Volume xx (200y), Number z, pp. 1–11

Example-Driven Deformations Based on Discrete Shells

Stefan Fröhlich, Mario Botsch

Computer Graphics Group, Bielefeld University

Abstract

Despite the huge progress made in interactive physics-based mesh deformation, manipulating a geometrically complex mesh or posing a detailed character is still a tedious and time-consuming task. Example-driven methods significantly simplify the modeling process by incorporating structural or anatomical knowledge learned from example poses. However, these approaches yield counter-intuitive, non-physical results as soon as the shape space spanned by the example poses is left. In this paper we propose a modeling framework that is both example-driven and physics-based and thereby overcomes the limitations of both approaches. Based on an extension of the discrete shell energy we derive mesh deformation and mesh interpolation techniques that can be seamlessly combined into a simple and flexible mesh-based inverse kinematics system.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Mesh deformation is a fundamental and challenging topic in digital geometry processing, with various applications in as diverse fields as shape design, engineering, and computer animation. As a consequence, a huge number of different modeling approaches has been proposed over the years, most of which can be classified as *physics-based* or *physics-inspired*: They employ elastic energies that approximate (or at least qualitatively mimic) the behavior of elastic objects known from continuum mechanics. Despite the impressive progress made in this field, manipulating geometrically complex shapes, for instance a detailed character model, can still be a time-consuming and tedious process even with state-of-the-art modeling tools. This is mainly because no knowledge about the semantic or anatomical structure of the model is incorporated into the physics-based deformation energy.

In contrast, *example-driven deformations*, as pioneered in the mesh-based inverse kinematics (MeshIK) system of Sumner et al. [SZGP05], achieve natural shape manipulations by a suitable blending of carefully designed example poses. While the input poses now provide the previously missing knowledge about the model's deformation behavior, existing example-driven methods are not physics-based: When the modeling constraints cannot be represented within the shape space spanned by the example poses (the *example space*), the deformation might yield unnatural results.

In this paper we propose a novel mesh deformation approach that combines the strengths of example-driven and physics-based techniques and thereby avoids their respective limitations: Our method incorporates given example poses, but gracefully falls back to physics-inspired deformations when leaving the example space.

In a general example-driven deformation framework the designer imposes modeling constraints and the system computes the shape closest to the example space meeting these constraints. Therefore such a framework conceptually consists of two main components:

- An *interpolation operator* that blends between the given input poses in order to find the mesh that best matches the designer's constraints *within* the example space.
- A *deformation operator* that deforms the interpolated shape to exactly satisfy the modeling constraints, thereby potentially deviating from the example space.

For instance, the interpolation and deformation operators of the original MeshIK [SZGP05] are both based on deformation gradients and linear Poisson systems. Linear deformation gradients, however, were recently shown to yield artifacts in the context of mesh deformation [BS08] and mesh interpolation [WDAH10], which consequently also lead to problems for the MeshIK system they are employed for.

Preprint, final version to appear in COMPUTER GRAPHICS *Forum*.

Inspired by MeshIK [SZGP05], our goal is to overcome the limitations inherent to deformation gradients. To this end, we base our modeling system on an extension of the nonlinear discrete shell energy [GHDS03], which measures stretching and bending as deviations of edge lengths and dihedral angles, respectively. This allows us to develop deformation and interpolation operators that are free of the above mentioned linearization artifacts.

Our deformation and interpolation techniques *on its own* are on a par with (but not necessarily superior to) state-of-the-art methods for these problems. The major strength of our approach is the seamless integration of these two components into a simple and elegant formulation for example-based *and* physics-based deformations. We incorporate the computation of the optimal interpolation weights, the interpolation itself, and the constrained mesh deformation into a simple nonlinear energy, which can easily be minimized through a straightforward Gauss-Newton method. We further present a multiresolution optimization that enables the editing of complex models at interactive rates. Our proposed modeling framework overcomes several limitations of current example-based approaches: It can interpolate large rotations between example poses, is fully rotation invariant, and falls back to physically-plausible deformations when leaving the example space.

2. Related Work

In this section we discuss existing example-driven deformation approaches as well as mesh deformation and mesh interpolation methods—since these two techniques are integral components of our modeling framework.

Mesh Deformation

For the following discussion we focus on surface-based deformation techniques—in contrast to space deformations—in order to enable the combination with a mesh interpolation operator. Following [BS08], we further classify these approaches into (i) shell-based techniques, which approximate the elastic energy of thin shells [TPBF87], and (ii) methods based on differential coordinates, which manipulate the shape through deformation gradients, Laplacians, or local frames [Sor06].

Most of the earlier techniques simplify the underlying energies to yield a linear variational optimization. The inherent drawbacks of this linearization have been analyzed in [BS08]: Shell-based methods typically have problems with large rotational deformations (e.g. [KCVS98, BK04]), and methods based on differential coordinates or local frames often are translation-insensitive (e.g. [YZX*04, ZRKS05, LSLC05, KG08]), which basically requires the user to prescribe position *and* orientation of handle points and thereby prevents a simple click-and-drag interaction.

Recent approaches employ nonlinear elastic energies to overcome these limitations, again being shell-based [BPGK06, BMWG07] or based on differential coordinates or frames [SK04, HSL*06, SZT*07, AFTCO07, SA07]. Compared to linear methods, the nonlinear optimization is considerably more involved. Nonlinear approaches therefore typically perform the optimization in a hierarchical, adaptive, or subspace manner.

In this paper we adapt the discrete shell energy [GHDS03] for our mesh deformation operator. Thanks to this nonlinear, physics-based energy our deformations are free of linearization artifacts, and we can control the local surface stiffness (similar to [PJS06, HZS*06, KG08]) as well as the relation of stretching and bending resistance. Note that while PriMo [BPGK06] performs equally well for pure deformation, it cannot be easily combined with mesh interpolation.

Mesh Interpolation

In order to avoid the artifacts from linear interpolation of vertex positions, several approaches interpolate per-triangle affine transformations instead, i.e., per-face deformation gradients [ACOL00, XZWB05, SZGP05]. In this case, per-face rotations have to be interpolated in a nonlinear manner, typically through spherical linear interpolation. This interpolation of deformation gradients can fail for per-face rotations of more than 180°, since then the shorter spherical interpolation path is taken, which might not be the correct one. Baran et al. [BVGP09] avoid large rotations by splitting the mesh into patches and interpolating per-face rotations *relative* to the patch rotation. An alternative is to replace the per-face rotations between example poses by *relative rotations* between neighboring frames [LSLC05, KG08]. The three cited methods successfully interpolate between meshes even in the presence of large rotations. However, they have to solve for rotations and vertex positions in sequence using two linear systems, which still is an efficient linear process, but cannot be easily used as a component of a MeshIK system.

