Bounded Biharmonic Weights for Real-Time Deformation

Alec Jacobson[1] Ilya Baran[2] Jovan Popović[3] Olga Sorkine[1,4]
[1]New York University [2]Disney Research, Zurich [3]Adobe Systems, Inc. [4]ETH Zurich

Abstract

Object deformation with linear blending dominates practical use as the fastest approach for transforming raster images, vector graphics, geometric models and animated characters. Unfortunately, linear blending schemes for skeletons or cages are not always easy to use because they may require manual weight painting or modeling closed polyhedral envelopes around objects. Our goal is to make the design and control of deformations simpler by allowing the user to work freely with the most convenient combination of handle types. We develop linear blending weights that produce smooth and intuitive deformations for points, bones and cages of arbitrary topology. Our weights, called bounded biharmonic weights, minimize the Laplacian energy subject to bound constraints. Doing so spreads the influences of the controls in a shape-aware and localized manner, even for objects with complex and concave boundaries. The variational weight optimization also makes it possible to customize the weights so that they preserve the shape of specified essential object features. We demonstrate successful use of our blending weights for real-time deformation of 2D and 3D shapes.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: shape deformation, articulated character animation, generalized barycentric coordinates, linear blend skinning

Links: DL PDF WEB VIDEO

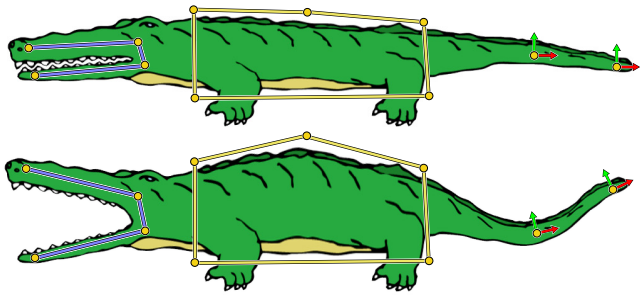

Figure 1: *Bounded biharmonic blending supports points, bones, and cages arranged in an arbitrary configuration. This versatility makes it possible to choose the right tool for each subtask: bones to control rigid parts, cages to enlarge areas and exert precise control, and points to transform flexible parts. The weight computation is done at bind time so that high-quality deformations can be computed in real time with low CPU utilization. In this and other figures, affine transformations specified at point handles are illustrated by colored frames. They are omitted when the transformation is just a translation.*

1 Introduction

Interactive space deformation is a powerful approach for editing raster images, vector graphics, geometric models and animated characters. This breadth of possibilities has led to an abundance of methods seeking to improve interactive deformation with real-time computation and intuitive use. Real-time performance is critical for both interactive design, where tasks require exploration, and interactive animation, where deformations need to be computed repeatedly, often sixty or more times per second. Among all deformation methods, linear blending and its variants dominate practical usage thanks to their speed: each point on the object is transformed by a linear combination of a small number of affine transformations.

In a typical workflow, the user constructs a number of handles and the deformation system binds the object to these handles; this is termed the *bind time*. The user then manipulates the handles (interactively or programmatically) and the system deforms the shape accordingly; this is the *pose time*. Unfortunately, linear blending schemes are not always easy to use. The user must choose the handle type a priori and different types have different advantages (Fig. 2). Free-form deformations rely on a lattice of handles, but the requirement for regular structure complicates control of concave objects. Skeleton-based deformations offer natural control for rigid limbs, but are less convenient for flexible regions. Generalized barycentric coordinates provide smooth weights automatically, but require construction of closed or nearly closed cages that fully encapsulate transformed objects and can be tedious to manipulate. In contrast, variational techniques support arbitrary handles at points or regions, but at a greater pose-time cost.

Real-time object deformations would be easier with support for all handle types above: points, skeletons, and cages. Points are quick to place and easy to manipulate. They specify local deformation properties (position, rotation and scaling) that smoothly propagate onto nearby areas of the object. Bones make some directions stiffer than others. If a region between two points appears too supple, bones can transform it into a rigid limb. Cages allow influencing a significant portion of the object at once, making it easier to control bulging and thinning in regions of interest.

Our goal is to supply weights for a linear blending scheme that produce smooth and intuitive deformation for handles of arbitrary topology (Fig. 1). We desire real-time interaction for deforming high-resolution images and meshes. We want smooth deformation near points and other handles, so that they can be placed directly *on* animated surfaces and warped textures. And, we seek a local support region for each handle to ensure that its influence dominates nearby regions and disappears in parts of the object controlled by other handles.

