On Linear Variational Surface Deformation Methods

Mario Botsch
Computer Graphics Laboratory
ETH Zurich

Olga Sorkine[†]
Computer Graphics Group
TU Berlin

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 three-fold: the deformation technique has to be sufficiently fast, robust, and 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 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 the 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

I. INTRODUCTION

T HIS 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 developing 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, e.g., 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, while space-deformations are well exposed in the literature (see, e.g., [6], [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 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 by 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 non-linear, 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, e.g., 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 goal is to help the reader make practical conclusions about the recited techniques; namely, we wish to answer the question: under which circumstances is each method useful, and how to choose the appropriate algorithm for my 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 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, 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, that 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 II and III). We then perform a practical comparison between representative methods (Section IV) that visualizes the main aspects of various deformation setups and methods. We dedicate a special section for questions and answers, where we shed light on several obscure points of the deformation approaches (Section V), and we conclude in Section VI.

II. 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 non-linear thin shell deformation model for continuous surfaces and show the typically employed simplifications and linearizations (Section II-A). After that we discuss its discretization to triangle meshes (Section II-B), which then leads to merely solving a sparse linear system (Section II-C). The linearization will be shown to distort fine-scale geometric details, which is taken care of by multiresolution deformations (Section II-D). In Section II-E we describe approaches related to the presented techniques, and classify them with respect to the methods they use for surface deformation and multiresolution detail preservation.

A. 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. While 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, parameterized by a function $\mathbf{p} : \Omega \subset \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$. 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, $\mathbf{I}(u,v), \mathbf{II}(u,v) \in \mathbb{R}^{2\times 2}$, can be used to measure geometrically intrinsic (i.e., 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]:

$$E_{\text{shell}}(\mathcal{S}') = \int_\Omega k_s \left\| \mathbf{I}' - \mathbf{I} \right\|_F^2 + k_b \left\| \mathbf{II}' - \mathbf{II} \right\|_F^2 \, dudv \,, \quad (1)$$

where $\mathbf{I}'$, $\mathbf{II}'$ 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 to fix certain surface parts $\mathcal{F} \subset \mathcal{S}$ and to prescribe displacements for the so-called *handle* region(s) $\mathcal{H} \subset \mathcal{S}$. In an interactive application $\mathcal{S}'$ has to be re-computed by minimizing E_{shell} each time the user manipulates the boundary constraints, for instance by moving the handle region $\mathcal{H}$.

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

$$\tilde{E}_{\text{shell}}(\mathbf{d}) = \int_\Omega k_s \left(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2 \right) + \qquad (2)$$
$$k_b \left(\|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \right) \, dudv \,,$$

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* PDE

$$-k_s \, \Delta \mathbf{d} \, + \, k_b \, \Delta^2 \mathbf{d} = \mathbf{0} \,, \qquad (3)$$

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

$$\Delta \mathbf{d} = \text{div}\,\nabla\,\mathbf{d} = \mathbf{d}_{uu} + \mathbf{d}_{vv} \,,$$
$$\Delta^2 \mathbf{d} = \Delta(\Delta \mathbf{d}) = \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv} \,.$$

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

In order for the change of second derivatives in (2) to closely approximate the change of surface curvatures (i.e., 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 et al. [26].
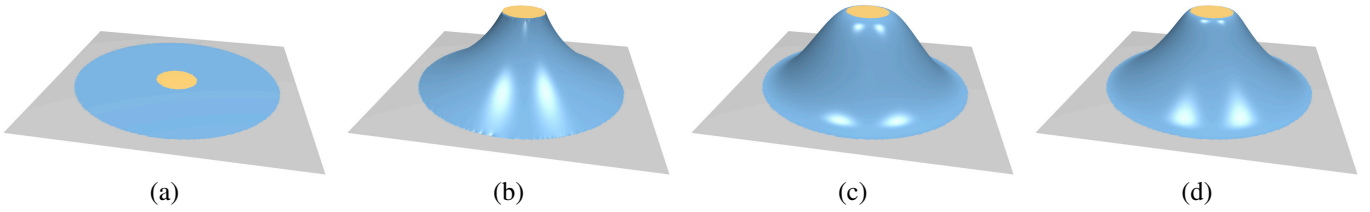
Fig. 1. The original surface $\mathcal{S}$ (a) is edited by minimizing its deformation energy, subject to user-defined constraints that fix the gray part $\mathcal{F}$ of the surface and prescribe the transformation of the yellow handle region $\mathcal{H}$. The linearized energy (2) consists of stretching and bending terms, and the examples show pure stretching with $k_s = 1$, $k_b = 0$ (b), pure bending with $k_s = 0$, $k_b = 1$ (c), and a weighted combination with $k_s = 1$, $k_b = 10$ (d).