Kilian et al. [KMP07] and Chao et al. [CPSS10] interpolate models along geodesics in suitable shape spaces, but their methods are either too complex [KMP07] or work for solid models only [CPSS10]. Recently, Winkler et al. [WDAH10] proposed to represent and interpolate meshes in terms of edge lengths and dihedral angles. In order to find the mesh that best matches the interpolated edge lengths and angles they employ a rather complicated hierarchical shape matching technique, which prevents their method from being directly used in a MeshIK system.

However, Winkler's approach [WDAH10] uses exactly the same local mesh properties (edge lengths and dihedral angles) for mesh interpolation as discrete shells [GHDS03] use for mesh deformation. Therefore these methods conceptually fit nicely together, and we develop a relatively simple and efficient energy minimization that replaces the complex hierarchical shape matching of [WDAH10].

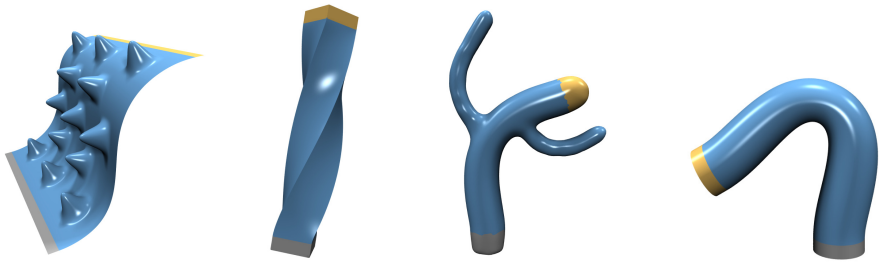


Figure 1: *Benchmark deformations from [BS08]: Translating the right border of a bumpy plane; twisting a bar by* 135°*; bending a cactus by* 70°*; bending a cylinder by* 120°*. The gray surface regions are fixed, the yellow regions are handle constraints, the blue regions are unconstrained and determined by minimizing the energy (5).*

Example-Based Deformation

Early approaches for example-based deformations were proposed in the context of skeleton-controlled articulated models, first by pose-dependent corrections of vertex positions for linear interpolation skinning [LCF00]. Later approaches [WSLG07, WPP07] yield superior results by employing deformation gradients for skinning, example-based corrections, and rotational regression. However, these methods require a skeleton to control the mesh.

Sumner et al. [SZGP05] were the first to propose a mesh-based (in contrast to skeleton-based) inverse kinematics system. Given a rest pose and several example poses, they extract per-face deformation gradients, which are used to interpolate between examples. The interpolation weights are determined automatically from the modeling constraints by a nonlinear optimization. However, the reconstruction of a mesh from the interpolated deformation gradients, which at the same time acts as deformation operator for incorporating the user's constraints, was shown in [BSPG06] to be equivalent to *linear* Poisson-based interpolation [XZWB05] and deformation [YZX*04]. As a consequence, MeshIK inherits their linearization problems: It cannot handle per-face rotations larger than 180° and yields unnatural deformation results when leaving the example space.

The highly efficient data-driven deformation method of Feng et al. [FKY08] fits a set of abstract skeleton bones to an animation sequence, from which it learns the local bone transformations using canonical correlation analysis. At runtime they derive the bone transformations from the modeling constraints and solve a Poisson system for the final vertex positions. It is this final *linear* step that again yields unnatural results when leaving the example space.

We are mostly inspired by the original MeshIK [SZGP05], and improve it by replacing the interpolation and deformation operators by a nonlinear, rotation-invariant technique based on the discrete shell energy. This allows us to handle arbitrary large rotations between example poses and to fall back to physically plausible deformations when leaving the example space.

In the following we first explain our deformation and interpolation operators (Sections 3 and 4), before combining them into an example-driven deformation framework (Section 5). The numerical minimization of the employed nonlinear energy using a Gauss-Newton method is described in Section 6. For increased performance we propose a multiresolution optimization in Section 7.

3. Mesh Deformation Operator

The elastic energy for our mesh deformation operator extends the discrete shell energy by a volume preservation term. In the following we assume a mesh $\mathcal{M}$ with vertex positions $\mathbf{x}_i \in \mathcal{V}$, edges $e_{ij} \in \mathcal{E}$, and faces $f_{ijk} \in \mathcal{F}$.

Analogous to discrete shells [GHDS03] we measure stretching and bending terms E_s and E_b as (weighted) squared deviations of current (lower-case) edge lengths l_e and dihedral angles θ_e from their original (upper-case) values L_e and Θ_e, respectively. To be able to extend the energy later, we denote by l_e^* and θ_e^* the desired target edge lengths and dihedral angles, which for now are just the initial values we want to preserve ($l_e^* = L_e$ and $\theta_e^* = \Theta_e$):

$$E_s = \frac{1}{2} \sum_{e \in \mathcal{E}} \left(l_e - l_e^*\right)^2 \frac{1}{L_e^2}, \tag{1}$$

$$E_b = \frac{1}{2} \sum_{e \in \mathcal{E}} \left(\theta_e - \theta_e^*\right)^2 \frac{L_e^2}{A_e}. \tag{2}$$

A_e denotes the sum of the areas of the two triangles sharing the edge e. Note that we omit the deviations of triangle areas from the original discrete shell energy, since that term did not have a significant impact in our examples.

The stretching and bending energies model two-dimensional surfaces (thin shells). Hence, input meshes are treated as hollow objects, even if they are intended to represent the boundary surface of a solid object. In that case, i.e., for closed surfaces without boundaries, we optionally add a global volume preservation term (similar to [HSL*06]):

$$E_v = \frac{1}{2} \left(v - v^*\right)^2 \frac{1}{V^2}. \tag{3}$$

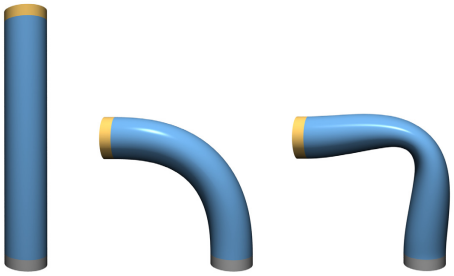


Figure 2: Global *stiffness control: Bending a cylinder (left) with either dominating bending stiffness (center) or stretching stiffness (right).*

Again, v, V, and v^* denote current, initial, and target volume, respectively (with $v^* = V$). As in [HSL*06] we compute the global volume v as the sum of per-face tetrahedral volumes

$$v = \frac{1}{6} \sum_{f_{i,j,k} \in \mathcal{F}} \left(\mathbf{x}_i \times \mathbf{x}_j\right) \cdot \mathbf{x}_k \,. \tag{4}$$

We note that our volume preservation is just a heuristic: The global volume preservation can compensate local volumes changes of one part of the model in a completely different surface region, thereby inducing strain. For a physically accurate solution local volumes (e.g. of a tetrahedral mesh) have to be preserved instead (see e.g. [CPSS10]).