Our solution computes blending weights automatically by minimizing the Laplacian energy subject to upper and lower bound constraints. Because the related Euler-Lagrange equations are biharmonic, we call these weights *bounded biharmonic weights* and the resulting deformation *bounded biharmonic blending*. The weights

weights computed once at bind time!

are computed once at bind time. At pose time, points on the object are transformed in real time by blending a small number of affine transformations. Our examples demonstrate that bounded biharmonic blending produces smooth deformations and that points, bones, and cages have intuitive local influences, even on objects with complex and concave boundaries. Our weight computation requires space discretization and optimization, which could be a drawback in some applications, but the generality of our formulation also makes it possible to provide additional control over the energy minimization, for example to define weights that preserve the shape of specified essential object features.

2 Previous work

Variational methods are known to compute high-quality shape-preserving deformations for arbitrary handles on the surface [Igarashi et al. 2005; Botsch et al. 2006; Sorkine and Alexa 2007; Botsch and Sorkine 2008] and some variational methods work with bones [Weber et al. 2007] or can be extended to other off-surface handles [Botsch et al. 2007]. The primary drawback of these techniques is that they rely on optimization at pose time. Although system matrices can be prefactored and back-substitution can be implemented on a GPU, it is not an embarrassingly parallel problem like linear blend skinning, and is therefore much slower. Even with significant performance tuning [Shi et al. 2007] or model reduction [Der et al. 2006; Sumner et al. 2007; Au et al. 2007], pose-time optimization is too slow to deform high-resolution objects at high framerate, as necessary, for example, for video games. Variational harmonic maps [Ben-Chen et al. 2009] are faster (though not as fast as linear blending), but they restrict the degrees of freedom to harmonic deformations of a (usually manually) specified cage.

Most methods that are fast at pose time compute the transformation at each object point by using a weighted blend of handle transformations. To perform the blending, some methods use moving least squares [Schaefer et al. 2006], some use dual quaternions [Kavan et al. 2008], but most use linear blend skinning (LBS) [Magnenat-Thalmann et al. 1988]. With LBS, the affine transformations of the handles are linearly averaged with different weights to transform each vertex. Although linearly blending rotations leads to well-known artifacts, LBS has been a popular technique for skeletal animation for over two decades because it is simple, predictable, and the pose-time computation can be implemented very efficiently on a GPU. In addition to skeletal animation, most cage-based deformation methods [Floater 2003; Ju et al. 2005; Joshi et al. 2007; Lipman et al. 2007; Hormann and Sukumar 2008] are effectively LBS, where the handle (cage vertex) transformations are restricted to be translations and the focus is choosing the weights. Additionally, the reduced-model variational shape deformation methods mentioned above use LBS to go from the reduced model to the full model.

The choice of weights for LBS determines whether the affine transformations of the handles affect the shape intuitively. In some cases, weights that have a closed form in terms of the handle structure have been used [Shepard 1968; Ju et al. 2005; Lipman et al. 2008], but more often they rely on precomputation at bind time or they are painted by hand. In Sec. 3.1 we formulate the desirable properties of LBS weights and in Sec. 3.2, we discuss previous weight choice schemes in the context of these properties.

3 Bounded biharmonic weights

Our goal is to define smooth deformations for 2D or 3D shapes by blending affine transformations at arbitrary handles. Let $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$ denote the volumetric domain enclosed by the union of the given shape $\mathcal{S}$ and cage controls (if any). We denote the (disjoint) control handles by $H_j \subset \Omega$, $j = 1, \ldots, m$. A handle can be a single point, a region, a skeleton bone (such that H_j consists of all the points on the bone line segment) or a vertex of a cage. The user defines an affine transformation T_j for each handle H_j, and all points $\mathbf{p} \in \Omega$ are deformed by their weighted combinations:

$$\mathbf{p}' = \sum_{j=1}^{m} w_j(\mathbf{p})\, T_j \mathbf{p}, \tag{1}$$

where $w_j : \Omega \to \mathbb{R}$ is the weight function associated with handle H_j. Note that cages are generally understood as closed polygons in 2D or polyhedra in 3D containing $\mathcal{S}$ or part of it, but our framework is agnostic to the cage topology and treats a cage simply as a collection of simplices, with the requirement that these simplices transform linearly as the cage vertices are translated. Hence, open cages are possible (Fig. 11). We do not consider cage faces (line segments in 2D or triangles in 3D) as handles; they receive linear weights, as we will see in Sec. 3.1. Note also that for skeleton bones connected by joints, we formally include each joint point in one single bone of those that share it (we assume that the skeleton is never torn apart, i.e., that all bones sharing a joint transform the joint to the same location). In practice, we constrain the weights at shared points to be equally distributed between the overlapping bones to maximize the symmetry of our weights.

