v_{n'+1}, \ldots, v_n)$ are constrained; that is, they constitute the fixed part $\mathcal{F}$ and handle region $\mathcal{H}$.

In order to discretize the above equations for triangle meshes, one can employ either finite differences or finite elements. The Finite Element Method (FEM) leads to more accurate approximations in general, but for thin shell problems like (1) and (2), it theoretically requires C^1 continuous shape functions [5]. In particular, on triangulated manifolds, those are rather complicated to design [15]. Mesh subdivision provides an elegant formulation for C^1 basis functions, as proposed in [16], [17], and [65] for static and dynamic deformations, respectively. As an alternative, so-called nonconforming C^0 elements are frequently and successfully employed in practice [32], although lacking some theoretical guarantees.

In comparison to FEM, a discretization based on finite differences is considerably easier to use, in particular since the Euler-Lagrange equations (4) only require a discretization of the Laplace-Beltrami operator. Given a piecewise linear scalar function $f : \mathcal{S} \to \mathbb{R}$ defined on the mesh $\mathcal{S}$, its discrete Laplace-Beltrami at a vertex v_i has the form

$$\Delta_\mathcal{S} f(v_i) \;=\; w_i \sum_{v_j \in \mathcal{N}_1(v_i)} w_{ij}\big(f(v_j) - f(v_i)\big), \tag{6}$$

where $v_j \in \mathcal{N}_1(v_i)$ are the incident 1-ring neighbors of v_i (see Fig. 2). The discretization depends on the per-vertex normalization weights w_i and the edge weights $w_{ij} = w_{ji}$. Although there are several variations of these weights (see also a comparison in Section 5.1), the de facto standard is the cotangent discretization [18], [45], [51]:

$$w_i = \frac{1}{A_i}\;, \qquad w_{ij} = \frac{1}{2}\big(\cot\alpha_{ij} + \cot\beta_{ij}\big), \tag{7}$$

where α_{ij} and β_{ij} are the two angles opposite to the edge (v_i, v_j), and A_i is the Voronoi area of vertex v_i. The latter is defined in [45] as the area of the surface region built by connecting incident edges' midpoints with triangle circumcenters (for acute triangles) or midpoints of opposite edges (for obtuse triangles), as shown in Fig. 2.

Higher order Laplacians are then simply defined recursively:

$$\begin{aligned}\Delta^k_{\mathcal{S}} f(v_i) &= w_i \sum_{v_j \in \mathcal{N}_1(v_i)} w_{ij} \left(\Delta^{k-1}_{\mathcal{S}} f(v_j) - \Delta^{k-1}_{\mathcal{S}} f(v_i)\right), \\ \Delta^0_{\mathcal{S}} f(v_i) &= f(v_i)\,.\end{aligned}$$

2.3 Numerical Solution

Using (6), the Laplace-Beltrami operator for the whole mesh can be written in matrix notation:

$$\begin{pmatrix} \Delta_{\mathcal{S}} f(v_1) \\ \vdots \\ \Delta_{\mathcal{S}} f(v_n) \end{pmatrix} = \underbrace{\mathbf{M}^{-1}\mathbf{L}_s}_{\mathbf{L}} \cdot \begin{pmatrix} f(v_1) \\ \vdots \\ f(v_n) \end{pmatrix}.$$

Here, $\mathbf{M}$ is a diagonal "mass" matrix of the normalization weights $\mathbf{M}_{ii} = 1/w_i = A_i$, and $\mathbf{L}_s$ is a symmetric matrix containing the edge weights w_{ij}:

$$(\mathbf{L}_s)_{ij} = \begin{cases} -\sum_{v_k \in \mathcal{N}_1(v_i)} w_{ik}\,, & i = j, \\ w_{ij}\,, & v_j \in \mathcal{N}_1(v_i), \\ 0\,, & \text{otherwise.} \end{cases}$$

The Euler-Lagrange equation (4) then leads to a sparse $n \times n$ linear system:

$$\left(-k_s \mathbf{L} + k_b \mathbf{L}^2\right)\mathbf{d} = \mathbf{0}.$$

The boundary constraints are incorporated into this system by moving each column corresponding to a constrained vertex $v_i \in \mathcal{F} \cup \mathcal{H}$ to the right-hand side and removing the respective row from the system (see also Section 5.3). This yields a nonzero right-hand side $\mathbf{b} \in \mathbb{R}^{n' \times 3}$ and leads to an $n' \times n'$ system that is solved for the x, y, and z-components of the displacements $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_{n'})$. Notice that, for notational convenience, we still denote the $n' \times n'$ submatrices by $\mathbf{L}$ and $\mathbf{L}^2$. Premultiplying the above system by $\mathbf{M}$ finally yields the symmetric system

$$\left(-k_s \mathbf{L}_s + k_b \mathbf{L}_s \mathbf{M}^{-1} \mathbf{L}_s\right)\mathbf{d} = \mathbf{M}\mathbf{b}, \qquad (8)$$

which, in addition, can be shown to be positive definite [51].

In an interactive application, the above linear system has to be solved for the deformed surface each time the user changes the boundary constraints, for example, by moving the constrained points, since that changes the right-hand side $\mathbf{b}$. Since the system is sparse, symmetric, and positive definite, an iterative method like conjugate gradients [25] could be employed, but the resulting $O(n^2)$ computational complexity is prohibitive for large meshes. Solving the system on a multigrid hierarchy of successively coarsened meshes, as proposed in [10] and [34], yields linear $O(n)$ complexity and, hence, also works for complex meshes. However, the implementation of an efficient multigrid solver can be quite complex, since it requires several problem-dependent design decisions [1], [54].

Although multigrid solvers are an efficient tool, they do not exploit the fact that the *same* linear system (8) is solved many times (three times each frame), which is only for different right-hand sides $\mathbf{Mb}$. In contrast, sparse direct Cholesky solvers first factor the matrix such that for each new right-hand side, only an efficient back-substitution has to be performed. Thanks to a matrix preordering, the resulting Cholesky factor is also sparse, leading to the basically linear complexity of both the factorization and the back-substitution. In comparison with iterative multigrid solvers, the direct solvers are not only easier to use but also provide better performance for the so-called multiple-right-hand-side problems [7], [54].

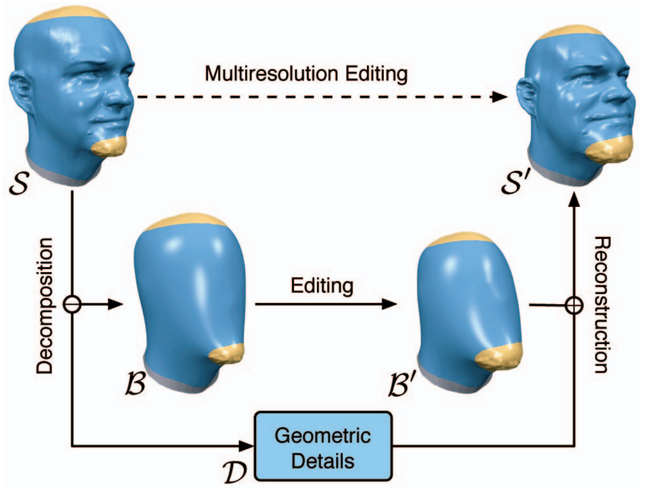

Fig. 3. A multiresolution editing framework consists of three main operators: the *decomposition* operator, which separates the low and high frequencies, the *editing* operator, which deforms the low frequencies, and the *reconstruction* operator, which adds the details back onto the modified base surface.

2.4 Multiresolution Hierarchies

The deformation techniques described above approximate the nonlinear shell energy (1) by the quadratic energy (2) in order to reduce the per-frame costs to the solution of the linear system (8). Although the global energy minimization guarantees smooth and C^1 continuous surface deformations, the linearization causes geometric details and protruding features to be distorted.

As can be seen in Fig. 4, even a pure translation of the handle $\mathcal{H}$ is intuitively expected to locally rotate the geometric details. Unfortunately, determining the required local rotations from position constraints alone is a nonlinear problem and, therefore, cannot be solved by a linearized technique (see Fig. 4b). In order to still be able to achieve intuitive detail preservation while using a linear deformation technique, one can complement the linear deformation model by a so-called *multiresolution* or *multiscale hierarchy*.

The main idea of multiresolution deformations is to consider the surface $\mathcal{S}$ as a "geometric signal" and to separate the low frequencies from the high frequencies. The low frequencies constitute the global shape of the model and are represented by a smooth base surface $\mathcal{B}$. The high frequencies are the difference between $\mathcal{S}$ and $\mathcal{B}$, that is, the geometric details $\mathcal{D} = \mathcal{S} \ominus \mathcal{B}$. The original surface $\mathcal{S}$ can be reconstructed by adding the geometric details to the base surface, $\mathcal{S} = \mathcal{B} \oplus \mathcal{D}$. A multiresolution deformation can now be computed by deforming $\mathcal{B}$ to $\mathcal{B}'$ and reconstructing $\mathcal{S}' = \mathcal{B}' \oplus \mathcal{D}$. This modifies the global shape $\mathcal{B}$ but preserves the fine-scale details $\mathcal{D}$. The whole process is schematically depicted in Fig. 3.

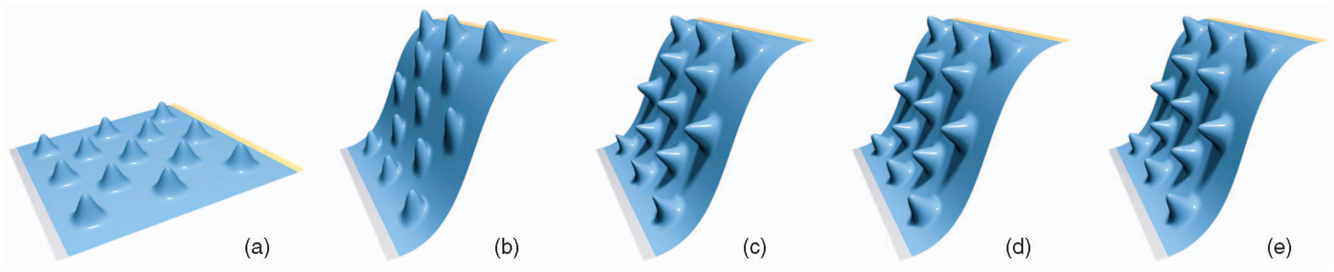

Fig. 4. The rightmost strip $\mathcal{H}$ of the bumpy plane (a) is translated up. The intuitive local rotations of geometric details cannot be achieved by a linearized deformation (8) alone, as can be seen in (b), but require a multiresolution decomposition. Normal displacements (c) correctly rotate local details but cause distortions under bending deformations, which can be seen in the leftmost row of bumps. The nonlinear displacement volumes (d) and the linear deformation transfer (e) provide more intuitive results.

Note that, in other contexts, the term multiresolution hierarchy is also used to denote *topological* hierarchies of coarser and coarser meshes [24]. In contrast, we are considering it as a *geometric* hierarchy of smoother and smoother meshes. Although for subdivision surfaces, these two concepts are coupled, for arbitrary irregular meshes, they are not.

In order to compute the low-frequency base surface $\mathcal{B}$, one typically removes high frequencies from $\mathcal{S}$ by mesh smoothing [18], [29], [63]. In the example shown in Fig. 3, the thin-plate energy (5) was minimized by solving $\Delta_{\mathcal{S}}^2 \mathbf{p} = \mathbf{0}$. The special operators $\ominus$ and $\oplus$ are the multiresolution *decomposition* and *reconstruction* and depend on the chosen representation of the geometric details $\mathcal{D}$. This, and the way that $\mathcal{B}$ is computed, is where the existing multiresolution editing techniques differ.