The total elastic energy is the sum of stretching, bending, and volume preservation terms, weighted by stiffness parameters λ, μ, and ν, respectively:

$$E = \lambda E_s + \mu E_b + \nu E_v \,. \tag{5}$$

The weighting coefficients in (1), (2), and (3) are chosen to account for irregular tessellations as well as to yield scale-invariant energy terms. The stiffness parameters in (5) do not have to be adjusted on a per model basis, but are typically set to $\lambda = 100$, $\mu = 1$, and $\nu = 1000$ in our examples.

Mesh editing now amounts to a minimization of the energy (5), while satisfying the user's modeling constraints (prescribed positions for a selected set of vertices). We formulate the energy minimization as a nonlinear least-squares problem and employ a Gauss-Newton method, since this technique requires first-order partial derivatives only and still provides super-linear convergence (see Section 6).

Figure 1 provides a qualitative evaluation of our shell-based deformation on the benchmark models of [BS08]. Comparing our results to Figure 10 of [BS08] reveals that our approach is qualitatively equivalent to PriMo [BPGK06], a state-of-the-art nonlinear approach. A quantitative comparison (in terms of performance) is given in Section 6.

A further advantage of our physics-based energy is the explicit control of stretching and bending stiffness (λ and μ), which allows to vary the surface deformation behavior

Figure 3: Local *stiffness control: Deriving per-edge stiffness from the example poses of Figure 9 leads to more natural deformations (left) than uniform stiffness weights (right).*

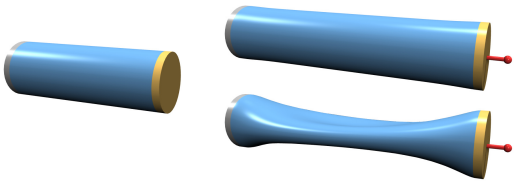


Figure 4: *Stretching a cylinder (left) to 150% of its original length* without *(top) and* with *volume preservation (bottom). The volume error is 42% and 0.007%, respectively.*

(Figure 2). Most approaches based on differential coordinates can (be extended to) locally adjust the surface stiffness, but cannot change the global deformation characteristic. Note that these stiffness parameters, although chosen constant in (5), can also be controlled on a per-edge basis. In Figure 3 we derive per-edge stretching and bending values from the example poses shown in Figure 9 (as described in Appendix A). A pure deformation using these learned stiffness values (without the MeshIK of Section 5) already yields convincing results compared to uniform stiffness parameters.

Figure 4 shows the effect of volume preservation on a stretched cylinder, where the designer can choose (or continuously blend) between a hollow or an (approximately) solid deformation behavior.

4. Mesh Interpolation Operator

The second core component of our example-based deformation framework is the interpolation operator: Given a rest-pose mesh $\mathcal{M}$ and a set of k compatibly tessellated example poses $\mathcal{M}_1, \ldots, \mathcal{M}_k$, we want to explore the shape space spanned by these meshes through a geometrically meaningful interpolation (and extrapolation) technique.

Following the idea of Winkler et al. [WDAH10] we interpolate meshes in terms of edge lengths and dihedral angles. However, we derive a simpler method for computing the mesh that best matches the interpolated edge lengths and angles, such that our interpolation operator can be combined with the deformation technique of the previous section.

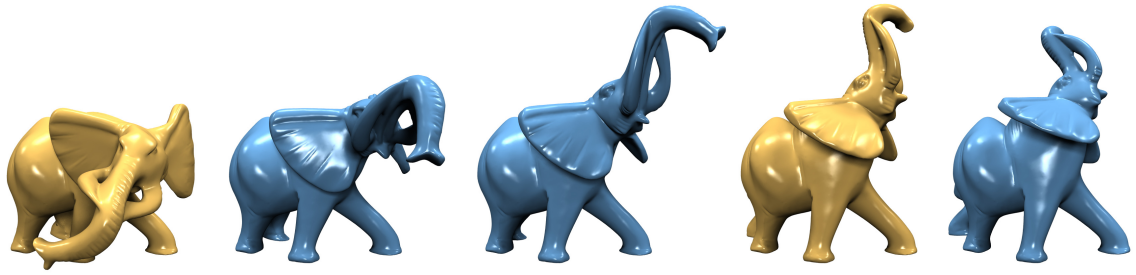


Figure 5: *Interpolation and extrapolation of the yellow example poses. The blending weights are* 0, 0.35, 0.65, 1.0, *and* 1.25.

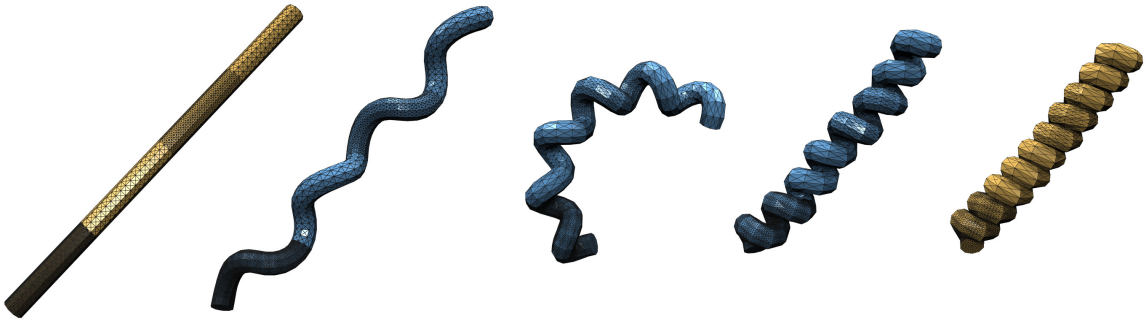


Figure 6: *Interpolation of an adaptively meshed and strongly twisted helix with blending weights* 0, 0.25, 0.5, 0.75, 1.0.

Finding the best matching mesh (in a least squares manner) requires to minimize an energy that penalizes the squared deviations from prescribed target edge lengths l_e^* and dihedral angles θ_e^*. Having the deformation operator of the previous section at hand, all we need to do is to replace the rest-pose lengths L_e and angles Θ_e by the desired interpolated values. Similarly, we can also interpolate the global volume V if the volume preservation is activated.