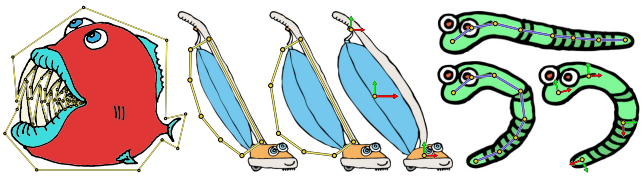

Figure 2: *Left to right: Although cages allow flexible control, setting up a closed cage can be both tedious and unintuitive: the* Pirahna*'s jaws require weaving around the teeth. In the case of the* Vacuum*, points can provide crude scaling effects, while cages provide precise scaling articulation. Point handles can provide loose and smooth control, while achieving the same effect with a skeleton results in an overly complex armature.*

3.1 Formulation

We propose to define the weights w_j as minimizers of a higher-order shape-aware smoothness functional, namely, the Laplacian energy, subject to constraints that enforce interpolation of the handles and several other desirable properties:

$$\underset{w_j,\ j=1,\ldots,m}{\arg\min} \sum_{j=1}^{m} \frac{1}{2} \int_{\Omega} \|\Delta w_j\|^2 dV \tag{2}$$

Sigma: over the volume

$$\text{subject to:} \quad w_j|_{H_k} = \delta_{jk} \tag{3}$$

weights on points on handles equals 1

$$w_j|_F \text{ is linear} \quad \forall F \in \mathcal{F}_{\mathcal{C}} \tag{4}$$

cage weights linear

$$\sum_{j=1}^{m} w_j(\mathbf{p}) = 1 \quad \forall \mathbf{p} \in \Omega \tag{5}$$

partition of parity constraint

$$0 \le w_j(\mathbf{p}) \le 1, \quad j = 1, \ldots, m, \quad \forall \mathbf{p} \in \Omega, \tag{6}$$

where $\mathcal{F}_{\mathcal{C}}$ is the set of all cage faces and δ_{jk} is Kronecker's delta. Fig. 3 shows an example of w_j computed for point handles.

Let us discuss several properties possessed by our weight functions w_j that allow for intuitive and high-quality deformations.

Smoothness: Lack of smoothness at the handles causes visible artifacts in 2D textured shapes (Fig. 9) and prevents placing handles

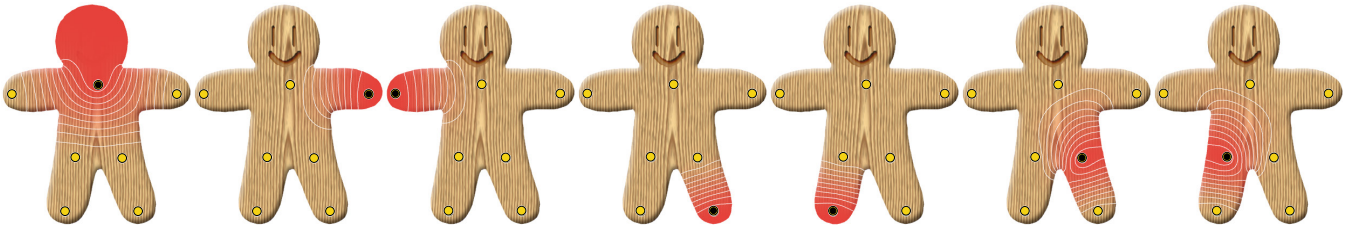

Figure 3: *Bounded biharmonic weights are smooth and local: the blending weight intensity for each handle is shown in red with white isolines. Each handle has the maximum effect on its immediate region and its influence disappears in distant parts of the object.*

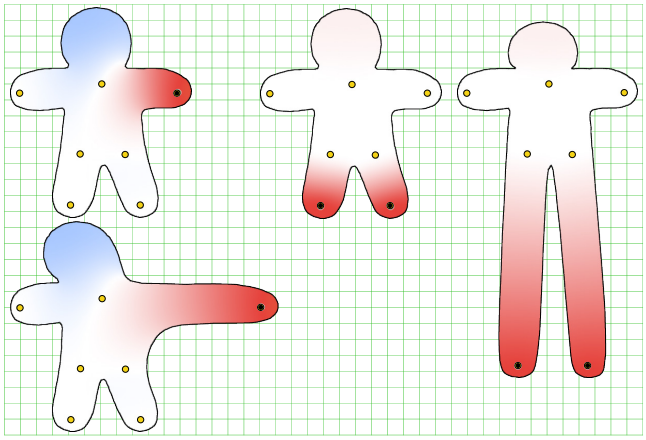

Figure 4: *Weights like unconstrained biharmonic functions that have negative weights (left) and extraneous local maxima (right) lead to undesirable and unintuitive behavior. Notice the shrinking of the head on the right.*

Property	Our method	1	2	3	4
Smoothness	Y	-	Y	Y	-
Non-negativity	Y	Y	-	Y	Y
Shape-awareness	Y	Y	Y	-	-
Partition of unity	Y	Y	Y	Y	Y
Locality and sparsity	Y*	-	-	-	Y
No local maxima	Y*	Y	-	-	Y