As a consequence, the Laplace operator Δ w.r.t. the parametrization $\mathbf{p}$ turns into the Laplace-Beltrami operator $\Delta_\mathcal{S} = \operatorname{div}_\mathcal{S} \nabla_\mathcal{S}$ w.r.t. 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 surface area and curvature are minimized instead of their changes, i.e., stretching and bending. The linearized *membrane* and *thin-plate* energies corresponding to (2) are defined as

$$\tilde{E}_{\text{memb}} (\mathbf{p}) \; = \; \int_\Omega \|\mathbf{p}_u\|^2 + \|\mathbf{p}_v\|^2 \; \mathrm{d}u \mathrm{d}v \, ,$$

$$\tilde{E}_{\text{plate}} (\mathbf{p}) \; = \; \int_\Omega \|\mathbf{p}_{uu}\|^2 + 2 \|\mathbf{p}_{uv}\|^2 + \|\mathbf{p}_{vv}\|^2 \; \mathrm{d}u \mathrm{d}v \, . \tag{5}$$

Analogously 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.

B. 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 w.l.o.g. that from the n vertices $(v_1, \ldots, v_n)$ the first n' vertices are free, while the last $k = n - n'$ vertices $(v_{n'+1}, \ldots, v_n)$ are constrained, i.e., they constitute the fixed part $\mathcal{F}$ and handle region $\mathcal{H}$.

In order to discretize the above equations for triangle meshes one can either employ finite differences or finite elements. The Finite Element Method (FEM) leads to more accurate approximations in general, but for thin shell problems like (1) or (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 non-conforming 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

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

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

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

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.
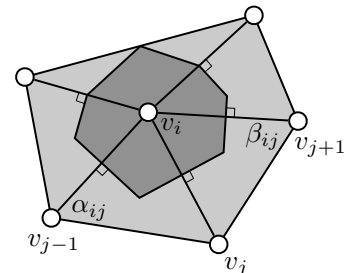

Fig. 2. The angles α_{ij} and β_{ij} and the (dark grey) Voronoi area A_i used to discretize the Laplace-Beltrami $\Delta_\mathcal{S}$ at a vertex v_i in equations (6) and (7).

Higher-order Laplacians are then simply defined recursively:

$$\Delta_{\mathcal{S}}^k f(v_i) = w_i \sum_{v_j \in \mathcal{N}_1(v_i)} w_{ij} \left(\Delta_{\mathcal{S}}^{k-1} f(v_j) - \Delta_{\mathcal{S}}^{k-1} f(v_i) \right),$$

$$\Delta_{\mathcal{S}}^0 f(v_i) = f(v_i).$$

C. Numerical Solution

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

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

where $\mathbf{M}$ 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 V-C). This yields a non-zero right-hand side $\mathbf{b} \in \mathbb{R}^{n' \times 3}$ and leads to a $n' \times n'$ 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 $n' \times n'$ sub-matrices by $\mathbf{L}$ and $\mathbf{L}^2$. Pre-multiplying the above system by $\mathbf{M}$ finally yields the symmetric system

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

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, e.g., 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], [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].

While 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), only for different right-hand sides $\mathbf{M} \mathbf{b}$. 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.
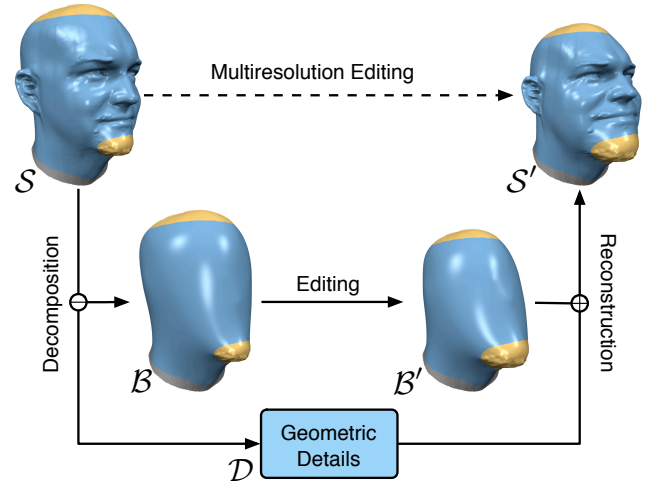

Fig. 3. A multiresolution editing framework consists of three main operators: the *decomposition* operator, which separates the low and high frequencies, the *editing* operator, which deforms the low frequencies, and the *reconstruction* operator, which adds the details back onto the modified base surface.

Thanks to a matrix pre-ordering the resulting Cholesky factor is also sparse, leading to 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 so-called multiple-right-hand-side problems [7], [54].