Given the rest-pose $\mathcal{M}$ and example poses $\mathcal{M}_1,\ldots,\mathcal{M}_k$, we denote by $L_e^{(i)}$, $\Theta_e^{(i)}$, and $V^{(i)}$ the edge lengths, dihedral angles, and volume of mesh $\mathcal{M}_i$, respectively. The target values for (1), (2), and (3) are then determined by linearly interpolating the differences of $\mathcal{M}_i$'s values to those of the rest pose $\mathcal{M}$, controlled by interpolation weights $\alpha_1,\ldots,\alpha_k$:

$$\begin{aligned} l_e^* &= L_e + \sum_{i=1}^{k} \alpha_i \left(L_e^{(i)} - L_e \right) , \\ \theta_e^* &= \Theta_e + \sum_{i=1}^{k} \alpha_i \left(\Theta_e^{(i)} - \Theta_e \right) , \\ v^* &= V + \sum_{i=1}^{k} \alpha_i \left(V^{(i)} - V \right) . \end{aligned} \tag{6}$$

The same nonlinear minimization as used in Section 3 and described in Section 6 then computes the interpolated mesh.

Although in general a mesh with prescribed edge lengths and dihedral angles does not exist, our least squares approximation leads to physically meaningful interpolated shapes. In Figure 5 we show the same elephant interpolation/extrapolation sequence as used in Figure 1 of [KMP07] and Figure 11 of [WDAH10]. Moreover, Figure 6 basically reproduces Figure 6 of [WDAH10] and demonstrates that large rotations and varying mesh resolution are naturally handled by our approach.

These results show our interpolation technique to be qualitatively on a par with the recent state-of-the-art approaches [KMP07, WDAH10]. Since we use the same target objective as Winkler et al. [WDAH10], we basically reproduce their results. However, while they compute the interpolated mesh by a hierarchical shape matching procedure, our method requires a comparatively straightforward energy minimization only, which is the key to combining the deformation and interpolation operators into the example-driven deformation framework described in the next section.

5. Example-Driven Deformation Framework

After introducing the deformation and interpolation operators, we are now ready to derive our example-driven framework. Like in the case of mesh interpolation, the input is a set of example poses $\mathcal{M},\mathcal{M}_1,\ldots,\mathcal{M}_k$. However, instead of just interpolating between the examples, the system should now "learn" the deformation behavior of the rest pose $\mathcal{M}$ from the examples, such that a simple click-and-drag modeling metaphor leads to natural and meaningful results.

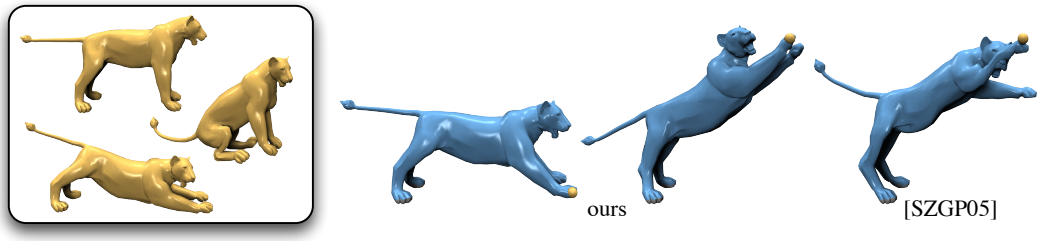


Figure 7: *Given a set of input poses (left) the user can drag handle points (yellow spheres) and the example-based deformation produces a meaningful result by finding the optimal interpolation weights for the input examples. While our rotation-invariant method makes use of the bottom example pose in a rotated fashion, the original MeshIK approach leads to artifacts.*

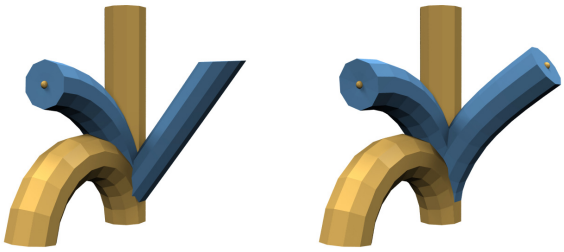


Figure 8: *The original MeshIK yields unwanted shears when leaving the example space (left), whereas our technique still provides physically plausible results (right).*

Instead of preserving original edge lengths and dihedral angles (as for mesh deformation), or prescribing interpolated lengths and angles (as for mesh interpolation), we now have to optimize for the interpolation weights α_i of (6), such that the interpolated mesh best fits the modeling constraints.

One particular strength of our approach is that this optimization of the interpolation weights, the interpolation operator, and the deformation operator can all seamlessly be integrated into one nonlinear energy formulation: We simply use the interpolated target values (6) in the elastic energy (5) (as for mesh interpolation) and furthermore extend the set of unknowns to include both the free vertex positions $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and the interpolation weights $\alpha_1, \ldots, \alpha_k$.

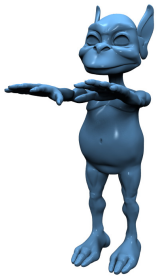
Since the number k of examples typically is very small compared to the number n of free vertices, and since the derivatives of the target objectives w.r.t. the weights α_i are trivial to compute, incorporating examples poses does not lead to a noticeable performance overhead compared to pure mesh deformation or interpolation (see Section 6).

If we compare the results of our example-driven deformation to that of existing example-based approaches [SZGP05, FKY08] we can identify three important advantages: First, our deformation and interpolation operators employ the physics-based discrete shell energy. As a consequence, even when leaving the space of example poses our method still yields physically meaningful results, as shown in Figure 8. This is in contrast to purely example-driven techniques, which lead to undesired shears when deviating too much from the examples (compare to Figure 15 of [FKY08] and Figure 5 of [SZGP05]).

Second, since our interpolation method can deal with large rotations between example poses, our MeshIK system can do as well. The original MeshIK [SZGP05] is based on linear deformation gradients and therefore might yield artifacts for rotations of more than 180° (compare Figure 5.2 of [Sum05] to Figure 6 and our accompanying video).

Third, our method is rotation-invariant, since examples are represented by (changes of) edge lengths and dihedral angles—instead of by per-face affine world-space transformations from the rest pose to the example poses (deformation gradients). As an immediate consequence, our method can use example poses in arbitrary orientations, e.g., to simplify rotation modeling of example poses (see Figure 7 and the accompanying video). In contrast, the original MeshIK cannot deal with rotated (i.e., mis-aligned) poses.

While this particular problem can potentially be fixed by including per-example rotations in MeshIK's optimization process, a similar problem can occur when combining rotations of example poses (see Figure 9). The reason for the failure of the original MeshIK is depicted on the right. In their case, the combined deformation "shoulder + elbow" (i.e. $\alpha_1 = \alpha_2 = 1$) first rotates the shoulder around the *global* y-axis, followed by a rotation of the forearms around the *global* x-axis, leading to a forearm twist. In contrast, our rotation-invariant framework yields the expected combined deformation, as reproduced by our IK system in Figure 9.