1 = [Joshi et al. 2007; Baran and Popović 2007]
2 = [Botsch and Kobbelt 2004]
3 = [Shepard 1968]
4 = [Sibson 1981]
Y* = only experimental confirmation

Table 1: *A summary of the properties of five methods for choosing blending weights. Our method appears to satisfy all of the necessary properties. We have empirical evidence but no formal proof for locality and sparsity or the lack of local maxima in our weights.*

directly *on* 3D shapes. Note that by calculus of variations, minimizing the Laplacian energy (2) amounts to solving the Euler-Lagrange equations, which are the biharmonic PDEs in this case: $\Delta^2 w_j = 0$. Equivalently, we could formulate our blending weights as minimizers of the linearized thin-plate energy, as it leads to the same biharmonic PDE (see e.g. [Botsch and Sorkine 2008]). The bounded biharmonic weights are C^1 at the handles and C^∞ everywhere else, provided that the posed boundary conditions are smooth. This is always the case, with the exception of skeletal joints and cages vertices: for bones connected by joints, the weights must have a discontinuity at the joints since w_j on bone H_j is 1 and it must be zero on the adjacent bone. However, this does not lead to smoothness problems for the actual deformations because the joints are always transformed to the same location by all emanating bones.

For cages, explicit linear interpolation constraints (4) on cage faces are required to achieve expected behavior, since otherwise the cage faces would not deform linearly when translating cage vertices. These linear constraints on cage faces preclude smoothness of the weights at the cage vertices. Therefore our deformations are not smooth at cage vertices, but they are smooth everywhere else, including across cage faces.

Non-negativity: Negative weights lead to unintuitive handle influences, because regions of the shape with negative weights move in the opposite direction to the prescribed transformation (Fig. 4). We explicitly enforce non-negativity in (6), since otherwise biharmonic functions (as in [Botsch and Kobbelt 2004]) are often negative, even if all boundary conditions are non-negative.

Shape-awareness: Informally, shape-awareness implies intuitive correspondence between the handles and the domain Ω. The influence of the handles should conform to the features of the shape and fall off with shape-aware (as opposed to Euclidean) distance. The best shape-aware behavior one can hope for is when the weights w_j depend on the metric of Ω alone and do not change for any possible embedding of Ω. Our weights are shape-aware since the bi-Laplacian operator is determined solely by the metric.

Partition of unity: This classical property (also seen in e.g. Bézier or NURBS) ensures that if the same transformation T is applied to all handles, the entire object will be transformed by T. We enforce this property explicitly in (5) since non-negative biharmonic weights do not sum to 1, unlike unconstrained biharmonic weights.

Locality and sparsity: Each handle should mainly control a shape feature in its vicinity, and each point in Ω should be influenced only by a few closest handles. Specifically, if every locally shortest path (in a shape-aware sense) from a point $\mathbf{p}$ to H_j passes near some other handle, then H_j is "occluded" from $\mathbf{p}$ and $w_j(\mathbf{p})$ should be zero. We observed this property of our weights in all our experiments.

No local maxima: Each w_j should attain its global maximum (i.e., value of 1) on H_j and should have no other local maxima. This property provides monotonic decay of a handle's influence and guarantees that no unexpected influences occur away from the handle. This property was experimentally observed in all our tests; likely, it is facilitated by imposing the bound constraints (6). Without these constraints, the biharmonic functions in general do not necessarily achieve maxima at the handles and cause deformation artifacts (Fig. 4).

3.2 Comparison to Existing Schemes

Existing schemes formulate and satisfy subsets of these properties, but not all. For example, Shepard's [1968] and similar weights (used in embedded deformation [Sumner et al. 2007] and moving least squares image deformation [Schaefer et al. 2006]) are dense and not shape-aware. Cage-based schemes do not support arbitrary

handles: for example, extending harmonic coordinates [Joshi et al. 2007] to handles in the cage interior results in a lack of smoothness (see Fig.9). Heat diffusion weights [Baran and Popović 2007] suffer from the same problem. Natural neighbor interpolation [Sibson 1981] is one of the few schemes that guarantees locality, but it is also not smooth at handles. Biharmonic weights without constraints [Botsch and Kobbelt 2004] are smooth, but can be negative (or greater than one), have local maxima away from handles and result in non-local influences. Notably, our weight functions satisfy all the axioms formulated for higher order barycentric coordinates [Langer and Seidel 2008]. Table 1 shows the properties satisfied by several methods.

A number of recent methods focus on locally preserving or prescribing angles [Lipman et al. 2008; Weber et al. 2009; Weber and Gotsman 2010]. While they have elegant formulations in terms of complex analysis, the methods by Weber et al. are restricted to 2D, while Green Coordinates [Lipman et al. 2008] are only defined for polyhedral cages.