A straightforward approach is to restrict $\mathcal{S}$ and $\mathcal{B}$ to have the same connectivity and to encode their geometric difference $\mathcal{D}$ by per-vertex *displacement vectors* $\mathbf{h}_i$ [29], [34], [74]:

$$\mathbf{p}_i \;=\; \mathbf{b}_i + \mathbf{h}_i \;, \quad \mathbf{h}_i \in \mathbb{R}^3,$$

where $\mathbf{b}_i \in \mathcal{B}$ is the vertex corresponding to $\mathbf{p}_i \in \mathcal{S}$. The vectors $\mathbf{h}_i$ have to be encoded in local frames with respect to $\mathcal{B}$ [22], determined by the normal vector $\mathbf{n}_i$ and two vectors spanning the tangent plane. When the base surface $\mathcal{B}$ is deformed to $\mathcal{B}'$, the displacement vectors rotate according to the rotations of the base surface's local frames, which then leads to a plausible detail reconstruction for $\mathcal{S}'$.

However, as we will see below, long displacement vectors might lead to instabilities, particularly for bending deformations. As a consequence, for numerical robustness, the vectors should be as short as possible, which is the case if they connect vertices $\mathbf{p}_i \in \mathcal{S}$ to their closest surface points on $\mathcal{B}$ instead of their corresponding vertices of $\mathcal{B}$. This idea leads to *normal displacements* that are perpendicular to $\mathcal{B}$, that is, parallel to its normal field $\mathbf{n}$:

$$\mathbf{p}_i \;=\; \mathbf{b}_i + h_i \cdot \mathbf{n}(\mathbf{b}_i) \;, \quad h_i \in \mathbb{R}. \tag{9}$$

The difference in length of general displacement vectors and normal displacements typically depends on how much $\mathcal{B}$ differs from $\mathcal{S}$. For instance, in Fig. 3, the general displacements are, on the average, about nine times longer than the normal displacements.

Notice, however, that normal displacements require a resampling of either $\mathcal{S}$ [30], [37] or $\mathcal{B}$ [35]. Since the resampling of the smooth surface $\mathcal{B}$ causes fewer aliasing artifacts than that of the high-frequency surface $\mathcal{S}$, the latter is the preferred approach. Hence, for each point $\mathbf{p}_i \in \mathcal{S}$, a local Newton iteration [35] finds a base point $\mathbf{b}_i \in \mathcal{B}$ such that

$$(\mathbf{p}_i - \mathbf{b}_i) \times \mathbf{n}(\mathbf{b}_i) \;=\; \mathbf{0}.$$

As a consequence, the base points $\mathbf{b}_i \in \mathcal{B}$ are not necessarily the vertices of $\mathcal{B}$. This also implies that the connectivity of $\mathcal{S}$ and $\mathcal{B}$ is no longer restricted to be identical, which can be exploited in order to remesh the base surface $\mathcal{B}$ for the sake of higher numerical robustness [11].

The main problem of normal displacements is that neighboring displacement vectors are not coupled in any way. When bending the surface in a convex or concave manner, the angle between neighboring vectors increases or decreases, leading to an undesired change of volume (see Figs. 4c and 5a). If displacement vectors cross each other, which happens if the curvature of $\mathcal{B}'$ becomes larger than the displacement length h_i, then it might even result in local self-intersections.

These problems are addressed by *displacement volumes* [9]. Each triangle $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ of $\mathcal{S}$, together with the corresponding points $(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k)$ on $\mathcal{B}$, defines a triangular prism, the

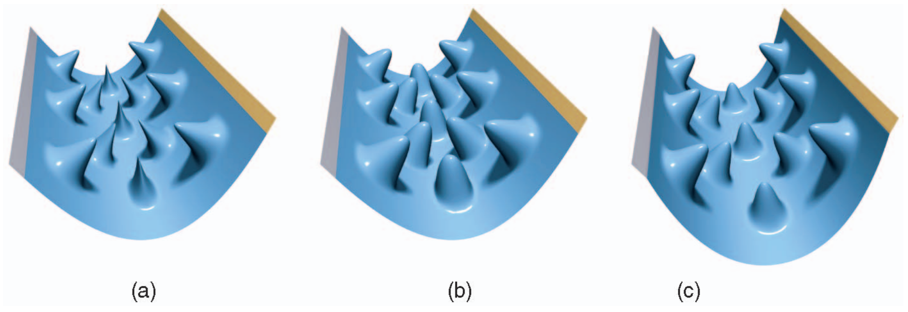

Fig. 5. A concave bending of the bumpy plane in Fig. 4. Multiresolution representations based on normal displacements (a) unnaturally distort geometric details and almost lead to self-intersections, whereas displacement volumes (b) and deformation transfer (c) achieve more natural results.

volume of which is kept constant during deformations. The local volume preservation leads to more intuitive detail reconstructions and avoids local self-intersections (see Figs. 4d and 5b). However, the improved detail preservation comes at the higher computational cost of a nonlinear detail reconstruction process.

Botsch et al. [14] recently proposed a multiresolution representation that provides results similar to displacement volumes (see Figs. 4e and 5c) but requires solving a sparse linear system only. It employs the deformation transfer framework [61] in order to transfer the deformation $\mathcal{B} \mapsto \mathcal{B}'$ to the fine-scale surface $\mathcal{S}$, which then yields $\mathcal{S}'$. They also show how the actual computation of the deformation transfer can be simplified such that the approach only requires the solution of a linear Poisson system (see also Section 5.5).

2.5 Related Approaches

In this section, we list and discuss several deformation approaches related to the techniques described so far and categorize them according to the energy minimization, multiresolution representation, and surface representation that they employ.

2.5.1 Subdivision Surfaces

Zorin et al. presented one of the first multiresolution editing approaches [74]. Their method is based on subdivision surfaces, and the geometric difference between successive subdivision hierarchy levels is encoded by general displacement vectors. Similarly, the displaced subdivision surfaces of Lee et al. [37] also represent the base surface by a subdivision surface but use normal displacements for the geometric details. In both cases, if the input mesh is not a subdivision surface, then it has to be remeshed to subdivision connectivity, which might lead to resampling artifacts.

To overcome this, Marinov and Kobbelt [44] represent only the base surface $\mathcal{B}$ by a subdivision surface and use normal displacements to encode the difference between $\mathcal{B}$ and the original *irregular* mesh $\mathcal{S}$. Marinov et al. [43] also presented a hardware-accelerated graphics processing unit (GPU) implementation of the latter multiresolution deformation technique.

The subdivision basis functions guarantee smooth deformations for these approaches but do not necessarily minimize a physically based deformation energy. The main drawback of subdivision-based methods is that global deformations have to be controlled on a coarse subdivision level, where only a small number of control points are available. This limits the modeling flexibility, since the control points might not be at the right position to perform a desired deformation. Moreover, the support of the deformation is predetermined by the coarse control grid and, hence, cannot be chosen precisely on the detailed mesh.

2.5.2 Irregular Meshes

Kobbelt et al. [34] deform irregular triangle meshes based on the variational energy minimization, as described in the previous sections, which therefore leads to high-quality physically based surface deformations. In contrast to the subdivision-based approaches, their multiresolution hierarchy is based on levels of different smoothness, not of different mesh complexities. The hierarchy levels are connected by normal displacements. Only to speed up the solution of the linear system (8) do they employ a multigrid hierarchy of topologically coarsened meshes. Their approach allows prescribing the constraints for each individual vertex of $\mathcal{S}$, which are then interpolated by an energy minimizing and, hence, fair deformation function $\mathbf{d}$.

Guskov et al. [29] apply hierarchical smoothing on a multiresolution pyramid of irregular meshes, which are coupled by general displacement vectors. However, similar to the subdivision-based methods, their approach is limited by the small number of control points available on the coarse hierarchy levels. Like the subdivision approaches, their deformations are smooth in a C^k sense but do not necessarily minimize a bending energy.

Lee [38] first parameterizes the editing area over the 2D unit square, interpolates user-defined deformation constraints there, and, finally, maps per-vertex displacements back onto the 3D surface $\mathcal{S}$. This approach incorporates precise per-vertex constraints, and the multilevel B-spline interpolation allows controlling their influence radius. However, the required parameterization might lead to geometric distortions.

Botsch and Kobbelt [10] extend [34] by anisotropic and triharmonic deformations and provide control of boundary continuity on a per-vertex basis. They also exploit the fact that the handle vertices $\mathcal{H}$ are typically transformed only affinely by the user. This allows precomputing the linear basis functions for the deformation $\mathbf{d}$, which can be evaluated in each frame more efficiently than solving the linear system (8). Although normal displacements were used in the original paper [10], a multiresolution representation based on deformation transfer was shown in [14] to yield better results, particularly for bending deformations.

Although all these approaches work well in many cases, there are limitations inherent to the linearization: Large deformations, particularly large rotations, might cause artifacts when performed in a single step (Section 4). In addition, all approaches need a multiresolution decomposition to correctly deform fine-scale details, which might require a more complicated multilevel hierarchy for geometrically or topologically complex models. These drawbacks were a major motivation for the deformation approaches based on differential coordinates, which are described in the next section.

3 Deformation Based on Differential Surface Representations

Surface deformation approaches based on differential representations have gained significant popularity over the past three years, probably mainly due to their robustness, speed, and ease of implementation. The main idea behind this family of deformation techniques is to use a surface representation that puts the local differential properties in focus and to preserve these differential properties under deformation, aspiring to obtain an intuitive detail-preserving deformation result. Hence, the motivation is to achieve a globally smooth deformation, which is induced by the modeling constraints, that at the same time preserves the local characteristics of the surface. Generally speaking, this is achieved by constructing the differential representation of the input surface, then manipulating this representation according to the modeling constraints, and finally performing “integration” or reconstruction of the surface coordinates from the modified differential representation. Various techniques differ in the particular differential properties that they use (Section 3.1) and the manipulation thereof (Section 3.2), but the general framework remains the same.

Differential surface manipulation was inspired by gradient-domain image manipulation. It has been noticed that the gradients of the image intensity function (or the three color channels) contain important visual information to which humans are sensitive. Many image techniques exploit this fact by applying certain manipulations to the input image gradients $\mathbf{g} = \nabla I$ and then reconstruct the resulting image by a global optimization process that looks for an image I' whose gradients are as close as possible to the modified gradients $\mathbf{g}'$:

$$I' \;=\; \underset{I'}{\operatorname{argmin}} \int_\Omega \|\nabla I' - \mathbf{g}'\|^2 \mathrm{d}x\mathrm{d}y\,.$$

Here, Ω denotes the domain of the image manipulation (the rectangular grid or part of it). By deriving the Euler-Lagrange equations of the functional above, we arrive at the famous Poisson equation

$$\Delta I' = \operatorname{div} \mathbf{g}', \tag{10}$$

to which some boundary conditions are added. One example of using this image manipulation framework is high dynamic range compression [21], where the input image I has intensities with too high a contrast to display it on a conventional display. The compression method modifies the intensity gradients $\mathbf{g}$ such that strong contrasts are attenuated while small intensity variations are preserved. The resulting image is reconstructed by solving the Poisson equation with Neumann boundary conditions.