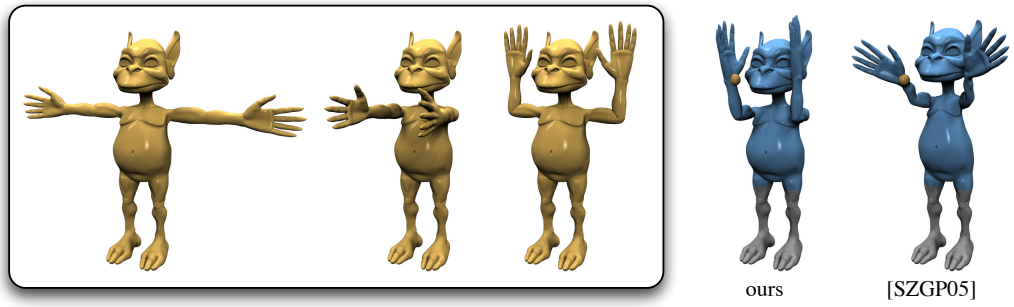


Figure 9: *Using three input poses (left) we fix the legs and drag the handle on the right wrist. Our MeshIK system successfully combines both example deformations, whereas the world-space deformation gradients of the original MeshIK lead to artifacts.*

6. Numerical Minimization

We have shown how to formulate the mesh deformation and mesh interpolation operators, as well as the complete example-driven deformation technique as a nonlinear energy minimization. The simplicity of our energy (5) enables the analytic computation of its gradients, such that we can employ an efficient Gauss-Newton minimization [MNT04].

6.1. Gauss-Newton Minimization

We start by formulating the energy minimization as a nonlinear least-squares problem. For a system with n unconstrained vertices $\mathbf{x}_1,\dots,\mathbf{x}_n$, m unconstrained edges, and k example poses, we set up a residual function $\mathbf{f}$ that maps the vector $\mathbf{x} = (x_1,y_1,z_1,\dots,x_n,y_n,z_n,\alpha_1,\dots,\alpha_k)^T$ of unknown vertex positions and interpolation weights to the residuals of edge lengths, dihedral angles, and global volume:

$$\mathbf{f}\colon \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_n \\ y_n \\ z_n \\ \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix} \mapsto \begin{bmatrix} w_{s,1}\,(l_1 - l_1^*) \\ \vdots \\ w_{s,m}\,(l_m - l_m^*) \\ w_{b,1}\,(\theta_1 - \theta_1^*) \\ \vdots \\ w_{b,m}\,(\theta_m - \theta_m^*) \\ w_v\,(v - v^*) \end{bmatrix}. \tag{7}$$

The target values l_i^*, θ_i^*, and v^* are again determined by interpolating the respective values of the example poses using the weights α_i as in (6). The weighting factors are chosen as

$$w_{s,e} = \sqrt{\lambda \frac{1}{L_e^2}}, \quad w_{b,e} = \sqrt{\mu \frac{L_e^2}{A_e}}, \quad w_v = \sqrt{\nu \frac{1}{V^2}}, \tag{8}$$

such that the energy (5) becomes $E(\mathbf{x}) = \frac{1}{2}\,\mathbf{f}(\mathbf{x})^T\,\mathbf{f}(\mathbf{x})$.

In each Gauss-Newton iteration we compute new vertex positions and interpolation weights by solving a linear least squares system for an update direction δ, which is then scaled by a step-size h and used to update $\mathbf{x}$:

$$\mathbf{J}(\mathbf{x})^T\,\mathbf{J}(\mathbf{x})\,\delta = -\mathbf{J}(\mathbf{x})^T\,\mathbf{f}(\mathbf{x}), \tag{9}$$

$$\mathbf{x} \leftarrow \mathbf{x} + h\delta. \tag{10}$$

In the above equation $\mathbf{J}$ denotes the $(2m+1)\times(3n+k)$ Jacobian matrix of $\mathbf{f}$. It is built from first-order partial derivatives of the components of $\mathbf{f}$ with respect to the vertex positions $\mathbf{x}_i$ or interpolation weights α_i. For all derivatives simple analytical expressions exist, which are given in Appendix B.

The step-size h is determined by simple bisection search: Starting with $h = 1$ we successively halve h until the scaled step decreases the energy, i.e., $E(\mathbf{x}+h\delta) < E(\mathbf{x})$. We stop the iteration if even a minimum step-size of $h_{\min} = 10^{-10}$ does not lead to a further decrease of the energy.

Note that this minimization procedure can be used for pure mesh deformation (no examples, no interpolation weights), pure mesh interpolation (k examples, interpolation weights known), as well as the full example-based framework (k examples, interpolation weights unknown).

6.2. Volume Preservation

A closer look at the linear system (9) reveals a challenging problem caused by the volume preservation. In each row the Jacobian $\mathbf{J}$ contains the partial derivatives of the corresponding residual with respect to the components of $\mathbf{x}$. For an edge length constraint this leads to $2\cdot 3 = 6$ non-zeros per row, for a dihedral angle with four involved vertices we get 12 non-zeros. The volume v, however, depends on all vertex positions, such that the bottom row of $\mathbf{J}$ is completely filled. As a consequence, the system matrix $\mathbf{J}^T\mathbf{J}$ is dense, such that efficient sparse solvers cannot be applied. In [HSL*06], volume constraints were treated as *hard* constraints using Lagrange multipliers. Our *soft* volume constraint with a controllable stiffness parameter requires a different approach.

We propose to exploit the Sherman-Morrison formula [GL89] to efficiently solve the system (9). To this end, we denote by $\mathbf{u}$ the (dense) bottom row of $\mathbf{J}$, which yields

$$\mathbf{J} = \begin{bmatrix} \tilde{\mathbf{J}} \\ \mathbf{u}^T \end{bmatrix} \quad \Rightarrow \quad \mathbf{J}^T\mathbf{J} = \tilde{\mathbf{J}}^T\tilde{\mathbf{J}} + \mathbf{u}\mathbf{u}^T =: \mathbf{A} + \mathbf{u}\mathbf{u}^T.$$

Hence, the system matrix $\mathbf{J}^T\mathbf{J}$ can be written as a rank-one update of $\mathbf{A} = \tilde{\mathbf{J}}^T\tilde{\mathbf{J}}$. The Sherman-Morrison formula states how to update a matrix inverse after such an update:

$$\left(\mathbf{A} + \mathbf{u}\mathbf{u}^T\right)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{A}^{-1}}{1 + \mathbf{u}^T\mathbf{A}^{-1}\mathbf{u}}.$$

The solution of $(\mathbf{A} + \mathbf{u}\mathbf{u}^T)\,\delta = \mathbf{b}$ with $\mathbf{b} = -\mathbf{J}^T\mathbf{f}$ becomes

$$\delta = \mathbf{A}^{-1}\mathbf{b} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{A}^{-1}\mathbf{b}}{1 + \mathbf{u}^T\mathbf{A}^{-1}\mathbf{u}}.$$

We can therefore compute the solution of (9) by an efficient three step procedure:

(i). Compute $\mathbf{y}$ by solving $\mathbf{A}\mathbf{y} = \mathbf{b}$.
(ii). Compute $\mathbf{z}$ by solving $\mathbf{A}\mathbf{z} = \mathbf{u}$.
(iii). Compute $\delta = \mathbf{y} - \mathbf{z}\,(\mathbf{u}^T\mathbf{y})/(1 + \mathbf{u}^T\mathbf{z})$.