3.3 Shape preservation

The energy minimization framework supports incorporating additional energy terms and constraints to customize the weight functions. One example of a useful addition is making all points of a specified region $\Pi \subset \Omega$ undergo the same transformation, i.e., have all the weight functions be constant on Π ($\nabla w_j|_\Pi = 0$). Since typically, we only prescribe translations, rotations and uniform scales at handles, this implies that Π will undergo a similarity transformation in 2D and an affine transformation in 3D, so that the shape of Π will be preserved. Similarly to the rigidity brush in [Igarashi et al. 2005], the user can paint Π with a (possibly soft) brush, creating a mask $\rho : \Pi \to \mathbb{R}^+$; we then add a least-squares term to our energy minimization:

$$\sum_{j=1}^{m} \frac{1}{2} \int_\Pi \rho \|\nabla w_j\|^2 dV. \tag{7}$$

See Fig. 5, where the shape-preservation brush helped retain the shape of the man's eye while deforming the nose. Note that this is different from placing a handle because no explicit transformation needs to be prescribed by the user; the painted region just follows the transformation from other handles.

3.4 Implementation

We discretize our constrained variational problem (2) using linear finite elements in order to solve it numerically with quadratic programming (we use the flattened mixed FEM formulation for fourth-order problems, as discussed in [Jacobson et al. 2010]). Assuming that the object $\mathcal{S}$ is given as a 2D polyline or triangle mesh in 3D, we sample vertices on all provided skeleton bones and cage faces, and mesh the domain Ω in a way compatible with all of the handles and the vertices of $\mathcal{S}$. The result is a triangle/tetrahedral mesh $\mathcal{M}$ whose vertices $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ include all discretized H_j's and the object itself. The weights become piecewise-linear functions whose vertex values we are seeking; we denote them by column vectors $\mathbf{w}_j = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})^T$. for ech bone V={v1,…vn} contains surface S and handles and all vertices of tet mesh inside Sigma

The Laplacian energy (2) is discretized using the standard linear FEM Laplacian $M^{-1}L$ where M is the lumped mass matrix (with Voronoi area/volume M_i of vertex $\mathbf{v}_i$ on each diagonal entry i) and L is the symmetric stiffness matrix (i.e., the cotangent Laplacian in 2D and its equivalent in 3D):

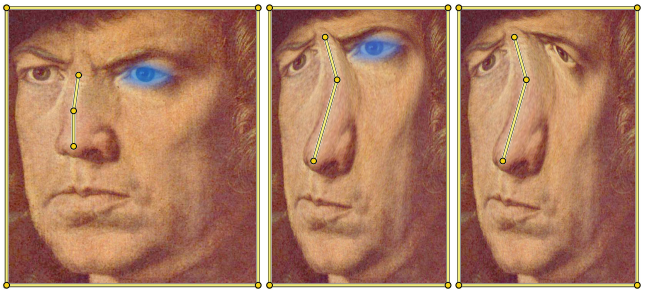

Figure 5: *The generality of our optimization framework makes it possible to compute weights that respect salient object features. In this example, marking a region preserves the shape of an eye (middle), which would otherwise be distorted (right). Interaction time remains fast and fluid because deformations are still the result of a weighted combination of prescribed transformations.*

$$\begin{aligned} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \|\Delta w_j\|^2 dV &\approx \sum_{j=1}^{m} \frac{1}{2} (M^{-1} L \mathbf{w}_j)^T M (M^{-1} L \mathbf{w}_j) \\ &= \frac{1}{2} \sum_{j=1}^{m} \mathbf{w}_j^T \left(L M^{-1} L \right) \mathbf{w}_j. \end{aligned} \tag{8}$$

We impose the constraints (3)-(6) using the discretized handles. To discretize the additional shape-preservation energy term (7), we employ the linear FEM gradient operator G (see its derivation in [Botsch et al. 2010]). $G\mathbf{w}_j$ is a vector of stacked gradients, one gradient per element (triangle in 2D and tetrahedron in 3D; since we deal with linear elements, the gradient over an element is constant). Let R be a diagonal matrix containing the integrals of the user brush ρ over each element and let $\bar{M}$ be the Voronoi per-element mass matrix (i.e., for each triangle/tet, $\bar{M}$ contains the Voronoi portion of its area/volume). Then the energy term in (7) is discretized as

$$\sum_{j=1}^{m} \frac{1}{2} \int_\Pi \rho \|\nabla w_j\|^2 dV \approx \sum_{j=1}^{m} \frac{1}{2} \mathbf{w}_j^T \left(G^T R \bar{M} G \right) \mathbf{w}_j. \tag{9}$$

Note that the matrix $G^T R \bar{M} G$ is a kind of weighted linear FEM Laplacian, and therefore its sparsity pattern is a subset of the main energy matrix $L M^{-1} L$, creating no new non-zeros. Hence adding this energy term does not increase the optimization complexity.