Another example of gradient manipulation that comes even closer to surface editing is Poisson Image Editing [50]. It is a set of image editing tools, the most prominent one being Poisson cloning, where a part cut out from one image is seamlessly pasted onto another image. The correct seamless transition between the target background image and the pasted source image part is achieved by feeding the right Dirichlet boundary conditions to the Poisson equation: The gradients of the image inside the pasted region Ω are required to equal the source image gradients, whereas the boundary conditions require the image values along the boundary to equal the target image:

$$I'|_{\partial\Omega} \;=\; I_{\text{target}}|_{\partial\Omega}\,.$$

In the spirit of the above techniques, differential surface manipulation approaches try to follow the same framework: manipulate the differential representation according to the task and then reconstruct the surface by means of a quadratic optimization with appropriate boundary conditions that stem from the user-defined modeling constraints (for the most part, Dirichlet boundary conditions that would prescribe the positions of some points on the surface). The discrete energy that these approaches minimize usually has the form

$$\mathbf{p}' = \arg\min_{\mathbf{p}'} \sum_i A_i \|\mathcal{D}(\mathbf{p}_i') - \mathbf{l}_i\|^2, \tag{11}$$

where $\mathcal{D}$ is an operator that extracts the differential quantities $\mathbf{l}_i = \mathcal{D}(\mathbf{p}_i)$ from the surface geometry, and A_i are local area elements such as the Voronoi areas defined in Section 2.2. If $\mathcal{D}$ can be expressed as a global linear operator $\mathbf{D}$, and we denote by $\mathbf{M}$ the diagonal matrix containing the weights A_i, then the above minimization sums up to solving the normal equations:

$$\mathbf{D}^T\mathbf{M}\mathbf{D}\,\mathbf{p}' \;=\; \mathbf{D}^T\mathbf{M}\,\mathbf{l}. \tag{12}$$

However, surfaces in 3D have several significant differences from images: They are generally not height functions, and when represented by polygonal meshes, they are not sampled over a regular domain. Moreover, there is a geometric connection among the three mesh coordinate functions (x, y, z) such that manipulating them in a decoupled fashion is possible only after some linearization assumptions. These fundamental differences prevent the direct carry-over of the technology developed for images onto surfaces, and the techniques that we review next deal with the challenge in various ways.

3.1 Differential Representations

Here, we review the different differential representations and show their basic surface reconstruction methods.

3.1.1 Gradient-Based Representation

The first approach to directly translate the gradient-based approach from image editing to surface editing [71] was to consider the gradients of the surface coordinate functions x, y, and z, defined over the base domain Ω (which is typically the input mesh $\mathcal{S}$). In the continuous formulation, the deformed surface is defined by the coordinate functions x', y', and z' that minimize

$$\int_\Omega \|\nabla x' - \mathbf{g}_x\|^2 \mathrm{d}u\mathrm{d}v$$

(and the same for y' and z'), under some modeling constraints, where $\mathbf{g}_x = \nabla x$ are the gradients of the original surface coordinate functions. The Euler-Lagrange equation of this minimization is the Poisson equation

$$\Delta x' \;=\; \operatorname{div} \mathbf{g}_x\,. \tag{13}$$

It is simple to define the gradients of the coordinate functions in the discrete setting: The mesh is a piecewise linear surface and, thus, the gradients of the coordinate functions are constant over each face. Intuitively, if the base domain is the mesh itself, then, in each triangle, the gradient of the x function is the projection of the unit x-axis vector $(1,0,0)^T$ onto the triangle's plane and similarly for the other two coordinate functions. Formally, let a piecewise linear scalar function f on the domain mesh $\mathcal{S}$ be defined by barycentric interpolation of per-vertex values $f_i = f(v_i)$ such that

$$f(u,v) = \sum_{i=1}^n f_i\,\phi_i(u,v),$$

where (u,v) are the parameters over the domain mesh and $\phi_i(\cdot)$ are the piecewise linear "hat" basis functions associated with the domain mesh vertices, that is, $\phi_i(v_k) = \delta_{ik}$. The gradient of f is then

$$\nabla f(u,v) = \sum_{i=1}^n f_i\,\nabla\phi_i(u,v)\,. \tag{14}$$

The gradients $\nabla\phi_i(u,v)$ are constant within each domain mesh face. If $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ are the vertices of a domain mesh triangle, then the gradients of the corresponding hat functions ϕ_i, ϕ_j, and ϕ_k are

$$(\nabla\phi_i, \nabla\phi_j, \nabla\phi_k) = \begin{pmatrix} (\mathbf{p}_i - \mathbf{p}_k)^T \\ (\mathbf{p}_j - \mathbf{p}_k)^T \\ \mathbf{n}^T \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix},$$

where $\mathbf{n}$ is the unit normal of the triangle. This formulation ensures that the gradients lie in the triangle's plane (for details on the derivation, see [14]). One can formulate (14) by using a global operator $\mathbf{G}$, which is expressed as a $3m \times n$ matrix that multiplies the n-vector $\mathbf{f}$ of the discrete values f_i to obtain a vector of m stacked gradients, each gradient having three spatial coordinates (m being the number of triangles). Thus, one can write down the following for the input mesh:

$$\mathbf{G}\mathbf{x}' = \mathbf{g}_x,$$

and the same for the other two coordinate functions. When the gradients of the surface are known (as functions over the domain mesh) and the coordinate functions are unknown, we can find them by minimizing (11), with $\mathbf{G}$ being the differential operator. Thus, we solve (12), where the $3m \times 3m$ weight matrix $\mathbf{M}$ contains the areas of the triangles:

$$\mathbf{G}^T\mathbf{M}\mathbf{G}\mathbf{x}' = \mathbf{G}^T\mathbf{M}\mathbf{g}_x\,.$$

The matrix $\mathbf{G}^T\mathbf{M}$ corresponds to the discrete divergence operator associated with the domain mesh, and $\mathbf{G}^T\mathbf{M}\mathbf{G}$ is none other than the cotangent discretization of the Laplace-Beltrami operator [14], as discussed in Section 2.2, so we can simply write

$$\mathbf{L}_s\mathbf{x}' = \mathbf{G}^T\mathbf{M}\mathbf{g}_x, \tag{15}$$

which is the discretized version of (13).

To deform a surface by using this gradient representation, a direct adaptation of Poisson Image Editing [50] would be simply to add Dirichlet boundary conditions to (15), corresponding to user-defined modeling constraints:

$$\mathbf{p}'_i = \mathbf{c}_i \tag{16}$$

for the fixed vertices $\mathcal{F}$ and handle vertices $\mathcal{H}$.

However, the result of such an editing approach is not satisfactory because it tries to preserve the original mesh gradients with their orientation in the global coordinate system. This ignores the fact that, in the deformed surface, the gradients should rotate, since they always lie in the triangles' planes, which transform as a result of the surface deformation. The effect is demonstrated in Fig. 6b, clearly showing that the resulting deformation is not intuitive.

This *local transformations problem* is central in *all* differential editing approaches. It stems from the fact that the representation is dependent on the particular placement of the surface in space, i.e., it is not rigid invariant, and thus, when the surface deforms, the representation must be updated. Unfortunately, it is a chicken-and-egg problem in its essence because the deformed surface is unknown. We review the different approaches to obtain the local transformations in Section 3.2.

3.1.2 Laplacian-Based Representation

Laplacian-based approaches represent the surface by the so-called differential coordinates or Laplacian coordinates [3], [59]. These coordinates are obtained by applying the Laplacian operator to the mesh vertices; that is, taking $f \equiv \mathbf{p}$ in (6), the resulting vector is the mean curvature normal:

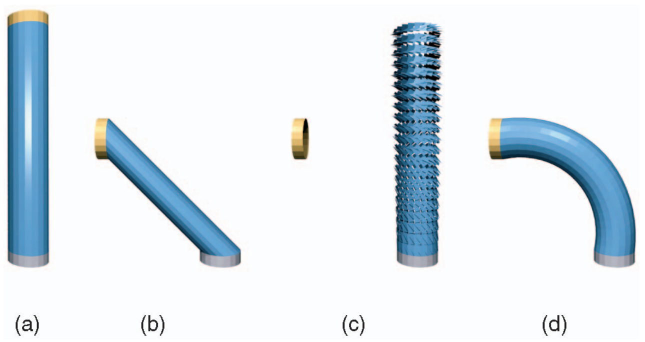

Fig. 6. Using gradient-based editing to bend the cylinder (a) by 90 degrees. (b) Reconstructing the mesh from new handle positions but *original* gradients distorts the object. (c) Applying damped local rotations derived from (25) to the individual triangles breaks up the mesh, but (d) solving the Poisson system (15) reconnects it and yields the desired result.

$$\boldsymbol{\delta}_i = \Delta_{\mathcal{S}}(\mathbf{p}_i) = -H_i\mathbf{n}_i, \tag{17}$$

where H_i is the mean curvature $H = \kappa_1 + \kappa_2$ at v_i. Modeling directly with these coordinates is meant to circumvent the need to decompose the surface into a low-frequency base surface and high-frequency details, as in the multiresolution approaches discussed in Section 2.4.

Laplacian editing was developed concurrently with gradient-based editing, and similar to the latter, the first naive attempt would be to formulate the deformation by directly minimizing the difference from the input surface coordinates $\boldsymbol{\delta}_i$. In the continuous setting, the energy minimization is formulated as

$$\min_{\mathbf{p}'} \int_\Omega \|\Delta\mathbf{p}' - \boldsymbol{\delta}\|^2 \mathrm{d}u\mathrm{d}v\,. \tag{18}$$

The Euler-Lagrange equation derived for the above minimization is

$$\Delta^2\mathbf{p}' = \Delta\boldsymbol{\delta}\,.$$

When we consider this equation, taking the input surface as the parameter domain, the Laplace operator turns into a Laplace-Beltrami $\Delta_{\mathcal{S}}$, and we arrive at the discretized equation

$$\mathbf{L}^2\mathbf{p}' = \mathbf{L}\boldsymbol{\delta}, \tag{19}$$

which can be separated into three coordinate components. The equation is constrained by the modeling constraints of the form (16). It is also possible to arrive at this equation by discretizing the continuous energy (18):

$$\min_{\mathbf{p}'} \sum_i A_i \|\Delta_{\mathcal{S}}(\mathbf{p}'_i) - \boldsymbol{\delta}_i\|^2\,. \tag{20}$$

This minimization amounts to solving (12) with $\mathbf{L} = \mathbf{M}^{-1}\mathbf{L}_s$ as the differential operator and the Voronoi areas stacked into the diagonal matrix $\mathbf{M}$:

$$\begin{aligned} \mathbf{L}^T\mathbf{M}\mathbf{L}\mathbf{p}' &= \mathbf{L}^T\mathbf{M}\boldsymbol{\delta}, \\ (\mathbf{M}^{-1}\mathbf{L}_s)^T\mathbf{M}(\mathbf{M}^{-1}\mathbf{L}_s)\mathbf{p}' &= (\mathbf{M}^{-1}\mathbf{L}_s)^T\mathbf{M}\boldsymbol{\delta}, \\ \mathbf{L}_s\mathbf{M}^{-1}\mathbf{L}_s\mathbf{p}' &= \mathbf{L}_s\boldsymbol{\delta}\,. \end{aligned} \tag{21}$$

Note that if we multiply both sides of (21) by $\mathbf{M}^{-1}$, then we arrive at the bi-Laplacian equation (19). Moreover, if the right-hand side is set to zero (that is, $\boldsymbol{\delta} = \mathbf{0}$), the equation solves for the minimizer of the linear thin-plate energy (5). This formulation was used to define the so-called Least Squares Meshes [58]—smooth surfaces formed by mesh connectivity and a sparse set of control points with geometry, incorporated by (16).

The positional constraints (16) may be incorporated as either hard or soft constraints. Hard constraints lead to the elimination of corresponding rows and columns of the system matrix, whereas soft constraints are added as additional terms of the form $\lambda\|\mathbf{p}_i - \mathbf{c}_i\|^2$ to the discrete energy functional in (20) (see Section 5 for details). Although the system is very simple and can be efficiently solved by the sparse direct methodologies mentioned earlier, the results only look satisfactory when the starting surface is a membrane or a thin plate (that is, the right-hand side of (21) is zero). In any other case, the surface details are distorted for the same reasons as with gradient-based editing or the variational minimization discussed in Section 2. The system tries to preserve the orientation of the Laplacian vectors with respect to the global coordinate system, whereas, in reality, they should rotate with the deformed surface.