D. Multiresolution Hierarchies

The deformation techniques described above approximate the non-linear 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 non-linear problem, and therefore cannot be solved by a linearized technique (cf. 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 *multi-scale 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 $\mathcal{B}$, i.e., 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.
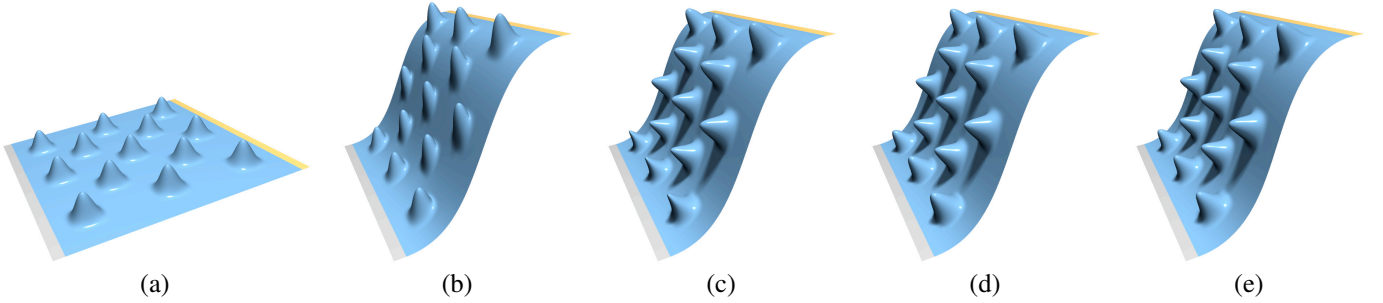
Fig. 4. The rightmost strip $\mathcal{H}$ of the bumpy plane (a) is translated up. The intuitive local rotations of geometric details cannot be achieved by a linearized deformation alone, as can be seen in (b), but require a multiresolution decomposition. Normal displacements (c) correctly rotate local details, but cause distortions under bending deformations, which can be seen in the left-most row of bumps. The non-linear displacement volumes (d) as well as 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 *geometric* hierarchy of smoother and smoother meshes. While for subdivision surfaces these two concepts are tightly 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 $\mathcal{B}$ 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 w.r.t. $\mathcal{B}$ [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, in particular 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 to their corresponding vertices of $\mathcal{B}$. This idea leads to *normal displacements* that are perpendicular to $\mathcal{B}$, i.e., 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 in average about 9 times longer than 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 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 (cf. Figs. 4c, 5a). If displacement vectors cross each other, which happens if the curvature of $\mathcal{B}'$ becomes larger than the displacement length h_i, 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 a prism, the volume of which is kept constant during deformations. The local volume preservation leads to more intuitive detail reconstructions and avoids local self-intersections (cf. Figs. 4d, 5b). However, the improved detail preservation comes at the higher computational cost of a non-linear detail reconstruction process.

Botsch et al. [14] recently proposed a multiresolution representation that provides results similar to displacement volumes (cf. Figs. 4e, 5c), but requires to solve 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 to simplify the actual computation of the deformation transfer, such that the approach only requires the solution of a linear Poisson system (see also Section V-E).

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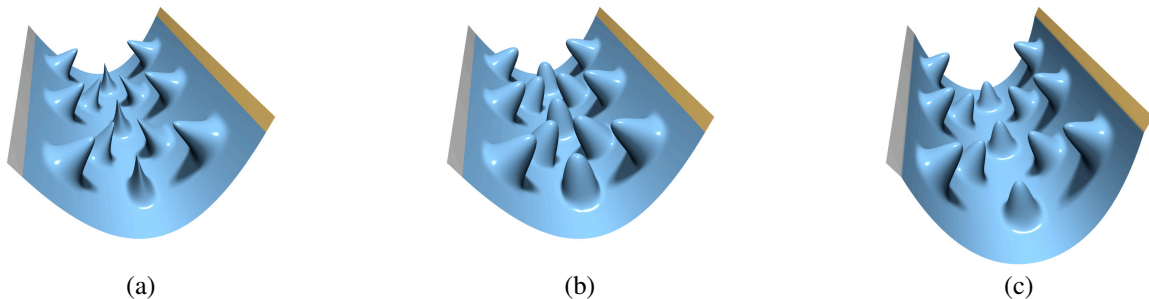
Fig. 5. A concave bending of the bumpy plane of Fig. 4. Multiresolution representations based on normal displacements (a) unnaturally distort geometric details and almost lead to self-intersections, whereas displacement volumes (b) and deformation transfer (c) achieve more natural results.

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, 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 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 is 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 pre-determined by the coarse control grid and hence cannot be chosen precisely on the detailed mesh.

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 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 to prescribe constraints for each individual vertex of $\mathcal{S}$, which are then interpolated by an energy minimizing, 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 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 multi-level B-spline interpolation allows to control their influence radius. However, the required parameterization might lead to geometric distortions.

Botsch and Kobbelt [10] extend [34] by anisotropic and tri-harmonic 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 to pre-compute linear basis functions for the deformation $\mathbf{d}$, which in each frame can be evaluated more efficiently than solving the linear system (8). While normal displacements were used in the original paper [10], a multiresolution representation based on deformation transfer was shown in [14] to yield better results, in particular for bending deformations.

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

III. 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, 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 by the particular differential properties they use (Section III-A) and the manipulation thereof (Section III-B), 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 that humans are sensitive to; 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 $\mathbf{g}'$:

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

where 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 asked to equal the source image gradients, while 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 positions

of some points on the surface). The discrete energy these approaches minimize has usually the form

$$ \mathbf{p}' = \underset{\mathbf{p}'}{\operatorname{argmin}} \sum_i A_i \left\| \mathfrak{D} \left(\mathbf{p}'_i \right) - \mathbf{l}_i \right\|^2 \, , \tag{11} $$

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

$$ \mathbf{D}^T \mathbf{M} \mathbf{D} \, \mathbf{p}' = \mathbf{D}^T \mathbf{M} \mathbf{1} \, . \tag{12} $$

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 between 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 direct carry-over of the technology developed for images onto surfaces, and the techniques we review next deal with the challenge in various ways.