We use Triangle [Shewchuk 1996] for 2D constrained Delaunay meshing and TetGen [Si 2003] for constrained tetrahedral meshing to create the discretized domains. In 2D, we configure Triangle to create triangles of near uniform size and shape. For all our 2D examples, Triangle takes less than a second, even for detailed images which require pixel-size triangles. In 3D, we configure TetGen to create rather graded tet meshes to reduce complexity (Fig. 13); for the *Armadillo* mesh of 43,234 vertices and 120 vertices sampled internally along bones, the resulting tet mesh has 46,898 vertices. For the *Armadillo* and all our 3D examples, TetGen takes a few seconds.

The energy terms in (8) and (9) are quadratic in the unknowns $\mathbf{w}_j$ and convex, and (3)-(6) are linear equality and inequality constraints. We use MOSEK [Andersen and Andersen 2000] as a sparse quadratic programming solver to compute the weights for all of the handles simultaneously.

Since the time necessary to solve the quadratic program is super-linear in the number of unknowns, dividing it into several smaller subproblems allows for a significant speedup. Notice that the optimization of each handle is independent from the rest if we drop the

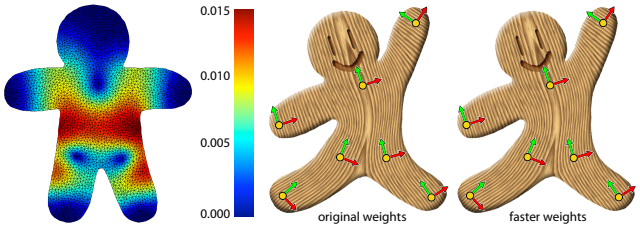

Figure 6: *Dropping the partition of unity constraint (5) greatly optimizes the precomputation of our weights without losing quality. Left: the mean absolute difference between the original and faster weights at each vertex, over all handle weights at that vertex. Deformations with the same handle configuration using our original (middle) and faster (right) weights are visually indistinguishable.*

partition of unity constraint (5). We have implemented this strategy, solving for each $\mathbf{w}_j$ separately and then normalizing the weights for each vertex in a postprocess. We have observed mostly negligible average differences between this faster solution and the original one, often resulting in visually indistinguishable deformations (see Fig. 6). Larger differences occasionally occur far from handles, but the weights have the same qualitative behavior: smoothness and observed local support/lack of spurious local maxima. For the *Gargoyle*, for example, computing the weights for 7 handles separately is 50 times faster than simultaneously. We report the timings as well as the difference between the original and these faster weights in Sec. 4.

Once the weights are computed, the deformation itself is real-time even for very large meshes, since it is computed with a GPU implementation of linear blend skinning (1).

Specifying handle transforms. In the cage-based systems of the various barycentric coordinates methods, the only inputs are the translations of the cage vertices. In our system, the user provides a full affine transformation at each handle. Depending on the application the user may choose to specify only translations, i.e., identity rotations and scales. However, non-trivial rotations are often necessary to achieve a desired effect, and these could be tedious to specify manually. We found it easier to have rotations inferred from the user-provided translations. To do this, the user supplies a set of *pseudo-edges* between point handles (Fig. 7). When the user translates a handle, its rotation is computed automatically as the average of the smallest rotations that take each pseudo-edge incident on the handle from its rest orientation to its pose orientation. In 2D these rotations are averaged as signed angles, in 3D as quaternions. Note that the pseudo-edges are in no way related to the computation of bounded biharmonic weights. They are merely user interface devices that assist the specification of rotations for a set of point handles.

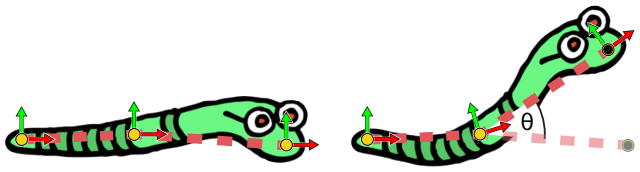

Figure 7: *User-provided pseudo-edges between point handles (left) allow rotations to be inferred automatically from translations. When the user translates handles, these pseudo-edges define rotations between their rest and pose orientations (right). Each handle receives the average of rotations defined by incident pseudo-edges.*

4 Results

Bounded biharmonic blending combines intuitive interaction with real-time performance. Its controls unify three different interaction metaphors so that simple tasks remain simple and complex task become easier to achieve.