3.1.3 Local-Frame-Based Representation

In search of a rigid-invariant shape representation, frame-based representation [42] turns to classical differential geometry and attempts to import elements from the theory of moving frames [28] into the discrete setting. The representation is inspired by the geometric invariance of the fundamental forms and aspires to formulate the deformation problem in the spirit of the elastic energy (1). This frame-based representation consists of a set of orthonormal frames $(\mathbf{a}_i, \mathbf{b}_i, \mathbf{n}_i)$ attached to each mesh vertex and sets of coefficients describing the relations between the frames, as well as the coordinates of the mesh 1-rings with respect to the frames. More precisely, the relationship between the local frames of two neighboring vertices v_i and v_j in the input surface is recorded by the coefficients of a 3×3 matrix $\mathbf{A}_{ij}$ such that

$$\left(\mathbf{a}_i - \mathbf{a}_j, \mathbf{b}_i - \mathbf{b}_j, \mathbf{n}_i - \mathbf{n}_j\right) \;=\; \mathbf{A}_{ij}(\mathbf{a}_i, \mathbf{b}_i, \mathbf{n}_i)\,. \qquad (22)$$

The 1-ring vectors are encoded with respect to the local frame by a set of three coefficients $(\alpha_{ij}, \beta_{ij}, \gamma_{ij})$ per edge:

$$\mathbf{p}_j - \mathbf{p}_i \;=\; \alpha_{ij}\mathbf{a}_i + \beta_{ij}\mathbf{b}_i + \gamma_{ij}\mathbf{n}_i\,. \qquad (23)$$

It is easy to verify that if the choice of the local frames is rotation invariant, then so are the coefficients $\mathbf{A}_{ij}$, α_{ij}, β_{ij}, and γ_{ij}, which singles out this representation from other differential coordinates. The (overdetermined) linear equation (23) can be used to model deformations in two steps: First, the local frames of the deformed surface are determined by one of the methods described in the next section, and then, the vertex positions are solved for in the least squares sense by using (23), where the frames obtained in the first step are plugged into the right-hand side. Note that the normal equations matrix of (23) is the symmetric uniform Laplacian.

3.2 Local Transformations

As mentioned above, the main problem in detail-preserving surface deformation is to correctly define the local transformations that occur during deformation. By "correct," one usually means such deformations that the local surface features retain their relative orientation and, possibly, their size. Therefore, the local transformations should be as close as possible to pure rotations and translations. In some cases, isotropic scales are also admissible. There are several methods to define the local transformations in the literature, as we describe below. Note that the problem is inherently nonlinear, so one can only offer a reasonable estimate if one desires to avoid global nonlinear optimizations. Once the local transformations $\mathbf{T}_i$ are defined, the differential representation of the input mesh is transformed by these, and the associated reconstruction problem (11) is solved to obtain the deformed surface:

$$\min_{\mathbf{p}'} \sum_i A_i \|\mathcal{D}(\mathbf{p}'_i) - \mathbf{T}_i(\mathbf{l}_i)\|^2\,. \qquad (24)$$

3.2.1 Geodesic Propagation

This method of local transformation assignment relies on additional user input to disambiguate the local transformations that should occur in the deformed surface. In addition to positional constraints (16), the user is required to provide a transformation matrix for the handle $\mathcal{H}$ that he/she manipulates (this can be deduced, for example, from a transform user interface attached to the handle, where the user visually manipulates the rotation axes). The provided handle transformation $\mathbf{T}$ is decomposed into rotational and scaling/shear components $\mathbf{T} = \mathbf{RS}$ [55], and both are interpolated over the region of interest (ROI) on the mesh according to the geodesic distance from the handle:

$$\mathbf{T}_i = \operatorname{slerp}(\mathbf{R}, \mathbf{I}, 1 - s_i) \cdot ((1 - s_i)\mathbf{S} + s_i\mathbf{I}), \qquad (25)$$

where slerp denotes quaternion interpolation and s_i is a value between 0 and 1, which is proportional to the geodesic distance from the handle $\mathcal{H}$:

$$s_i = \frac{\operatorname{dist}(\mathbf{p}_i, \mathcal{F})}{\operatorname{dist}(\mathbf{p}_i, \mathcal{F}) + \operatorname{dist}(\mathbf{p}_i, \mathcal{H})}\,.$$

In this fashion, the handle transformation is propagated over the ROI, and small-scale surface details are properly transformed.

3.2.2 Harmonic Propagation

Discrete geodesic distances turned out to be a suboptimal parameter to propagate the transformations because they may be nonsmooth and also attenuate the transformations of highly protruding features too much [42], [72]. Harmonic functions were proposed instead: The values of s_i are determined from a harmonic scalar field $\mathbf{s}$ defined over the mesh vertices by the Laplace equation

$$\mathbf{Ls} \;=\; \mathbf{0}, \qquad (26)$$

where the Dirichlet boundary conditions require $s_i = 0$ for $v_i \in \mathcal{F}$ and $s_i = 1$ for $v_i \in \mathcal{H}$. This results in a smooth transformation propagation with the added advantage of economic computation, when the matrix in (26) is the same matrix used for editing (15); thus, the same factorization can be used.

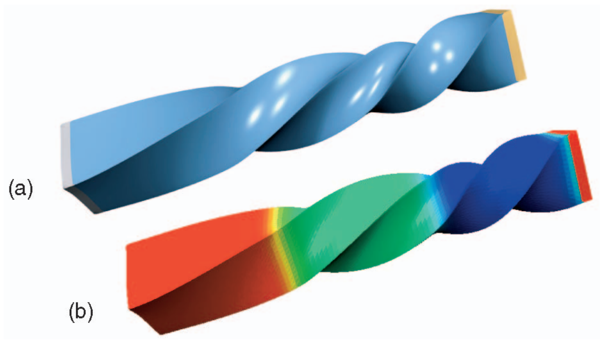

Fig. 7. (a) A nonuniform twist using the material-aware deformation technique [52], (b) with stiffness weights color-coded.

3.2.3 Material-Aware Propagation

It is possible to control the surface material properties, namely, local stiffness, by carefully designing the interpolation weights s_i, as done in [52]. The user may define stiffness by a painting interface that provides a scalar field φ_{ij} over the mesh edges. The interpolation weights are then determined by solving the following weighted quadratic minimization:

$$\min_{\mathbf{s}} \sum_{j \in \mathcal{N}_1(v_i)} \varphi_{ij} \| s_i - s_j \|^2 .$$

Thus, where the stiffness parameter φ_{ij} is higher, the interpolation parameters tend to be closer, which assigns similar local transformations and makes the surface locally stiffer (see Fig. 7). The constraints of the minimization above are the same as in harmonic propagation, that is, 0 for fixed vertices $\mathcal{F}$ and 1 for the handle $\mathcal{H}$. Curiously, the minimization of the quadratic energy above leads to a Laplace equation (see [20] for details), weighted by the stiffness parameters.

Note that any transformation propagation technique would only work if the transformation of the handle (for example, rotation) is actually provided. If the handle is only translated, then all propagated local transformations will equal the identity. This phenomenon is called *translation insensitivity* [12] because the method might not generate intuitive local rotations when the modeling constraint contains translation.

3.2.4 Explicit Optimization

This method produces the local transformations by solving for the local frames of the deformed surface in the least squares sense [42] by using (22). The coefficients are derived from the original mesh. The idea is to optimize the local transformations so as to preserve the relationships between the local frames. The handle frames need to be constrained, similarly to the transformation propagation methods, such that this method is also translation insensitive. Note that the least squares solution might produce nonorthonormal frames, which may lead to area and volume shrinkage. If the modeling constraints on the frames involve solely orthogonal transformations, it is advisable to normalize and orthogonalize the frames.

3.2.5 Estimation from an Initial-Guess Solution

Local transformations may be estimated from a naive solution of the deformation (computed without transforming the differential representation). This approach is akin to the multiresolution techniques in the sense that the initial guess of the deformed surface lacks details. The details are then transformed by the estimated local transformations. The local transformations are estimated from the initial guess by comparing k-ring vectors $\mathbf{V}_i$ of the original surface with corresponding k-rings $\mathbf{V}'_k$ in the deformed surface [41] (the columns of $\mathbf{V}_i$ are vectors from the center vertex $\mathbf{p}_i$ to its k-order neighbors and, in addition, the normal at $\mathbf{p}_i$) or, alternatively, from pairs of corresponding triangles and their normals [14]. A least squares fit of the local transformation is

$$\tilde{\mathbf{T}}_i = \mathbf{V}'_k \mathbf{V}_k^+ ,$$

where $(\cdot)^+$ denotes the pseudoinverse of a matrix. The local transformations are then orthogonalized to obtain rigid $\mathbf{T}_i$s (isotropic scales may also be allowed, depending on the user's requirements). The assumption is that the deformed surface is mostly smooth; thus, the naive solution provides a good guess for the deformed underlying base surface and only small-scale details need to be rotated to correct their orientation. The larger k is, the smoother the estimation becomes, at larger computational cost, naturally. Note that this method is *sensitive to translations* of the handle.

3.2.6 Implicit Optimization

Implicit optimization of transformations tries to tackle the "chicken-and-egg" problem of local transformations by expressing these unknown transformations in terms of the unknown deformed geometry: $\mathbf{T}_i = \mathbf{T}_i(\mathbf{p}')$. The local transformations are then found together with the deformed surface in the global optimization process (24). Notice that the three spatial mesh coordinate functions are no longer decoupled in the global optimization. The local transformations should be also constrained to rigid or similarity transformations only. The coefficients of $\mathbf{T}_i$ are functions of $\mathbf{p}'$. In the optimal case, $\mathbf{T}_i$ would be constrained to rotations alone, but this would require the use of nonlinear combinations of $\mathbf{p}'$, turning (24) into a global nonlinear optimization. It is possible, however, to linearize the *similarity* transformations [60]:

$$\mathbf{T}_i = \begin{pmatrix} s & -h_3 & h_2 \\ h_3 & s & -h_1 \\ -h_2 & h_1 & s \end{pmatrix} . \tag{27}$$

The parameters s and $\mathbf{h}$ are determined by writing down the desired transformation constraints, that is, $\mathbf{T}_i(\mathbf{p}_i - \mathbf{p}_j) = \mathbf{p}'_i - \mathbf{p}'_j$, for each $v_j \in \mathcal{N}_k(v_i)$ and, thus, extracting s and $\mathbf{h}$ as linear combinations of $\mathbf{p}'$. The precise derivation can be found in [40]. Plugging the linear expression for $\mathbf{T}_i$ back into (24) results in a linear least squares problem. It should be noted that the expression for $\mathbf{T}_i$ accommodates isotropic scales in addition to rotations; therefore, when the handle is "pulled," for example, the deformed surface will scale and inflate. When this effect is undesired, it can be eliminated by scaling the differential representations (gradients/Laplacians) of the deformed surface back to their length in the original surface and solving (11) with this corrected representation. Another solution, as proposed in [33], is to scale the triangles of the deformed surface back to their original size and restitch the mesh by using the Poisson setup (15). The implicit optimization of local transformations is also sensitive to translational modeling constraints.

3.3 Related Approaches

In this section, we list and discuss several deformation approaches that employ differential surface representations and categorize them according to the particular representation and local transformation handling.

The first use of differential coordinates for mesh editing was sketched by Alexa in [2]. He suggested using the original surface Laplacians with soft modeling constraints in (21). Since no appropriate local transformations were computed, this approach was suitable for editing smooth surfaces with no features or for performing deformations that almost do not involve rotations.