A. Differential Representations

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

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

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

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, 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) using a global operator $\mathbf{G}$, 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 3 spatial coordinates (m being the number of triangles). Thus, one can write down the following formula 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 non other than the cotangent discretization of the Laplace-Beltrami operator [14], discussed in Section II-B, so we can simply write

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

which is the discretized version of (13).

To deform a surface 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

$$\mathbf{p}'_i = \mathbf{c}_i \qquad (16)$$

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

However, the result of such 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
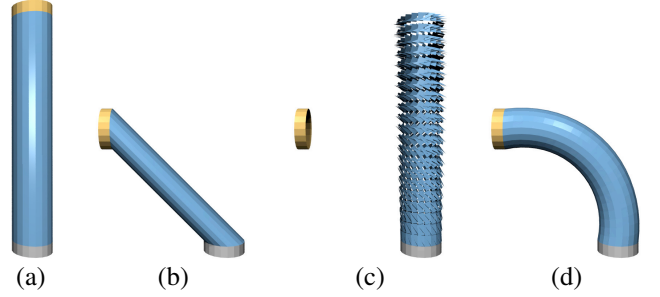

Fig. 6. Using gradient-based editing to bend the cylinder (a) by 90°. Reconstructing the mesh from new handle positions, but *original* gradients distorts the object (b). Applying damped local rotations derived from (25) to the individual triangles breaks up the mesh (c), but solving the Poisson system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

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 III-B.

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, i.e., taking $f \equiv \mathbf{p}$ in (6); the resulting vector is the mean curvature normal:

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

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 II-D.

Laplacian editing was developed concurrently with gradient-based editing, and similarly 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

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

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 Laplace-Beltrami $\Delta_{\mathcal{S}}$ and we arrive at the discretized equation

$$\mathbf{L}^2 \mathbf{p}' = \mathbf{L} \boldsymbol{\delta} , \qquad (19)$$

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):

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

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

$$\mathbf{L}^T\mathbf{M}\mathbf{L}\mathbf{p}' = \mathbf{L}^T\mathbf{M}\boldsymbol{\delta}$$
$$\left(\mathbf{M}^{-1}\mathbf{L}_s\right)^T \mathbf{M} \left(\mathbf{M}^{-1}\mathbf{L}_s\right)\mathbf{p}' = \left(\mathbf{M}^{-1}\mathbf{L}_s\right)^T\mathbf{M}\boldsymbol{\delta}$$
$$\mathbf{L}_s\mathbf{M}^{-1}\mathbf{L}_s\mathbf{p}' = \mathbf{L}_s\boldsymbol{\delta}. \qquad (21)$$

Note that if we multiply both sides of (21) by $\mathbf{M}^{-1}$ we arrive at the bi-Laplacian equation (19). Moreover, if the right-hand side is set to zero (i.e., $\boldsymbol{\delta} = \mathbf{0}$), 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 either incorporated as hard or soft constraints. Hard constraints lead to 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 V for details). Although the system is very simple and can be efficiently solved by sparse direct methodologies mentioned earlier, the results only look satisfactory when the starting surface is a membrane or thin-plate (i.e., 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 II: the system tries to preserve the orientation of the Laplacian vectors w.r.t. the global coordinate system, whereas in reality they should rotate with the deformed surface.

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 one-rings with respect to the frames. More precisely, the relationship between the local frames of two neighboring vertices v_i, v_j in the input surface is recorded by the coefficients of a 3×3 matrix $\mathbf{A}_{ij}$ such that

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

The one-ring vectors are encoded with respect to the local frame by a set of 3 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, so are the coefficients $\mathbf{A}_{ij}, \alpha_{ij}, \beta_{ij}, \gamma_{ij}$, which singles out this representation from other differential coordinates. The (over-determined) 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 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.

B. 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 non-linear, so one can only offer a reasonable estimate if one desires to avoid global non-linear 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:

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

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 $\mathcal{H}$ he/she manipulates (this can be deduced, e.g., from a transform UI 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]; both are interpolated over the region of interest (ROI) on the mesh according to the geodesic distance from the handle:

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

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

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

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

2) Harmonic propagation:
The discrete geodesic distances turned out to be a suboptimal parameter to propagate the transformations because they may be non-smooth 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

$$\mathbf{L}\mathbf{s} = \mathbf{0}, \qquad (26)$$

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
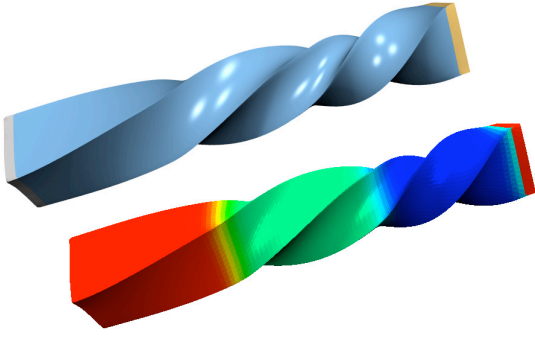
Fig. 7. A non-uniform twist using the material-aware deformation technique [52], with stiffness weights color-coded in the bottom image.