Note that $\mathbf{A} = \tilde{\mathbf{J}}^T\tilde{\mathbf{J}}$ is sparse, since $\tilde{\mathbf{J}}$ does not contain the volume constraint. Furthermore, because $\tilde{\mathbf{J}}$ includes $2m \approx 6n$ constraints, $\tilde{\mathbf{J}}$ has full rank and $\mathbf{A}$ is symmetric positive definite. We can therefore use a sparse Cholesky solver [DH05] for steps (i) and (ii). Since the rather expensive matrix factorization of (i) can be re-used in step (ii), our three-step procedure comes at a negligible additional cost compared to an optimization without volume constraints.

Note that this technique can not only be used for the efficient computation of volume constraints. It can also be employed for other constraints that depend on a large number of mesh vertices, such as, for instance, constraining the area of a large surface patch in the framework of [EP09]. We remark that there are several ways to compute a factorization of a matrix with dense rows through matrix updates. For instance, a general method based on updating a QR factorization is presented in [Sun95]. Our proposed solution has the advantage that it is straightforward to implement with any of the commonly employed sparse Cholesky solvers, such as for instance CHOLMOD [DH05].

6.3. Performance & Robustness

Compared to linear deformation methods, nonlinear approaches are computationally more expensive, might get stuck in local minima, and are in general numerically more sensitive to "bad" input data. Extremely short edges, large dihedral angles (fold-overs), or other kinds of degenerate triangles can cause the minimization to stall or even to fail. However, in this respect our method is not more or less sensitive than most other nonlinear deformation techniques.

For instance, the synthetic examples shown in Section 3 converged in just a few iterations, starting from the initial pose. For a more demanding test we compare to the dragon example of PriMo [BPGK06]. As shown in Figure 10, even for this complex model (50k vertices) and a bad starting configuration our minimization converges in just 9 iterations, which took 52s in total on a MacBookPro 2.53 GHz Intel Core 2 Duo. In comparison, PriMo requires 6 iterations and 45s on the same machine (on the original mesh resolution, i.e., without hierarchical optimization). Since our interpolation and MeshIK techniques are basically equivalent in terms of computations, their robustness and performance are comparable to those of the mesh deformation.

Figure 10: *Even from this bad initial configuration our nonlinear deformation method converges in 9 iterations.*

Input meshes that contain local fold-overs in joint regions are problematic for our method, such as, e.g., the Lion and Goblin poses of Figures 7 and 9, which have been produced by low quality skinning methods. The "hinge-like" fold-overs constitute local minima where the optimization might get stuck. However, these configurations are easy to detect and surprisingly easy to handle: As soon as the dihedral angle of an edge e changes by more than 180° between the rest pose $\mathcal{M}$ and some example pose $\mathcal{M}_i$, we deactivate that edge by setting its bending stiffness μ_e to zero. This simple heuristic solved the fold-over problem for all our examples.

7. Multiresolution Optimization

The above results demonstrate our nonlinear optimization to work reliably, but its performance is clearly too slow for interactive applications. Since this is a standard problem for nonlinear editing or interpolation approaches, several techniques have been proposed to increase performance (and further improve robustness): performing the optimization on a reduced model [DSP06, FKY08] or in a suitable subspace [HSL*06], or computing it in a hierarchical manner [BPGK06, SZT*07, KMP07, WDAH10].

The cage-based subspace technique [HSL*06], while being an elegant formulation, is not directly applicable to our setting, since building compatible cages for all example poses is a challenging problem by itself. We instead follow the approach of Kilian et al. [KMP07] and concurrently decimate all input meshes to a complexity of about 1000 ver-

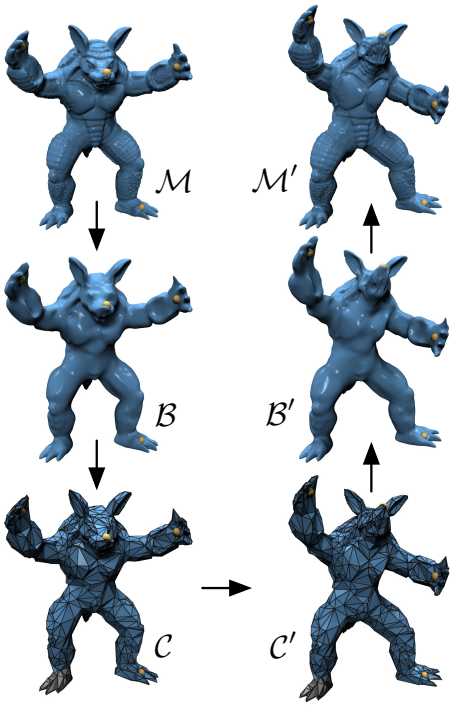


Figure 11: *The different meshes involved in our multiresolution optimization: Original resolution (top), smooth base surface (middle), coarse simulation mesh (bottom), all in initial (left) and deformed configuration (right).*

Model	$\lvert\mathcal{V}\rvert$	G.-N.	$\mathcal{B}$ & $\mathcal{C}$	$\mathcal{B}'$ & $\mathcal{M}'$
Helix	612	18		
Lion	5k	250	377	23
Elephant	40k	3100	4110	271
Dragon	50k	3700	5300	323
Armadillo	166k		21000	1278

Table 1: *Performance statistics, with times given in milliseconds, measured on a MacPro 2.66GHz Intel Xeon. From left to right: number of vertices; time for one Gauss-Newton iteration on the* original *resolution, for multiresolution preprocessing (computing $\mathcal{C}$ and $\mathcal{B}$), and for reconstructing $\mathcal{B}'$ and $\mathcal{M}'$ from $\mathcal{C}'$. On the coarse mesh, one Gauss-Newton step takes about 30ms. On the original Armadillo model, the Gauss-Newton step failed due to memory restrictions.*

Blending weight	Elephant	Dragon
0.25	0.23%	0.49%
0.35	0.29%	
0.50	0.34%	0.77%
0.65	0.39%	
0.75	0.40%	0.69%

Table 2: *Hausdorff distance (in percent of the bounding box diagonal) between the result of the multiresolution optimization and the full high-resolution optimization.*

tices, where we use the sum of per-mesh quadric error metrics [GH97] to prioritize halfedge collapses.