In 2004, Lipman et al. [41] proposed adding local rotation estimation to the simple Laplacian editing paradigm, computing it from the naive solution of [2]. They have shown that smoothing the estimated transformations by using larger neighborhoods with special weighted averaging may significantly improve the results, although at larger computational cost. Still, note that this two-step deformation process only requires two solutions by back-substitution (plus intermediate transformation computations), since the system matrix (21) remains the same and can be prefactored. The approach works well for relatively smooth surfaces with no largely protruding features; otherwise, the underlying assumption that the initial guess by naive Laplacian editing provides a good rotation guess no longer holds, and the rotation estimation fails. In particular, the approach may have difficulty with features that cannot be described as a height field over the base smooth surface.

Botsch et al. [14] proposed a conceptually similar technique: They estimate the local rotations from a base surface $\mathcal{B}$ and its deformed version $\mathcal{B}'$ (see Fig. 3 and Section 5.5); they then apply these rotations to the gradients of the input mesh to reconstruct the final result by using the Poisson framework (15).

Sorkine et al. [60] proposed using the Laplacian representation coupled with implicit transformation optimization derived from 1-rings. To eliminate isotropic scaling, they rescale the Laplacians of the deformed surface back to their original length and solve (21). This technique can handle more complex surfaces with large features. It is limited, however, in the allowed rotation range because the linearized approximation of local rotations is only valid for small angles. In practice, rotations of up to $\pi/2$ can be well performed [57]. For larger rotations, several steps of the technique should be applied to break the large rotation into smaller ones.

The method of Sorkine et al. [60] was applied to the manipulation of triangulated 2D shapes by Igarashi et al. [33]. They used implicit transformation optimization. Note that in 2D, similarity transformations can be exactly linearly parameterized:

$$\mathbf{S} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}.$$

In a second step, Igarashi et al. remove the unwanted uniform scaling from the local transformations and resolve for the vertex positions by using edge equations, as in (23). The technique is very effective for 2D shape editing, thanks to the exact rotation formulation and the meshing of the interior of the shape.

The idea of combining the Laplacian representation with implicit transformation optimization was further developed by Fu et al. [23]. They propose a hybrid approach that combines implicit optimization with two-step local transformation estimation. In the first stage, Laplacian editing is performed with implicit transformations, which are *not* constrained to linearized similarity transformations but instead are allowed to be any affine transformations $\mathbf{T}_i = \mathbf{U}_i'\mathbf{U}_i^+$, where $\mathbf{U}_i$ are the 1-ring vectors of vertex v_i. In addition, the local transformations are asked to be locally smooth, which is expressed by additional quadratic terms in the deformation energy $\|\mathbf{T}_i - \mathbf{T}_j\|^2$ for neighboring vertices v_i, v_j. The resulting deformed surface is then used as an initial guess to estimate the actual local transformations: those are orthogonalized, and Laplacian editing (21) is applied. This approach enables larger rotations than [60], but it requires tweaking the relative weighting of local transformation smoothness terms. Moreover, the formulation of implicit transformations may be ill defined for flat 1-rings, in which case a perturbation is required.

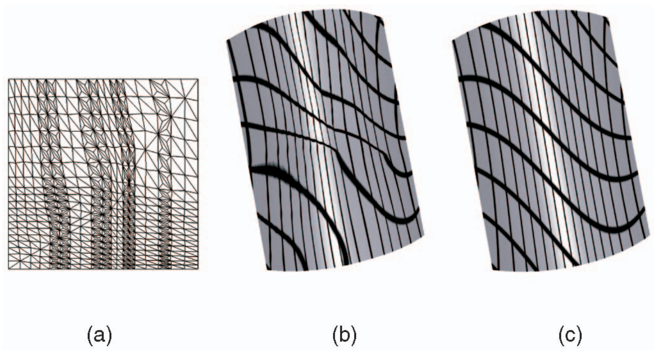

Fig. 8. (a) A 100,000-triangle version of the mesh (a) from [27] was bent to compare (b) the original Laplacian editing formulation $\mathbf{L}^T\mathbf{L}\mathbf{p} = \mathbf{L}^T\delta$ to (c) the correct one $\mathbf{L}^2\mathbf{p} = \mathbf{L}\delta$. Both (b) and (c) use the cotangent Laplacian.

It is worth noting that the above approaches used a slightly erroneous version of the discrete energy (20), since they omitted the area weights A_i, which correspond to the discretization of the L_2 product on the mesh [67]. This leads to normal equations of the form

$$\mathbf{L}^T\mathbf{L}\mathbf{p}' = \mathbf{L}^T\delta,$$

which differ from the correctly discretized (21). Such formulation may lead to problems on irregular meshes, as demonstrated in Fig. 8.

Laplacian editing was further used for a sketch-based editing system [49] and volume graph deformations [73]. Nealen at al. [49] employed implicit transformation propagation and proposed using sketched curves on the surface as handles and deformation constraints, which leads to an intuitive silhouette and feature editing tool. In the classical handle metaphor, the position of the handle is directly manipulated by the user and, thus, hard positional constraints are preferred. In a sketch-based system, soft constraints are actually advantageous, since they allow the user to place imprecise strokes that are only meant to hint at the desired shape but not specify it exactly. Thus, Nealen et al. allowed varying the weight on the sketched positional constraints to achieve rough sketching (small weights) or carefully drawn sketching (high weights; see an example in Fig. 9). Zhou et al. [73] proposed similar skeleton-curve deformation constraints for character animation, combined with Laplacian editing of a volumetric graph (they used a variant of geodesic transformation propagation). They augment the surface mesh with an inner

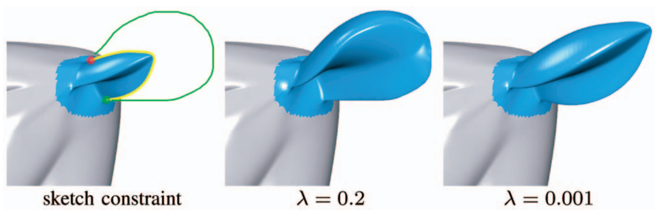

Fig. 9. Varying the weighting of soft positional constraints may be beneficial, for example, in a sketch-based interface [49]. Here, the handle is the silhouette curve, and the positional constraints are sketched in green. A small relative weight λ leads to a rough approximation of the sketch, preserving the surface details along the silhouette, whereas a larger weight makes the system follow the sketched curve more precisely.

grid of vertices (which should be coarser than the surface mesh for complexity reasons). Performing Laplacian editing on such volumetric graph creates connections between distant points on the surface and thus tends to better preserve the volume of the shape.

Yu et al. [71] proposed the gradient-based representation for mesh editing, combined with geodesic propagation of local transformations. Zayer et al. [72] replaced the geodesic propagation by harmonic interpolation and showed that this leads to smoother distributed local transformations and, thus, better results. Popa et al. [52] generalized the harmonic propagation to material-dependent transformation assignment. In contrast to previously cited techniques, all these methods only work when an appropriate handle transformation is specified in addition to translation (they are translation insensitive). It is worth noting that deformation gradients, closely related to the gradient-based representation, were used for deformation transfer of one deforming mesh sequence onto another by Sumner and Popović [61].

Lipman et al. [42] developed the frame-based representation and used it for surface editing and interpolation. This technique employs a rigid-invariant representation, where the local transformations are found explicitly by optimization (22). The deformation constraints may include very large rotations (up to π), and the details remain preserved. The limitation, however, is again translation insensitivity, since solving for the frames is decoupled from the positional constraints; thus, explicit rotational constraints for handle frames must be specified.

Note that all the differential deformation approaches require solving global linear sparse systems involving symmetric positive definite matrices and, thus, can benefit from fast Cholesky factorization in a preprocess and interactive back-substitution, as described in Section 2.3. Nonetheless, it is worth noting that, recently, Shi et al. [54] developed a multiresolution solver specifically tailored for solving Poisson systems, which may be useful in scenarios where the ROI changes frequently or the Cholesky factor is too large to fit into memory. They also proposed a modification of the frame-based editing approach of Lipman et al. [42] to demonstrate the abilities of their solver: Instead of solving (22), the frames of the deformed surface are computed by harmonic interpolation of the handle transformation, whereas the geometry reconstruction step (23) remains the same. It can be shown that when all handle constraints involve rotations about the same axis, this framework produces optimal results in terms of curvature preservation [39].

4 Comparison and Discussion

In this section, we compare the different mesh editing techniques described in Sections 2 and 3. Since it is hard to evaluate and compare the techniques solely based on the (differing) examples given in the original papers, we perform exactly the same deformations by using a representative subset of the described techniques. Notice that our goal is *not* to show the best possible results that each method can produce, since these images can be found in the original publications. Instead, we rather want to show under which circumstances each individual method fails. Hence, in Fig. 10, we picked extreme deformations that identify the respective limitations of the different techniques. For comparison, we show the results of the nonlinear surface deformation PriMo [12], which does not suffer from linearization artifacts.

The first technique that we examine is the variational bending energy minimization [10] in combination with the multiresolution technique based on deformation transfer [14]. This approach works fine for pure translations; that is, it yields a smooth deformation and locally rotates the geometric details. However, due to the linearization from (1) to (2), this method has problems with large rotations, which can be seen in the bend and twist examples. Notice that for these two examples, anisotropic deformations were used. After a principal component analysis, the model is anisotropically scaled along its principal axes to have uniform variation and the cotangent weights (7) are derived from the scaled coordinates, similar to [10]. The cactus model is difficult because of its strongly protruding arms. The default base surface $\mathcal{B}$, as computed by minimizing curvature energy, has degenerate triangles in these regions such that no multiresolution hierarchy was used for this example.

The gradient-based Poisson editing [71], [72] updates the surface gradients by using the gradient of the deformation, that is, its rotation and scale/shear components. Consequently, the technique works very well for rotations. However, as mentioned in Section 3.2, the explicit propagation of local rotations is translation insensitive such that the plane example is neither smooth nor detail preserving.

In contrast, the Laplacian surface editing [60] implicitly determines the per-vertex rotations and hence works similarly well for translations and rotations. Its main drawback is the required linearization of rotations, which yields artifacts for large local rotations. Notice that although the original paper employed the uniform Laplacian discretization, our examples were done using the cotangent weights (see also Section 5.1).

The rotation-invariant coordinates [42] solve a linear system to preserve the relative orientation of the local frames, which works very well for rotations and does not have problems with protruding features like in the cactus example. However, since this linear system does not consider positional constraints, this method is also translation insensitive. In addition, the linear system for reconstructing the positions from local frames corresponds to a uniform Laplacian, which causes the asymmetries for the regular tessellation of the bumpy plane.

From those examples, one can derive the following guidelines for picking the "right" deformation technique for a specific application scenario.