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.

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 was done in [52]. The user may define stiffness by a painting interface, which 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 (cf. Fig. 7). The constraints of the minimization above are the same as in harmonic propagation, i.e., 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 (e.g. rotation) is actually provided. If the handle is only translated, 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.

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], 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 non-orthonormal frames, which may lead to area and volume shrinkage. If the modeling constraints on the frames involve solely orthogonal transformations, it is therefore advisable to normalize and orthogonalize the frames.

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 pseudo-inverse 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, and 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 the smoother the estimation is, naturally at larger computational cost. Note that this method is *sensitive to translations* of the handle.

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 to use non-linear combinations of $\mathbf{p}'$, turning (24) into a global non-linear optimization. It is possible, however, to linearize the *similarity* transformations [60]:

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

The parameters $s, \mathbf{h}$ are determined by writing down the desired transformation constraints, i.e., $\mathbf{T}_i(\mathbf{p}_i - \mathbf{p}_j) = \mathbf{p}'_i - \mathbf{p}'_j$ for each $v_j \in \mathcal{N}_k(v_i)$ and thus extracting $s, \mathbf{h}$ 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, proposed in [33] is to scale the triangles of the deformed surface back to
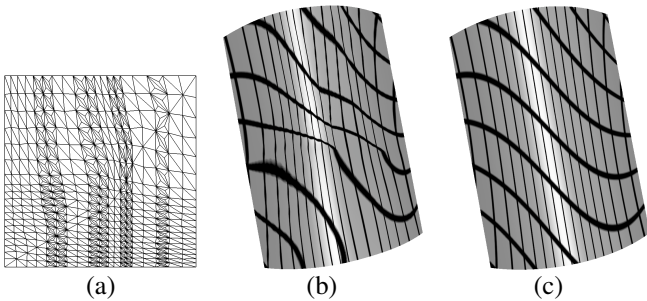
Fig. 8. A 100k triangle version of the mesh (a) from [27] was bent to compare the original Laplacian editing formulation $\mathbf{L}^T\mathbf{L}\mathbf{p} = \mathbf{L}^T\boldsymbol{\delta}$ (b), and the correct one $\mathbf{L}^2\mathbf{p} = \mathbf{L}\boldsymbol{\delta}$ (c). Both (b) and (c) use the cotangent Laplacian.

their original size and re-stitching the mesh using the Poisson setup (15). The implicit optimization of local transformations is also sensitive to translational modeling constraints.

C. 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 to add 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 solves by back-substitution (plus intermediate transformation computations), since the system matrix (21) remains the same and can be pre-factored. 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 V-E); they then apply these rotations to the gradients of the input mesh to reconstruct the final result using the Poisson framework (15).

Sorkine et al. [60] proposed to use the Laplacian representation coupled with implicit transformation optimization, derived from one-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 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 re-solve for the vertex positions 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, that are *not* constrained to linearized similarity transformed but instead are allowed to be any affine transformations $\mathbf{T}_i = \mathbf{U}'_i \mathbf{U}_i^+$, where $\mathbf{U}_i$ are the one-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 one-rings, in which case a perturbation is required.

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 discretization of the L_2 product on the mesh [67]. This lead to normal equations of the form

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

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)

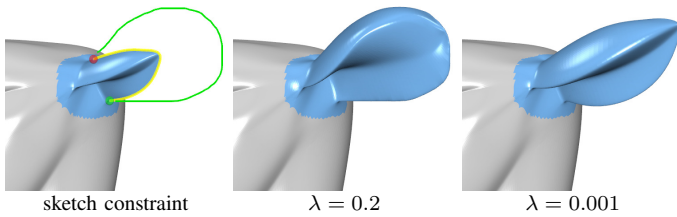sketch constraint $\lambda = 0.2$ $\lambda = 0.001$

Fig. 9. Varying the weighting of soft positional constraints may be beneficial, e.g., 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, while a larger weight makes the system follow the sketched curve more precisely.

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 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 et al. [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 an 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 II-C. Yet 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, while 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].

IV. COMPARISON & DISCUSSION

In this section we compare the different mesh editing techniques described in Section II and Section III. 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 using a representative subset of the described techniques. Notice that our goal is *not* to show the best-possible results 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 non-linear surface deformation PriMo [12], which does not suffer from linearization artifacts.

The first technique 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, i.e., 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 using the gradient of the deformation, i.e., its rotation and scale/shear components. Consequently, the technique works very well for rotations. However, as mentioned in Section III-B, 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 V-A).