Experiments. Points are a particularly elegant metaphor for manipulating flexible objects [Igarashi et al. 2005]. Although similar transformations could be accomplished with bones, Figures 8 and 12 illustrate the simplicity of direct point manipulation of supple regions and highlight the inappropriateness of using rigid bones for the same task. In contrast to previous techniques [Igarashi et al. 2005; Joshi et al. 2007], our approach deforms shapes smoothly even when the handle transformations are large. Fig. 9 illustrates the importance of smoothness to minimize texture tearing.

We have observed our weights to be local and free of spurious local maxima in all examples tested. Fig. 16 compares the support regions of our weights to the biharmonic functions of [Botsch and Kobbelt 2004], which are globally supported and contain many local extrema.

Some tasks are more easily accomplished by controlling both points and lines. Fig. 10 demonstrates our weights with smooth point-based warping while the external cage maintains or resizes the image boundary. Cages are ideally suited for precise area control. In Fig. 11, we use an arbitrary collection of open and closed lines to manipulate the shape and orientation of the tower. These deformations and fine adjustments, needed to account for perspective distortions, are difficult to achieve with points or lines alone.

Our approach generalizes naturally to 3D. At bind time, the optimization distributes the weights over the volume so that linear blending delivers smooth deformation at run time. This scheme ensures real-time performance and low CPU utilization even for high-resolution meshes. We note that cages can be even more tedious to setup in 3D than in 2D, particularly when they are required to envelop objects fully. For tasks such as hand manipulation shown in Fig. 14 (left), skeletons are easier to embed and use to manipulate a 3D object. Skeletons still suffer from joint-collapse problems and lack the precise volume control offered by cages and our approach supports and simplifies the combined use of bones and cages. In

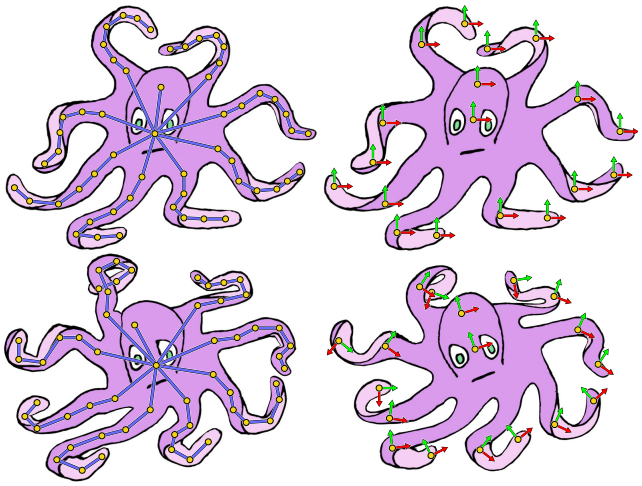

Figure 8: *Flexible objects could be deformed with bones, but points are quicker and easier to specify. At best, supple deformation requires a skeleton with many bones, but these can be difficult to control even with inverse kinematics and are often still too rigid.*

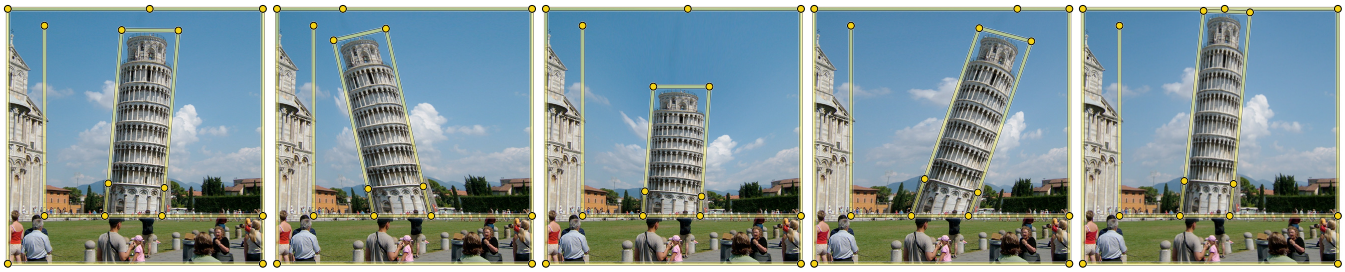

Figure 11: *Deformation of the leaning tower (original is shown left). Cages provide more exact control over area than other handle types.*

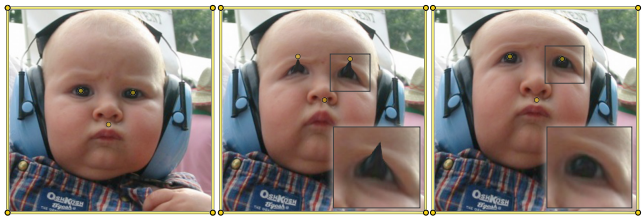

Figure 9: *Weights must be smooth everywhere, especially at internal handles, which are likely to correspond to important features. Weights that have discontinuities at handles, like Harmonic Coordinates (center), introduce tearing artifacts with even slight changes in the handles. Our weights are smooth, as shown on the right.*