In technical CAD-like engineering applications, the required shape deformations are typically rather small since, in many cases, an existing prototype only has to be

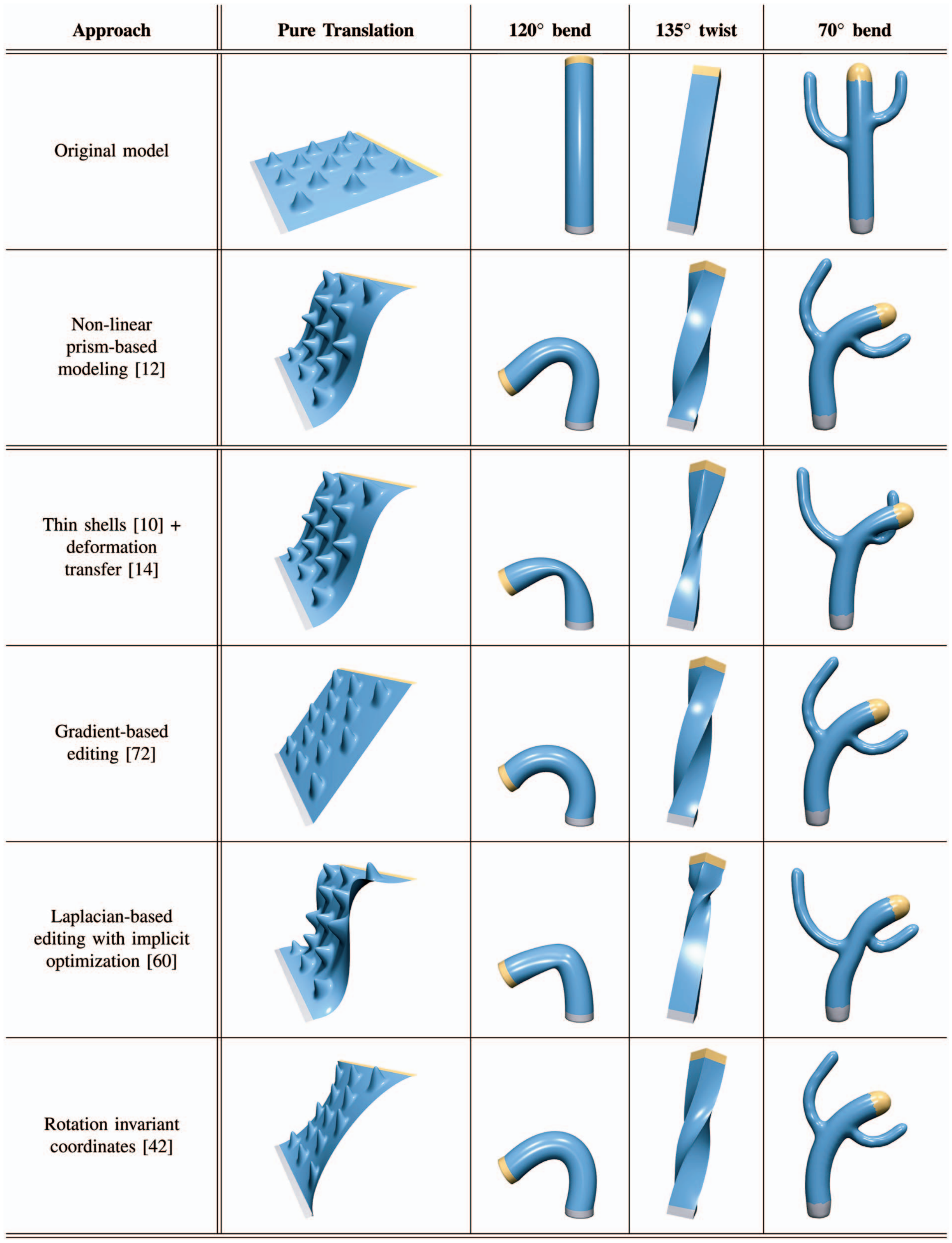
Approach
Pure Translation
120° bend
135° twist
70° bend
Original model
Non-linear prism-based modeling [12]
Thin shells [10] + deformation transfer [14]
Gradient-based editing [72]
Laplacian-based editing with implicit optimization [60]
Rotation invariant coordinates [42]

Fig. 10. The extreme examples shown in this comparison matrix were particularly chosen to reveal the limitations of the respective deformation approaches. The respective strengths and weaknesses of the depicted techniques, as well as the reasons of the artifacts, are discussed in Section 4.

adjusted slightly, but they have high requirements on surface fairness, boundary continuity, and the precise control thereof. For such problems, a linearized shell model like [10] was shown to be well suited.

In contrast, applications like character animation mostly involve (possibly large) rotations of limps around bends and joints. Here, methods based on differential coordinates clearly are the better choice. Moreover, the required

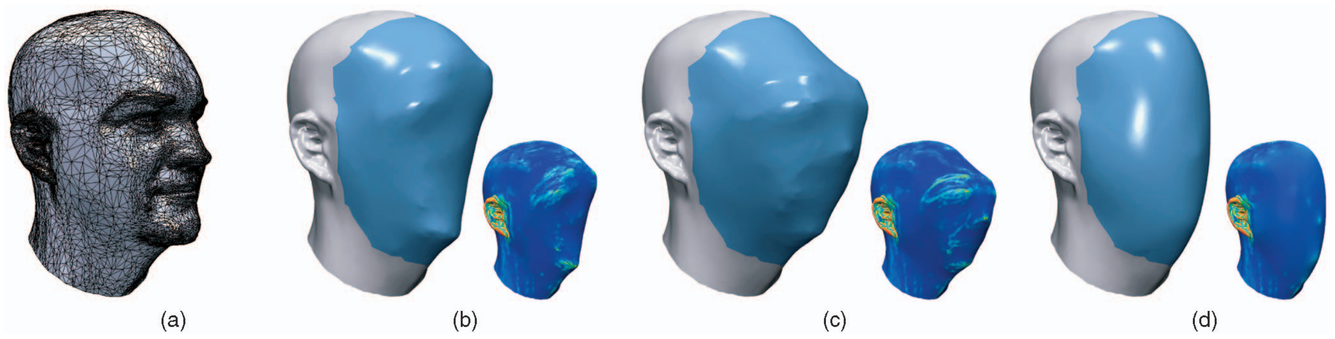

Fig. 11. Different Laplace-Beltrami discretizations are evaluated by minimizing the thin-plate energy of (a) the irregular mesh by solving the Euler-Lagrange equation $\Delta_{\mathcal{S}}^2 \mathbf{p} = \mathbf{0}$. Both (b) the uniform Laplacian and (c) the cotangent Laplacian without the area term yield artifacts in regions of high vertex density. (d) The cotangent discretization including the per-vertex normalization clearly gives the best results. The small images show the respective mean curvatures.

rotations might be available, for example, from a sketching interface [49], [73] or a motion capture system [54].

Applications that require both large-scale translation and rotations are problematic for all linear approaches. In this case, one can either employ a more complex, nonlinear technique or split up large deformations into a sequence of smaller ones. Although nonlinear techniques are computationally and implementationwise more involved, splitting up deformations or providing a denser set of constraints complicates the user interaction. With the rapidly increasing computational power of today's computers, nonlinear methods have become much more tractable, which already has led to a first set of nonlinear yet interactive mesh deformation approaches [4], [12], [31], [36], [53], [62], [66].

5 Deformation FAQ

After describing, comparing, and discussing the various shape editing techniques in Sections 2 and 3, we finally want to answer a set of questions most frequently asked in the context of mesh-based surface deformations.

5.1 What Is the Influence of the Laplacian Discretization?

Most of the approaches described in Sections 2 and 3 derive the deformed surface by solving a Laplacian or bi-Laplacian linear system. Hence, they all require a discretization of the Laplacian operator and their results strongly depend on this choice. There exist several variations of the weights used in the typically employed Laplacian discretization (6). The uniform Laplacian, employed in [34], [41], [60], and [63], for instance, uses the weights

$$w_{ij} = 1 , \quad w_i = \frac{1}{\sum_j w_{ij}} .$$

Since this discretization takes neither edge lengths nor angles into account, it cannot provide a good approximation for irregular meshes. Better results can be achieved by

$$w_{ij} = \frac{1}{2}\left(\cot \alpha_{ij} + \cot \beta_{ij}\right) , \quad w_i = 1,$$

which now considers angles but not varying vertex densities [69], [71]. The best results are obtained by including the per-vertex normalization weights (see Section 2.2):

$$w_{ij} = \frac{1}{2}\left(\cot \alpha_{ij} + \cot \beta_{ij}\right) , \quad w_i = \frac{1}{A_i},$$

as proposed in [18], [45], and [51] and employed, for instance, in [10] and [11]. A qualitative comparison of the three discretizations is given in Fig. 11. In this example, curvature energies are minimized by solving $\Delta_{\mathcal{S}}^2 \mathbf{p} = \mathbf{0}$, since smooth surfaces are visually easier to evaluate than smooth deformations. A more detailed analysis of different discretizations, with a focus on their convergence properties, can be found in [27] and [70].

Although the cotangent discretization clearly gives the best results, it can also lead to numerical problems in the presence of near-degenerate triangles, since the cotangent values degenerate and the resulting matrices become singular. In this case, the degenerate triangles would have to be eliminated [8] in a preprocess. Alternatively, the whole base surface $\mathcal{B}$ could be remeshed isotropically, as proposed in [11].

5.2 What Is the Difference between the Thin-Plate Energy $\int \kappa_1^2 + \kappa_2^2$ and the Mean Curvature Energy $\int H^2$?

With the mean curvature $H = \kappa_1 + \kappa_2$ and Gaussian curvature $K = \kappa_1 \kappa_2$, we have

$$\int_{\mathcal{S}} H^2 \,\mathrm{d}A \;=\; \int_{\mathcal{S}} \left(\kappa_1^2 + \kappa_2^2\right) \mathrm{d}A \;+\; 2 \int_{\mathcal{S}} K \,\mathrm{d}A,$$

i.e., the two energies basically differ in the integral $\int K$, which, by the Gauss-Bonnet theorem, only depends on the (fixed) Dirichlet boundary constraints on $\partial\,\Omega$ and therefore stays constant [19]. Hence, the minimizers of the two energies are equivalent for identical Dirichlet boundary constraints. This also holds for the linearized energies, which are

$$\begin{aligned} \int_{\mathcal{S}} \kappa_1^2 + \kappa_2^2 \,\mathrm{d}A &\approx \int_{\mathcal{S}} \|\mathbf{p}_{uu}\|^2 + 2\|\mathbf{p}_{uv}\|^2 + \|\mathbf{p}_{vv}\|^2 \mathrm{d}u\mathrm{d}v, \\ \int_{\mathcal{S}} H^2 \,\mathrm{d}A &\approx \int_{\mathcal{S}} \|\mathbf{p}_{uu}\|^2 + 2\mathbf{p}_{uu}^T \mathbf{p}_{vv} + \|\mathbf{p}_{vv}\|^2 \mathrm{d}u\mathrm{d}v. \end{aligned}$$

Variational calculus yields the identical Euler-Lagrange equation $\Delta^2 \mathbf{p} = \mathbf{0}$ for both linearized energies, and its discretization (and symmetrization) leads to $\mathbf{L}_s \mathbf{M}^{-1} \mathbf{L}_s \mathbf{p} = \mathbf{0}$ to be solved for its minimizer surface (see Section 2.3). Even when discretizing the mean curvature energy instead of the above Euler-Lagrange equations, one arrives at the same linear system [67].

5.3 What Is the Difference between Hard Constraints and Soft Least Squares Constraints?

As introduced in Section 2, the first n' vertices $(v_1, \ldots, v_{n'})$ are considered free, and the last $k = n - n'$ vertices $(v_{n'+1}, \ldots, v_n)$ are constrained by prescribing positions $\mathbf{c}_i$ or displacements $\mathbf{d}_i = \mathbf{c}_i - \mathbf{p}_i$.

Most mesh editing approaches consider those constraints as *hard* constraints. For instance, solving a bi-Laplacian system, as described in Section 2, gives the initial linear system

$$\begin{pmatrix} & \mathbf{L}^2 & \\ \mathbf{0} & & \mathbf{I}_k \end{pmatrix} \begin{pmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_n \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{c}_{n'+1} - \mathbf{p}_{n'+1} \\ \vdots \\ \mathbf{c}_n - \mathbf{p}_n \end{pmatrix},$$

with $\mathbf{L}^2 \in \mathbb{R}^{n \times n}$ and $\mathbf{I}_k$ being the $k \times k$ identity matrix. Eliminating rows and columns corresponding to the constrained vertices by bringing them to the right-hand side then yields the upper left $n' \times n'$ submatrix as the linear system to be solved for the displacements $\mathbf{d}_1, \ldots, \mathbf{d}_{n'}$.