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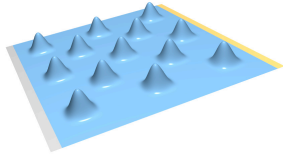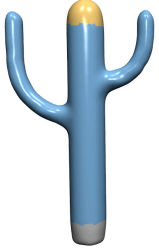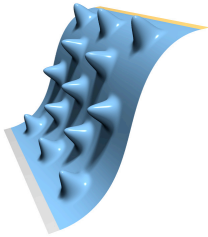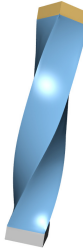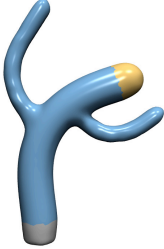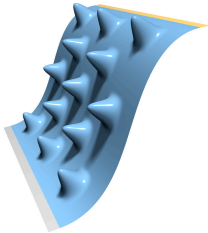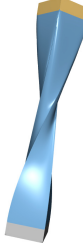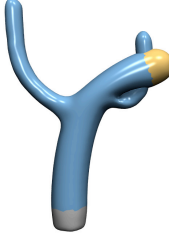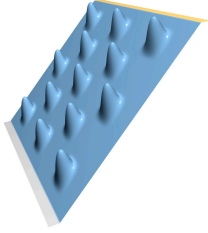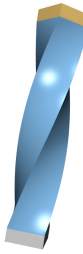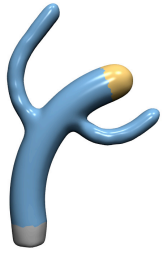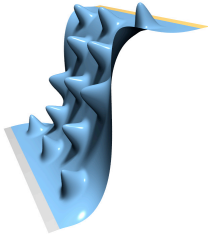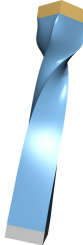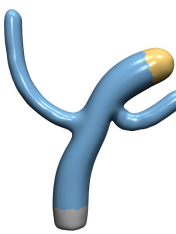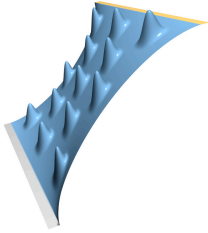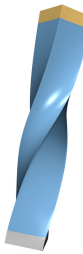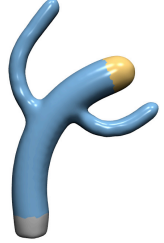
Approach	Pure Translation	120° bend	135° twist	70° bend
Original model				
Non-linear prism-based modeling [12]				
Thin shells [10] + deformation transfer [14]				
Gradient-based editing [72]				
Laplacian-based editing with implicit optimization [60]				
Rotation invariant coordinates [42]				

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 IV.

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 adjusted slightly, but they have high requirements on surface fairness, boundary continuity, and the precise control thereof. For such kind of 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 rotations might be available from, e.g., a sketching interface [49], [73] or a motion capture system [54].

Applications that require both large-scale translations and rotations are problematic for all linear approaches. In this case one can either employ are more complex non-linear technique, or split up large deformations into a sequence of smaller ones. While the non-linear techniques are computationally and implementation-wise 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, non-linear methods become more and more tractable, which already lead to a first set of non-linear, yet interactive, mesh deformation approaches [4], [12], [31], [36], [53], [62], [66].

V. DEFORMATION FAQ

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

A. "What is the influence of the Laplacian discretization?"

Most of the approaches described in Section II and Section III 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 for instance in [34], [41], [60], [63], 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 II-B)

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

as proposed by [18], [45], [51] and employed for instance in [10], [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], [70].

While the cotangent discretization clearly gives the best results, it can also lead to numerical problems in the presence of near-degenerate triangles, since then 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 re-meshed isotropically, as proposed in [11].

B. "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

$$\int_\Omega H^2 \, \mathrm{d}u\mathrm{d}v \; = \; \int_\Omega \left(\kappa_1^2 + \kappa_2^2 \right) \, \mathrm{d}u\mathrm{d}v \; + \; 2 \int_\Omega K \, \mathrm{d}u\mathrm{d}v,$$

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

$$\int_\Omega \kappa_1^2 + \kappa_2^2 \, \mathrm{d}u\mathrm{d}v \;\; \approx \;\; \int_\Omega \|\mathbf{p}_{uu}\|^2 + 2 \|\mathbf{p}_{uv}\|^2 + \|\mathbf{p}_{vv}\|^2 \, \mathrm{d}u\mathrm{d}v$$

$$\int_\Omega H^2 \, \mathrm{d}u\mathrm{d}v \;\; \approx \;\; \int_\Omega \|\mathbf{p}_{uu}\|^2 + 2\mathbf{p}_{uu}^T \mathbf{p}_{vv} + \|\mathbf{p}_{vv}\|^2 \, \mathrm{d}u\mathrm{d}v.$$

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 II-C). Even when discretizing the mean curvature energy instead of the above Euler-Lagrange equations one arrives at the same linear system [67].