The nonlinear optimization is then performed on the reduced meshes exactly as described in the previous sections. One Gauss-Newton iteration on the coarse mesh takes about 30ms only, which allows for real-time mesh editing (since we initialize the Gauss-Newton optimization with the previous frame's result it typically converges in less than five iterations). However, an obvious limitation is that modeling constraints can only be placed on coarse mesh vertices.

In order to transfer the coarse solution to the original, high-resolution mesh, we adapt the multiresolution modeling approach proposed in [BSPG06]. The main idea is to interpret the coarse mesh's vertices as modeling constraints (*anchors*) on the high-resolution mesh. After decimating the input meshes, we pre-compute a high-resolution but low-frequency base surface $\mathcal{B}$ by minimizing the thin-plate energy of the original mesh $\mathcal{M}$ with the anchors as constraints. This amounts to solving a bi-Laplacian linear system. After deforming the coarse mesh $\mathcal{C}$ into $\mathcal{C}'$, we compute the deformed base surface $\mathcal{B}'$ by updating the anchor positions and again minimizing the thin-plate energy. The resulting base surface deformation $\mathcal{B} \mapsto \mathcal{B}'$ is finally applied to the original mesh $\mathcal{M}$ by deformation transfer [SP04], resulting in the deformed high-resolution and detailed mesh $\mathcal{M}'$. This process is depicted in Figure 11.

It is important to note that this multiresolution reconstruction is also rotation-invariant: If a rotation $\mathbf{R}$ is applied to the coarse mesh, i.e., $\mathcal{C}' = \mathbf{R}(\mathcal{C})$, then the mesh fairing with rotated anchors will reproduce this rotation, $\mathcal{B}' = \mathbf{R}(\mathcal{B})$, as will the deformation transfer, such that $\mathcal{M}' = \mathbf{R}(\mathcal{M})$.

In our interactive modeling system, the nonlinear deformation can be performed in real-time on the coarse mesh $\mathcal{C}$, and as soon as the mouse is released, the multiresolution reconstruction computes the deformed mesh $\mathcal{M}'$. Table 1 gives performance statistics and compares the nonlinear optimization on the fine mesh (Section 6) to the proposed multiresolution optimization.

An important question is how much the results of the multiresolution optimization deviate from the results of the full high-resolution optimization. We analyzed this difference for interpolation sequences of the Elephant model (Figure 5) and the Dragon model (Figure 10). Table 2 shows that the Hausdorff distance between the two optimizations is always below 1% of the bounding box diagonal, which proves our multiresolution optimization to be a good approximation. If needed, the more accurate high-resolution result can easily be obtained in 2–3 Gauss-Newton iterations starting from the fine mesh $\mathcal{M}'$.

In addition to the figures shown in the paper, the accompanying video also demonstrates the proposed example-based deformation system for a few modeling sessions.

8. Conclusion

We presented a framework for combined physics-based *and* example-driven mesh deformations. Thanks to our fully nonlinear formulation, we can perform large-scale shape manipulations in a high quality, rotation-invariant manner.

The interpolation and deformation operators by themselves deliver comparable results to established methods in terms of both quality and speed. Compared to purely geometric or purely physics-based approaches, incorporating example poses into the deformation process leads to more natural results for models with non-trivial deformation behavior. In contrast to purely example-driven approaches, our system provides physically meaningful deformations even when leaving the space spanned by the input poses. In addition to mesh deformations, our system also features nonlinear mesh interpolation and deformation transfer in a rotation-invariant manner.

Investigating subspace optimizations will be an interesting direction for future work, since this would lead to an even better preservation of target values, and hence to physically more accurate results. In terms of performance a parallelization or GPU-implementation of the muliresolution reconstruction would allow to interact with the high-resolution model, while still computing on the coarse simulation mesh.

Acknowledgements:

The authors are grateful to Martin Kilian for the elephant model of Figure 5, to Tim Winkler for the helix of Figure 6, and to Bob Sumner for the lion model of Figure 7 and an executable of the original MeshIK [SZGP05]. The Armadillo and Dragon models are courtesy of the Stanford 3D Scanning Repository. The authors thank Peter Schröder and Eitan Grinspun for helpful discussions about discrete shells, and Miguel Otaduy for pointing us to the Sherman-Morrison update. Both authors are supported by the Deutsche Forschungsgemeinschaft (Center of Excellence in "Cognitive Interaction Technology", CITEC).

Appendix A: Deriving Stiffness Values from Examples

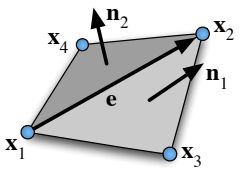
Similar to [PJS06], we derive per-edge stretching stiffness λ_e and bending stiffness μ_e by linearly mapping the stretching and bending between the k example poses to stiffness values:

$$\tilde{\lambda}_e = \max_{i=1,\dots,k} \left| L_e^{(i)} - L_e \right|, \qquad \lambda_e = 1 - \frac{\tilde{\lambda}_e}{\max_e \tilde{\lambda}_e + \varepsilon},$$

$$\tilde{\mu}_e = \max_{i=1,\dots,k} \left| \Theta_e^{(i)} - \Theta_e \right|, \qquad \mu_e = 1 - \frac{\tilde{\mu}_e}{\max_e \tilde{\mu}_e + \varepsilon}.$$

Appendix B: Partial Derivatives for Constructing J

Here we give the partial derivatives of the individual components of the residual function $\mathbf{f}$ from (7) required to build the Jacobian $\mathbf{J}$. The local mesh configuration is shown to the right.