In contrast, the Laplacian editing papers [41], [49], [60] handle constraints as *soft* constraints by adding them to the energy in the form

$$E(\mathbf{p}') = \sum_{i=1}^{n} \|\Delta_{\mathcal{S}}(\mathbf{p}'_i) - \delta'_i\|^2 + \lambda \cdot \sum_{j=n'+1}^{n} \|\mathbf{p}'_j - \mathbf{c}_j\|^2 .$$

The minimum of this energy can be found by solving the overdetermined $(n + k) \times n$ system:

$$\begin{pmatrix} & \mathbf{L} & \\ \mathbf{0} & & \lambda\mathbf{I}_k \end{pmatrix} \begin{pmatrix} \mathbf{p}'_1 \\ \vdots \\ \mathbf{p}'_n \end{pmatrix} = \begin{pmatrix} \delta'_1 \\ \vdots \\ \delta'_n \\ \lambda\mathbf{c}_{n'+1} \\ \vdots \\ \lambda\mathbf{c}_n \end{pmatrix}$$

in the least squares sense. This requires solving the normal equations, which leads to the $n \times n$ system:

$$\left[\mathbf{L}^T\mathbf{L} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \lambda^2\mathbf{I}_k \end{pmatrix}\right] \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{pmatrix} = \mathbf{L}^T \begin{pmatrix} \delta'_1 \\ \vdots \\ \delta'_n \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \lambda^2\mathbf{c}_{n'+1} \\ \vdots \\ \lambda^2\mathbf{c}_n \end{pmatrix}.$$

In order to get (close to) the interpolation of the constraints $\mathbf{c}_i$, one has to choose a sufficiently large weight λ, which, unfortunately, depends on the geometric position of the $\mathbf{c}_i$, as well as on the relative number of constraints k/n. Moreover, since the condition number of the above matrix grows linearly with λ, a higher weight can cause numerical problems.

However, with growing λ, the solution of the above system approaches the solution of the $n' \times n'$ system:

$$\mathbf{L}^T\mathbf{L} \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_{n'} \end{pmatrix} = \mathbf{L}^T \begin{pmatrix} \delta'_1 \\ \vdots \\ \delta'_{n'} \end{pmatrix},$$

with *hard* constraints. Since its condition number is much better, it is therefore advisable to solve the latter system instead when exact interpolation of the constraints is required. See the effect of varying λ in Fig. 9.

5.4 What Is the Relation of the Bending Energy Minimization and the Laplacian-Based Differential Deformation?

In the following, we focus on the surface deformation and neglect detail preservation techniques such as the multiresolution decomposition in Section 2 and the rotations of Laplacians in Section 3. As described in Section 2, the variational minimization of the bending energy

$$\int_{\Omega} \|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \,\mathrm{d}u\mathrm{d}v$$

leads to the Euler-Lagrange equation

$$\Delta^2\mathbf{d} = \mathbf{0}\,.$$

Similarly, the Laplacian editing energy

$$\int_{\Omega} \|\Delta\mathbf{p}' - \delta\|^2 \,\mathrm{d}u\mathrm{d}v$$

introduced in Section 3 yields

$$\Delta^2\mathbf{p}' = \Delta\delta\,.$$

From $\mathbf{p}' = \mathbf{p} + \mathbf{d}$ and $\delta = \Delta\mathbf{p}$, we can immediately see that the two Euler-Lagrange equations are equivalent, and so are their corresponding linear systems $\mathbf{L}^2\mathbf{d} = \mathbf{0}$ and $\mathbf{L}^2\mathbf{p}' = \mathbf{L}\delta$. Notice that this is not true for the original formulation presented in [41], [60], since there, the slightly incorrect system $\mathbf{L}^T\mathbf{L}\mathbf{p}' = \mathbf{L}^T\delta$ was solved (see Section 3.3).

The basic variational bending energy minimization and Laplacian-based surface deformation can, therefore, be considered equivalent. They differ in the way that they are extended to preserve fine-scale details, i.e., finding the local rotations of the geometric details. Although all multiresolution approaches derive those rotations from the deformation of the low-frequency base surface (Section 2.4), there are several approaches to rotate the differential coordinates (Section 3.2).

5.5 What Is the Relation of Gradient-Based Deformation and Deformation Transfer?

In [61], Sumner and Popović transfer the deformation $\mathcal{S} \mapsto \mathcal{S}'$ for a given source mesh $\mathcal{S}$ and its deformed version $\mathcal{S}'$ onto a target mesh $\mathcal{T}$. This yields a deformed mesh $\mathcal{T}'$ such that the two deformations $\mathcal{S} \mapsto \mathcal{S}'$ and $\mathcal{T} \mapsto \mathcal{T}'$ are as similar as possible.

They add a fourth point to each triangle t_i, turning the triangle into a tetrahedron such that these four points in $\mathcal{S}$ and $\mathcal{S}'$ uniquely determine the affine transformation $\mathbf{x} \mapsto \mathbf{S}_i\mathbf{x} + \mathbf{t}_i$. They then consider the gradient of this affine mapping (so-called *deformation gradient*), which is the 3×3 matrix $\mathbf{S}_i$ containing the rotation and scale/shear part. Finally, new vertex positions $\mathbf{p}'_i \in \mathcal{T}'$ are found such that the resulting deformation gradients $\mathbf{T}_i$ for $\mathcal{T}$ are close to the given $\mathbf{S}_i$, which leads to the area-weighted least squares system

$$\tilde{\mathbf{G}}^T\mathbf{M}\tilde{\mathbf{G}} \begin{pmatrix} \mathbf{p}'^T_1 \\ \vdots \\ \mathbf{p}'^T_{\tilde{n}} \end{pmatrix} = \tilde{\mathbf{G}}^T\mathbf{M} \begin{pmatrix} \mathbf{S}^T_1 \\ \vdots \\ \mathbf{S}^T_m \end{pmatrix},$$

where $\tilde{n} = n + m \approx 3n$ is the number of vertices including the additional fourth points and $\tilde{\mathbf{G}}$ is the $3m \times \tilde{n}$ matrix that computes the deformation gradients from the vertex positions.

In this context, the gradient-based deformation [71] is similar, but, here, the user directly prescribes the local rotations $\mathbf{S}_i$, which are then applied to the gradients $\mathbf{G}_i \in \mathbb{R}^{3\times 3}$ of the original mesh $\mathcal{T}$, resulting in $\mathbf{G}'_i$. In order to find the mesh $\mathcal{T}'$ that has the desired gradients $\mathbf{G}'_i$, the Poisson system

$$\mathbf{G}^T\mathbf{M}\mathbf{G} \begin{pmatrix} \mathbf{p}'^T_1 \\ \vdots \\ \mathbf{p}'^T_n \end{pmatrix} = \mathbf{G}^T\mathbf{M} \begin{pmatrix} \mathbf{G}'_1 \\ \vdots \\ \mathbf{G}'_m \end{pmatrix}$$

is solved, as described in Section 3.

It was shown in [14] that one can safely remove the fourth points from the first system, which reduces the size of the system from $\tilde{n} \times \tilde{n}$ to $n \times n$. After that reduction, the matrices $\mathbf{G}$ and $\tilde{\mathbf{G}}$—and, hence, the whole linear systems—can be shown to be equal. Hence, deformation transfer can also be considered as a special case of Poisson editing, where the local per-triangle transformations are determined from $\mathcal{S}$ and $\mathcal{S}'$.

6 Conclusions

In this survey, we attempted to give a systematic description and classification of the plethora of surface editing methods that can be generally seen as linear variational techniques. Our goal was to first explain the original motivation behind these techniques, which comes from continuous formulations and is closely related to physically based energies and classical differential geometry. Then, we showed how the different methods simplify and discretize these settings in order to achieve interactive and robust mesh deformation methods. Finally, we performed practical comparison of several representative methods to reveal the characteristic strengths and weaknesses of each approach in extreme deformation cases. We hope that our qualitative description and practical illustrations will help the readers understand the ideas behind these methods and also choose the right method for each particular editing scenario.

We focused on linear variational methods since they comprise a large body of work over the recent years, yet they have not been surveyed in an elaborated and comparative manner. In addition, this group of methods has gained high visibility in computer graphics research, as evident by the number of citations. This popularity is owed to the robustness and ease of implementation of these approaches, especially thanks to the availability of advanced sparse linear solvers. One obvious conclusion of this survey, however, is that there is no perfect technique that would work satisfactory in every case. Apart from the fact that a "perfect" result may be a subjective and application-dependent notion, all the reviewed methods share the same property: for the sake of speed and robustness, they linearize the inherently nonlinear deformation problem. The various types of machinery, which are meant to mask the linearization errors, work in some scenarios but fail in others, as we demonstrated. As computing resources become much faster, and previously infeasible numerical methods become tractable, there is now room for nonlinear methods and optimizations to be explored in interactive applications.

In light of the above, we felt that this is an appropriate point in time to summarize the linear variational deformation methods. We are confident that these techniques are yet to conquer the commercial modeling applications and that researchwise, there are yet many areas where they can be incorporated and explored further. Moreover, we anticipate further development of nonlinear deformation techniques, exploiting the knowledge and experience gained from the linear methods. After all, each general problem is solved by iterating and refining linear approximations.

Acknowledgments

Olga Sorkine is supported by the Alexander von Humboldt Foundation. The authors wish to thank Leif Kobbelt and Daniel Cohen-Or for encouraging them to prepare this survey and for coauthoring numerous papers recited here. The authors are also grateful to them, as well as to Marc Alexa, Markus Gross, Denis Zorin, Max Wardetzky, and Klaus Hildebrandt, for the various discussions that helped improve this manuscript. The authors also thank the anonymous reviewers for their valuable comments and suggestions. The results in Fig. 7 are courtesy of Tiberiu Popa and Alla Sheffer.

References

[1] B. Aksoylu, A. Khodakovsky, and P. Schröder, "Multilevel Solvers for Unstructured Surface Meshes," *SIAM J. Scientific Computing,* vol. 26, no. 4, pp. 1146-1165, 2005.

[2] M. Alexa, "Local Control for Mesh Morphing," *Proc. Int'l Conf. Shape Modeling and Applications (SMI '01),* pp. 209-215, 2001.

[3] M. Alexa, "Differential Coordinates for Local Mesh Morphing and Deformation," *The Visual Computer,* vol. 19, no. 2, pp. 105-114, 2003.

[4] O.K.-C. Au, C.-L. Tai, L. Liu, and H. Fu, "Dual Laplacian Editing for Meshes," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 3, pp. 386-395, May-June 2006.

[5] K.-J. Bathe, *Finite Element Procedures.* Prentice Hall, 1995.

[6] D. Bechmann, "Space Deformation Models Survey," *Computers and Graphics,* vol. 18, no. 4, pp. 571-586, 1994.

[7] M. Botsch, D. Bommes, and L. Kobbelt, "Efficient Linear System Solvers for Mesh Processing," *IMA Math. of Surfaces XI,* pp. 62-83, 2005.

[8] M. Botsch and L. Kobbelt, "A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes," *Proc. Sixth Int'l Fall Workshop Vision, Modeling, and Visualization (VMV '01),* pp. 402-410, 2001.

[9] M. Botsch and L. Kobbelt, "Multiresolution Surface Representation Based on Displacement Volumes," *Computer Graphics Forum (Proc. Eurographics),* vol. 22, no. 3, pp. 483-491, 2003.

[10] M. Botsch and L. Kobbelt, "An Intuitive Framework for Real-Time Freeform Modeling," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 23, no. 3, pp. 630-634, 2004.