C. "What is the difference between hard constraints and soft least-squares constraints?"

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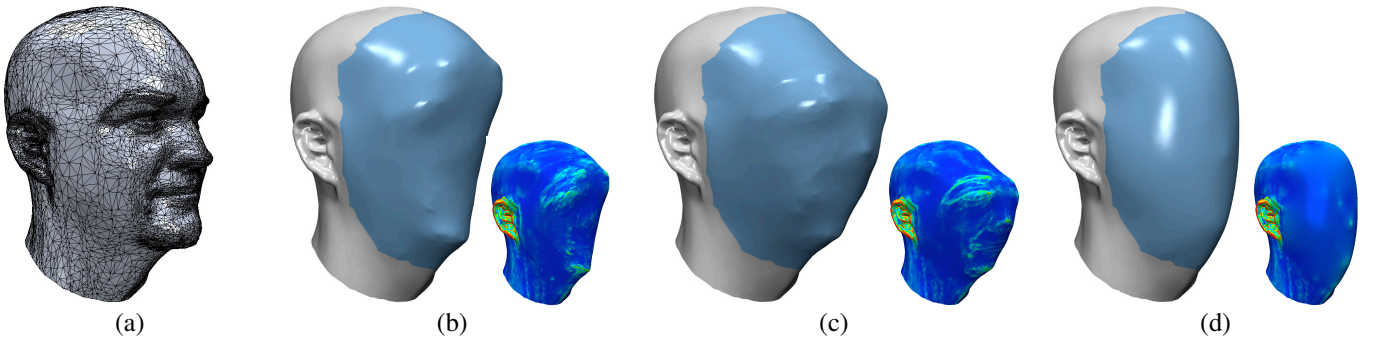
Fig. 11. Different Laplace-Beltrami discretizations are evaluated by minimizing the thin-plate energy of the irregular mesh (a) by solving the Euler-Lagrange equation $\Delta_{\mathcal{S}}^2 \mathbf{p} = \mathbf{0}$. Both the uniform Laplacian (b) and the cotangent Laplacian without the area term (c)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

Most mesh editing approaches consider those constraints as *hard* constraints. For instance, solving a bi-Laplacian system as described in Section II 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 $n' \times n'$ sub-matrix 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

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

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

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

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

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

In order to get (close to) 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

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

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.

D. "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 multiresolution decomposition in Section II and rotations of Laplacians in Section III. As described in Section II, 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

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

introduced in Section III yields the equations

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

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

The basic variational bending energy minimization and Laplacian-based surface deformation can therefore be considered equivalent. They differ in the way they are extended to

preserve fine-scale details, i.e., finding local rotations of the geometric details. While all multiresolution approaches derive those rotations from the deformation of the low-frequency base surface (Section II-D), there are several approaches to rotate the differential coordinates (Section III-B).

E. "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 $\mathcal{T}$, which 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 to each triangle t_i a fourth point, 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

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

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 $\mathbf{G}_i \in \mathbb{R}^{3\times3}$ 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

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

is solved, as described in Section III.

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 $\mathcal{S}'$.

VI. 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 of all explain the original motivation behind these techniques, that 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 an weaknesses of each approach in extreme deformation cases. We hope that our qualitative description and practical illustrations will help the readers to understand the ideas behind these methods and also to 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 is 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 non-linear deformation problem. The various machinery that is meant to mask the linearization errors works in some scenarios, but fails in others, as we demonstrated. As computing resources become faster and faster, and previously infeasible numerical methods become tractable, there is now room for non-linear 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 research-wise there are yet many areas where they can be incorporated and explored further. Moreover, we anticipate further development of non-linear 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.

References

[1] Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. Multilevel solvers for unstructured surface meshes. *SISC*, 26(4):1146–1165, 2005.

[2] Marc Alexa. Local control for mesh morphing. In *Proceedings of Shape Modeling International*, pages 209–215. IEEE Computer Society Press, 2001.

[3] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003.

[4] Oscar Kin-Chung Au, Chiew-Lan Tai, Ligang Liu, and Hongbo Fu. Dual Laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):386–395, 2006.

[5] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, 1995.

[6] Dominique Bechmann. Space deformation models survey. *Computers & Graphics*, 18(4):571–586, 1994.

[7] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. *IMA Mathematics of Surfaces XI, Lecture Notes in Computer Science*, 3604:62–83, 2005.

[8] Mario Botsch and Leif Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *Proceedings of Vision, Modeling, and Visualization*, pages 402–410, 2001.

[9] Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum (Proceedings of Eurographics)*, 22(3):483–491, 2003.

[10] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 23(3):630–634, 2004.

[11] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 189–196. ACM Press, 2004.

[12] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. PriMo: Coupled prisms for intuitive surface modeling. In *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 11–20, 2006.

[13] Mario Botsch, Mark Pauly, Christian Rössl, Stephan Bischoff, and Leif Kobbelt. Geometric modeling based on triangle meshes. In *Eurographics 2006 Course Notes*, 2006.

[14] Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. Deformation transfer for detail-preserving surface editing. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, pages 357–364, 2006.

[15] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. In *Proceedings of ACM SIGGRAPH*, pages 257–266. ACM Press, 1991.

[16] Fehmi Cirak, Michael Ortiz, and Peter Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[17] Fehmi Cirak, Michael Scott, Peter Schröder, Michael Ortiz, and Erik Antonsson. Integrated modeling, finite-element analysis, and design for thin-shell structures using subdivision. *CAD*, 34(2):137–148, 2002.

[18] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH*, pages 317–324. ACM Press/Addison-Wesley Publishing Co., 1999.

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

[20] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 173–182. ACM Press, 1995.

[21] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 21(3):249–256, 2002.