We define the the following vectors

$$\begin{aligned} \mathbf{e} &= \mathbf{x}_2 - \mathbf{x}_1, \\ \mathbf{n}_1 &= (\mathbf{x}_3 - \mathbf{x}_2) \times (\mathbf{x}_3 - \mathbf{x}_1), \\ \mathbf{n}_2 &= (\mathbf{x}_4 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_2). \end{aligned}$$

The derivative of the length constraint for edge $(\mathbf{x}_1, \mathbf{x}_2)$ is simply the normalized edge direction

$$\frac{\partial}{\partial \mathbf{x}_1} w\left(\|\mathbf{e}\| - l^*\right) = w\frac{-\mathbf{e}}{\|\mathbf{e}\|}.$$

The partial derivatives for constraining the dihedral angle $\theta = \angle(\mathbf{n}_1, \mathbf{n}_2)$ are surprisingly simple (see [BMF03]):

$$\frac{\partial}{\partial \mathbf{x}_1} w\left(\theta - \theta^*\right) = w\left[\frac{(\mathbf{x}_3 - \mathbf{x}_2)^T \mathbf{e}}{\|\mathbf{e}\| \, \|\mathbf{n}_1\|^2}\mathbf{n}_1 + \frac{(\mathbf{x}_4 - \mathbf{x}_2)^T \mathbf{e}}{\|\mathbf{e}\| \, \|\mathbf{n}_2\|^2}\mathbf{n}_2\right],$$

$$\frac{\partial}{\partial \mathbf{x}_3} w\left(\theta - \theta^*\right) = w\frac{\|\mathbf{e}\|}{\|\mathbf{n}_1\|^2}\mathbf{n}_1.$$

The derivative of the volume constraint with respect to a vertex position $\mathbf{x}_k$ is computed by summing over $\mathbf{x}_k$'s incident faces $f_{ijk} \in \mathcal{N}(\mathbf{x}_k)$ (see (4)):

$$\frac{\partial}{\partial \mathbf{x}_k} w\left(v - v^*\right) = w \sum_{f_{ijk} \in \mathcal{N}(\mathbf{x}_k)} \left(\mathbf{x}_i \times \mathbf{x}_j\right).$$

Finally, the derivative of a constraint (e.g. edge length) with respect to an interpolation weight is (see (6))

$$\frac{\partial}{\partial \alpha_i} w\left(l - l^*\right) = -w\left(L^{(i)} - L\right).$$

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proc. of ACM SIGGRAPH* (2000), pp. 157–164. 2

[AFTCO07] AU O. K.-C., FU H., TAI C.-L., COHEN-OR D.: Handle-aware isolines for scalable shape editing. *ACM Transactions on Graphics 26*, 3 (2007). 2

[BK04] BOTSCH M., KOBBELT L.: An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics 23*, 3 (2004), 630–634. 2

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proc. of ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 28–36. 10

[BMWG07] BERGOU M., MATHUR S., WARDETZKY M., GRINSPUN E.: TRACKS: toward directable thin shells. *ACM Transactions on Graphics 26*, 3 (2007). 2

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Eurographics symposium on Geometry processing* (2006), pp. 11–20. 2, 4, 8

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230. 1, 2, 3, 4

[BSPG06] BOTSCH M., SUMNER R., PAULY M., GROSS M.: Deformation transfer for detail-preserving surface editing. In *Proc. of Vision, Modeling, and Visualization* (2006), pp. 357–364. 3, 9

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM Transactions on Graphics 28*, 3 (2009), 36:1–36:6. 2

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Transactions on Graphics 29*, 4 (2010), 38:1–38:6. 2, 4

[DH05] DAVIS T. A., HAGER W.: CHOLMOD: supernodal sparse cholesky factorization and update/downdate. http://www.cise.ufl.edu/research/sparse/cholmod, 2005. 8

[DSP06] DER K. G., SUMNER R. W., POPOVIĆ J.: Inverse kinematics for reduced deformable models. *ACM Transactions on Graphics 25*, 3 (2006), 1174–1179. 8

[EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Computer Graphics Forum 28*, 2 (2009), 551–558. 8

[FKY08] FENG W.-W., KIM B.-U., YU Y.: Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Transactions on Graphics 27*, 3 (2008), 91:1–91:9. 3, 6, 8

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *Proc. of ACM SIGGRAPH* (1997), pp. 209–216. 9

[GHDS03] GRINSPUN E., HIRANI A. N., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Proc. of ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 62–67. 2, 3

[GL89] GOLUB G. H., LOAN C. F. V.: *Matrix Computations*. 1989. 8

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM Transactions on Graphics 25*, 3 (2006), 1126–1134. 2, 3, 4, 7, 8

[HZS*06] HUANG J., ZHANG H., SHI X., LIU X., BAO H.: Interactive mesh deformation with pseudo material effects. *Computer Animation and Virtual Worlds 17*, 3-4 (2006), 383–392. 2

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proc. of ACM SIGGRAPH* (1998), pp. 105–114. 2

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Transactions on Graphics 27*, 2 (2008), 12:1–12:13. 2

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Transactions on Graphics 26*, 3 (2007). 2, 5, 8

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of ACM SIGGRAPH* (2000), pp. 165–172. 3

[LSLC05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics 24*, 3 (2005), 479–487. 2

[MNT04] MADSEN K., NIELSON H. B., TINGLEFF O.: *Methods for Non-Linear Least Squares Problems (2nd ed.)*. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004. 7

[PJS06] POPA T., JULIUS D., SHEFFER A.: Material-aware mesh deformations. In *Proc. of IEEE International Conference on Shape Modeling and Applications* (2006), pp. 141–152. 2, 10

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. of Eurographics symposium on Geometry processing* (2007), pp. 109–116. 2

[SK04] SHEFFER A., KRAEVOY V.: Pyramid coordinates for morphing and deformation. In *Proc. of Symp. on 3D Data Processing, Visualization and Transmission* (2004), pp. 68–75. 2

[Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum 25*, 4 (2006), 789–807. 2

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (2004), 399–405. 9

[Sum05] SUMNER R. W.: *Mesh Modification Using Deformation Gradients*. PhD thesis, Massachusetts Institute of Technology, 2005. 6

[Sun95] SUN C.: *Dealing with Dense Rows in the Solution of Sparse Linear Least Squares Problems*. Tech. rep., Cornell University, 1995. 8

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Transactions on Graphics 24*, 3 (2005), 488–495. 1, 2, 3, 6, 10

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Transactions on Graphics 26*, 3 (2007). 2, 8

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proc. of ACM SIGGRAPH* (1987), pp. 205–214. 2

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Computer Graphics Forum 29*, 2 (2010), 309–318. 1, 2, 4, 5, 8

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-time enveloping with rotational regression. *ACM Transactions on Graphics 26*, 3 (2007). 3

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Computer Graphics Forum 26*, 3 (2007), 265–273. 3

[XZWB05] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. In *Proc. of ACM symposium on Solid and Physical Modeling* (2005), pp. 267–274. 2, 3

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Graphics 23*, 3 (2004), 644–651. 2, 3

[ZRKS05] ZAYER R., RÖSSL C., KARNI Z., SEIDEL H.-P.: Harmonic guidance for surface deformation. *Computer Graphics Forum 24*, 3 (2005), 601–609. 2