[11] M. Botsch and L. Kobbelt, "A Remeshing Approach to Multiresolution Modeling," *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing,* pp. 189-196, 2004.

[12] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt, "PriMo: Coupled Prisms for Intuitive Surface Modeling," *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing,* pp. 11-20, 2006.

[13] M. Botsch, M. Pauly, C. Rössl, S. Bischoff, and L. Kobbelt, "Geometric Modeling Based on Triangle Meshes," *Eurographics Course Notes,* 2006.

[14] M. Botsch, R. Sumner, M. Pauly, and M. Gross, "Deformation Transfer for Detail-Preserving Surface Editing," *Proc. 11th Int'l Fall Workshop Vision, Modeling, and Visualization (VMV '06),* pp. 357-364, 2006.

[15] G. Celniker and D. Gossard, "Deformable Curve and Surface Finite-Elements for Free-Form Shape Design," *Proc. ACM SIGGRAPH '91,* pp. 257-266, 1991.

[16] F. Cirak, M. Ortiz, and P. Schröder, "Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis," *Int'l J. Numerical Methods in Eng.,* vol. 47, no. 12, pp. 2039-2072, 2000.

[17] F. Cirak, M. Scott, P. Schröder, M. Ortiz, and E. Antonsson, "Integrated Modeling, Finite-Element Analysis, and Design for Thin-Shell Structures Using Subdivision," *Computer-Aided Design,* vol. 34, no. 2, pp. 137-148, 2002.

[18] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr, "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow," *Proc. ACM SIGGRAPH '99,* pp. 317-324, 1999.

[19] M.P. do Carmo, *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[20] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. ACM SIGGRAPH '95,* pp. 173-182, 1995.

[21] R. Fattal, D. Lischinski, and M. Werman, "Gradient Domain High Dynamic Range Compression," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 21, no. 3, pp. 249-256, 2002.

[22] D. Forsey and R. Bartels, "Hierarchical B-spline refinement," *Proc. ACM SIGGRAPH,* pp. 205-212, 1988.

[23] H. Fu, O.K.-C. Au, and C.-L. Tai, "Effective Derivation of Similarity Transformations for Implicit Laplacian Mesh Editing," *Computer Graphics Forum,* vol. 26, no. 1, pp. 34-45, 2007.

[24] M. Garland, "Multiresolution Modeling: Survey & Future Opportunities," *Eurographics State of the Art Report,* 1999.

[25] G.H. Golub and C.F. Van Loan, *Matrix Computations,* third ed. Johns Hopkins Univ. Press, 1996.

[26] G. Greiner and J. Loos, "Data Dependent Thin Plate Energy and Its Use in Interactive Surface Modeling," *Computer Graphics Forum (Proc. Eurographics),* vol. 15, no. 3, pp. 175-185, 1996.

[27] E. Grinspun, Y. Gingold, J. Reisman, and D. Zorin, "Computing Discrete Shape Operators on General Meshes," *Computer Graphics Forum (Proc. Eurographics),* vol. 25, no. 3, pp. 547-556, 2006.

[28] H.W. Guggenheimer, *Differential Geometry.* McGraw-Hill, 1963.

[29] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution Signal Processing for Meshes," *Proc. ACM SIGGRAPH,* pp. 325-334, 1999.

[30] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder, "Normal Meshes," *Proc. ACM SIGGRAPH,* pp. 95-102, 2000.

[31] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S. Teng, H. Bao, B. Guo, and H.-Y. Shum, "Subspace Gradient Domain Mesh Deformation," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 25, no. 3, pp. 1126-1134, 2006.

[32] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Prentice Hall, 1987.

[33] T. Igarashi, T. Moscovich, and J.F. Hughes, "As-Rigid-As-Possible Shape Manipulation," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 24, no. 3, pp. 1134-1141, 2005.

[34] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive Multi-Resolution Modeling on Arbitrary Meshes," *Proc. ACM SIGGRAPH,* pp. 105-114, 1998.

[35] L. Kobbelt, J. Vorsatz, and H.-P. Seidel, "Multiresolution Hierarchies on Unstructured Triangle Meshes," *Computational Geometry: Theory and Applications,* vol. 14, pp. 5-24, 1999.

[36] V. Kraevoy and A. Sheffer, "Mean-Value Geometry Encoding," *Int'l J. Shape Modeling,* vol. 12, no. 1, pp. 29-46, 2006.

[37] A. Lee, H. Moreton, and H. Hoppe, "Displaced Subdivision Surfaces," *Proc. ACM SIGGRAPH,* pp. 85-94, 2000.

[38] S. Lee, "Interactive Multiresolution Editing of Arbitrary Meshes," *Computer Graphics Forum,* vol. 18, no. 3, pp. 73-82, 1999.

[39] Y. Lipman, D. Cohen-Or, R. Gal, and D. Levin, "Volume and Shape Preservation via Moving Frame Manipulation," *ACM Trans. Graphics,* vol. 26, no. 1, 2007.

[40] Y. Lipman, O. Sorkine, M. Alexa, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Laplacian Framework for Interactive Mesh Editing," *Int'l J. Shape Modeling,* vol. 11, no. 1, pp. 43-62, 2005.

[41] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Differential Coordinates for Interactive Mesh Editing," *Proc. Int'l Conf. Shape Modeling (SMI '04),* pp. 181-190, 2004.

[42] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear Rotation-Invariant Coordinates for Meshes," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 24, no. 3, pp. 479-487, 2005.

[43] M. Marinov, M. Botsch, and L. Kobbelt, "GPU-Based Multiresolution Deformation Using Approximate Normal Field Reconstruction," *J. Graphics Tools,* vol. 12, no. 1, pp. 27-46, 2007.

[44] M. Marinov and L. Kobbelt, "Automatic Generation of Structure Preserving Multiresolution Models," *Computer Graphics Forum (Proc. Eurographics),* vol. 24, no. 3, pp. 479-486, 2005.

[45] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds," *Visualization and Math. III,* H.-C. Hege and K. Polthier, eds., pp. 35-57, 2003.

[46] T. Milliron, R.J. Jensen, R. Barzel, and A. Finkelstein, "A Framework for Geometric Warps and Deformations," *ACM Trans. Graphics,* vol. 21, no. 1, pp. 20-51, 2002.

[47] H.P. Moreton and C.H. Séquin, "Functional Optimization for Fair Surface Design," *Proc. SIGGRAPH,* pp. 167-176, 1992.

[48] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically Based Deformable Models in Computer Graphics," *Computer Graphics Forum,* vol. 25, no. 4, pp. 809-836, 2006.

[49] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A Sketch-Based Interface for Detail-Preserving Mesh Editing," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 24, no. 3, pp. 1142-1147, 2005.

[50] P. Pérez, M. Gangnet, and A. Blake, "Poisson Image Editing," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 22, no. 3, pp. 313-318, 2003.

[51] U. Pinkall and K. Polthier, "Computing Discrete Minimal Surfaces and Their Conjugates," *Experimental Math.,* vol. 2, no. 1, pp. 15-36, 1993.

[52] T. Popa, D. Julius, and A. Sheffer, "Material-Aware Mesh Deformations," *Proc. IEEE Int'l Conf. Shape Modeling and Applications (SMI '06).* pp. 141-152, 2006.

[53] A. Sheffer and V. Kraevoy, "Pyramid Coordinates for Morphing and Deformation," *Proc. Second Int'l Symp. 3D Data Processing, Visualization, and Transmission (3DPVT '04),* pp. 68-75, 2004.

[54] L. Shi, Y. Yu, N. Bell, and W.-W. Feng, "A Fast Multigrid Algorithm for Mesh Deformation," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 25, no. 3, pp. 1108-1117, 2006.

[55] K. Shoemake and T. Duff, "Matrix Animation and Polar Decomposition," *Proc. Conf. Graphics Interface,* pp. 258-264, 1992.

[56] O. Sorkine, "Differential Representations for Mesh Processing," *Computer Graphics Forum,* vol. 25, no. 4, pp. 789-807, 2006.

[57] O. Sorkine, "Laplacian Mesh Processing," PhD dissertation, School of Computer Science, Tel Aviv Univ., 2006.

[58] O. Sorkine and D. Cohen-Or, "Least-Squares Meshes," *Proc. IEEE Int'l Conf. Shape Modeling and Applications (SMI '04),* pp. 191-199, 2004.

[59] O. Sorkine, D. Cohen-Or, and S. Toledo, "High-Pass Quantization for Mesh Encoding," *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing,* pp. 42-51, 2003.

[60] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian Surface Editing," *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing,* pp. 179-188, 2004.

[61] R.W. Sumner and J. Popović, "Deformation Transfer for Triangle Meshes," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 23, no. 3, pp. 399-405, 2004.

[62] R.W. Sumner, M. Zwicker, and C. Gotsman, "Mesh-Based Inverse Kinematics," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 24, no. 3, pp. 488-495, 2005.

[63] G. Taubin, "A Signal Processing Approach to Fair Surface Design," *Proc. ACM SIGGRAPH '95,* pp. 351-358, 1995.

[64] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," *Proc. ACM SIGGRAPH '87,* pp. 205-214, 1987.

[65] B. Thomaszewski, M. Wacker, and W. Strasser, "A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 107-116, 2006.

[66] W. von Funck, H. Theisel, and H.-P. Seidel, "Vector Field Based Shape Deformations," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 25, no. 3, pp. 1118-1125, 2006.

[67] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun, "Discrete Quadratic Curvature Energies," *Computer Aided Geometric Design,* to appear, 2007.

[68] W. Welch and A. Witkin, "Variational Surface Modeling," *Proc. ACM SIGGRAPH '92,* pp. 157-166, 1992.

[69] D. Xu, H. Zhang, Q. Wang, and H. Bao, "Poisson Shape Interpolation," *Proc. Ninth ACM Symp. Solid and Physical Modeling (SPM '05),* pp. 267-274, 2005.

[70] G. Xu, "Discrete Laplace-Beltrami Operators and Their Convergence," *Computer-Aided Geometric Design,* vol. 21, no. 8, pp. 767-784, 2004.

[71] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh Editing with Poisson-Based Gradient Field Manipulation," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 23, no. 3, pp. 644-651, 2004.

[72] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel, "Harmonic Guidance for Surface Deformation," *Computer Graphics Forum (Proc. Eurographics),* pp. 601-609, 2005.

[73] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large Mesh Deformation Using the Volumetric Graph Laplacian," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 24, no. 3, pp. 496-503, 2005.

[74] D. Zorin, P. Schröder, and W. Sweldens, "Interactive Multiresolution Mesh Editing," *Proc. ACM SIGGRAPH '97,* pp. 259-268, 1997.

Mario Botsch received the MS degree in mathematics from the University of Erlangen, Germany, in 1999 and the PhD degree from the RWTH Aachen University of Technology in 2005. He is a postdoctoral research associate and lecturer at the Computer Graphics Laboratory, ETH Zurich. From 1999 to 2000, he worked as a research associate at the Max-Planck Institute for Computer Science, Saarbrücken, Germany. From 2001 to 2005, he worked as a research associate at RWTH Aachen. His research interests include geometry processing, particularly mesh generation, mesh optimization, shape editing, and point-based representations.

Olga Sorkine received the BSc degree in mathematics and computer science and the PhD degree in computer science from Tel Aviv University in 2000 and 2006, respectively. She is a postdoctoral researcher at the Computer Graphics Group, Technical University of Berlin. Dr. Sorkine's research focuses on computer graphics and geometric modeling, with an emphasis on interactive applications. She is interested in theoretical foundations and practical algorithms for digital content creation tasks, such as shape representation, approximation and editing, artistic modeling techniques, computer animation and digital image manipulation.