[22] David Forsey and Richard Bartels. Hierarchical B-spline refinement. In *Proceedings of ACM SIGGRAPH*, pages 205–212. ACM Press, 1988.

[23] Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Computer Graphics Forum*, 26(1):34–45, 2007.

[24] Michael Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics State of the Art Report*, 1999.

[25] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[26] Günther Greiner and Joachim Loos. Data dependent thin plate energy and its use in interactive surface modeling. *Computer Graphics Forum (Proceedings of Eurographics)*, 15(3):175–185, 1996.

[27] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. *Computer Graphics Forum (Proceedings of Eurographics)*, 25(3):547–556, 2006.

[28] Heinrich W. Guggenheimer. *Differential Geometry*. McGraw–Hill, New York, 1963.

[29] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *Proceedings of ACM SIGGRAPH*, pages 325–334. ACM Press, 1999.

[30] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. In *Proceedings of ACM SIGGRAPH*, pages 95–102, 2000.

[31] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shanghua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 25(3):1126–1134, 2006.

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

[33] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):1134–1141, 2005.

[34] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 105–114. ACM Press, 1998.

[35] Leif Kobbelt, Jens Vorsatz, and Hans-Peter Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry: Theory and Applications*, 14:5–24, 1999.

[36] Vladislav Kraevoy and Alla Sheffer. Mean-value geometry encoding. *International Journal of Shape Modeling*, 12(1):29–46, 2006.

[37] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *Proceedings of ACM SIGGRAPH*, pages 85–94. ACM Press, 2000.

[38] Seungyong Lee. Interactive multiresolution editing of arbitrary meshes. *Computer Graphics Forum (Proceedings of Eurographics)*, 18(3):73–82, 1999.

[39] Yaron Lipman, Daniel Cohen-Or, Ran Gal, and David Levin. Volume and shape preservation via moving frame manipulation. *ACM Transactions on Graphics*, 26(1), 2007.

[40] Yaron Lipman, Olga Sorkine, Marc Alexa, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Laplacian framework for interactive mesh editing. *International Journal of Shape Modeling*, 11(1):43–62, 2005.

[41] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[42] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):479–487, 2005.

[43] Martin Marinov, Mario Botsch, and Leif Kobbelt. GPU-based multiresolution deformation using approximate normal field reconstruction. *Journal of Graphics Tools*, 12(1):27–46, 2007.

[44] Martin Marinov and Leif Kobbelt. Automatic generation of structure preserving multiresolution models. *Computer Graphics Forum (Proceedings of Eurographics)*, 24(3):479–486, 2005.

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

[46] Tim Milliron, Robert J. Jensen, Ronen Barzel, and Adam Finkelstein. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, 2002.

[47] Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 167–176. ACM Press, 1992.

[48] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[49] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):1142–1147, 2005.

[50] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 22(3):313–318, 2003.

[51] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, 2(1):15–36, 1993.

[52] Tiberiu Popa, Dan Julius, and Alla Sheffer. Material-aware mesh deformations. In *Proceedings of Shape Modelling International*, pages 141–152. IEEE Computer Society, 2006.

[53] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *Proceedings of the Second International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)*, pages 68–75. IEEE Computer Society Press, 2004.

[54] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 25(3):1108–1117, 2006.

[55] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of Graphics Interface*, pages 258–264, 1992.

[56] Olga Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[57] Olga Sorkine. *Laplacian Mesh Processing*. PhD thesis, School of Computer Science, Tel Aviv University, 2006.

[58] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[59] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 42–51. Eurographics Association, 2003.

[60] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.

[61] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 23(3):399–405, 2004.

[62] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):488–495, 2005.

[63] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 351–358. ACM Press, 1995.

[64] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of ACM SIGGRAPH*, pages 205–214. ACM Press, 1987.

[65] Bernhard Thomaszewski, Markus Wacker, and Wolfgang Strasser. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 107–116, 2006.

[66] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 25(3):1118–1125, 2006.

[67] Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. Discrete quadratic curvature energies. *CAGD*, 2007. To appear in the special issue on Discrete Differential Geometry.

[68] William Welch and Andrew Witkin. Variational surface modeling. In *Proceedings of ACM SIGGRAPH*, pages 157–166, 1992.

[69] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, pages 267–274. ACM Press, 2005.

[70] Guoliang Xu. Discrete Laplace-Beltrami operators and their convergence. *Computer Aided Geometric Design*, 21(8):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 Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 23(3):644–651, 2004.

[72] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum (Proceedings of Eurographics)*, pages 601–609, 2005.

[73] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph Laplacian. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):496–503, 2005.

[74] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *Proceedings of ACM SIGGRAPH*, pages 259–268. ACM Press/Addison-Wesley Publishing Co., 1997.

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

Olga Sorkine is a postdoctoral researcher at the Computer Graphics Group of Technische Universität Berlin. She received the BSc degree in mathematics and computer science from Tel Aviv University in 2000 and completed her PhD in computer science in 2006, also at Tel Aviv University. Her research interests are in computer graphics and include interactive geometric modeling, shape approximation, shape and image manipulation, computer animation and expressive modeling techniques.