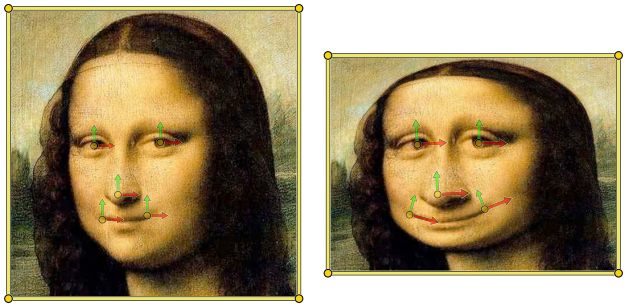

Figure 10: *Points handles deform the image by blending the affine transformations specified at each point. The cage on the boundary maintains the rectangular image shape or allows its resizing.*

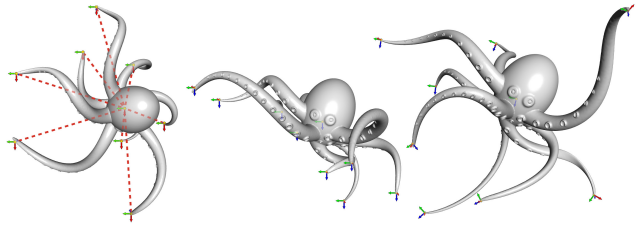

Figure 12: *Using point handles to manipulate the flexible* Octopus. *Pseudo-edges are shown in their rest state on the left.*

	$\lvert\mathcal{S}\rvert$	$\lvert\Omega\rvert$	BT/h	E_{mean}	E_{max}
Gingerman		5,040	0.1397	4.3e-3	5.8e-2
Frowny		5,442	0.0906	4.5e-3	9.0e-2
Alligator		7,019	0.1779	1.3e-3	5.5e-2
Pisa		12,422	0.3174	2.5e-3	6.0e-2
Mona Lisa		32,258	1.2417	5.0e-3	1.1e-1
Gargoyle	20,000	46,003	1.1939	4.3e-3	1.8e-1
Hand	28,692	51,263	3.1268	2.0e-3	3.7e-1
Mouse	26,294	112,355	8.4464	4.1e-3	1.1e-1
Armadillo	86,442	142,073	12.0870	4.1e-3	4.0e-1

Table 2: *Statistics for the various examples in the paper.* $\lvert\mathcal{S}\rvert$ *is the number of triangles of the input 3D model,* $\lvert\Omega\rvert$ *is the number of elements in the discretization of* Ω, BT/h *is the bind time per handle in seconds.* E_{mean} *and* E_{max} *are respectively the mean and max absolute difference between our original weights with (5) enforced explicitly and our faster weights where each handle's weights are solved independently and then normalized. Mean and max values are taken over both handles and vertices.*

particular, our approach supports partial cages that control parts of the object but are not required to surround it fully. Fig. 14 (right) shows a simple cage used to enlarge the belly of the *Mouse*. As always, partial cages can be combined with points and bones, and this combined use of all three metaphors is often the most powerful.

Fig. 13 shows combined use of points and skeletons. We create a sequence of poses of the *Armadillo* by embedding a human skeleton. The skeleton does not control the tail or the ears so their deformations need to be adjusted. This is done most directly by attaching a few points. Direct surface manipulation makes it easy to bend the tail into a more realistic pose and expressively curl the ears. Likewise in Fig. 15, point handles are a natural and simple choice of control for stretching and bending the wings of the *Gargoyle*. Bounded biharmonic weights combine the motion of the skeleton and configuration of these points to yield smooth deformations.

Discussion. We have tested our method on a MacPro Quad-Core Intel Xeon 2.66GHz computer with 8GB memory. The bind time measurements of our unoptimized code are reported in Table 2. One limitation of our solution is the optimization time needed to compute the weights at bind time. We discretized the problem using linear FEM, although other choices may be more efficient, such as the multiresolution framework used in e.g. [Botsch et al. 2007; Joshi et al. 2007]. Generating bounded biharmonic weights in 3D requires a discretization of the volume. Note that once a volume is computed, an arbitrary embedded object (e.g., polygon soup) may be deformed without regard for its topology.

Our bounded biharmonic weights do not have the linear precision property, i.e., they do not necessarily reproduce linear functions. This property is necessary for cage-based deformations (e.g. [Ju et al. 2005; Joshi et al. 2007]) that apply the deformation solely by interpolating the positions of cage vertices because otherwise, they would distort the shape when the cage is rotated. In contrast, our approach allows arbitrary transformations to be supplied at the handles and blends them over the shape; we therefore do not have to rely on linear precision to be able to work with rotations. A com-

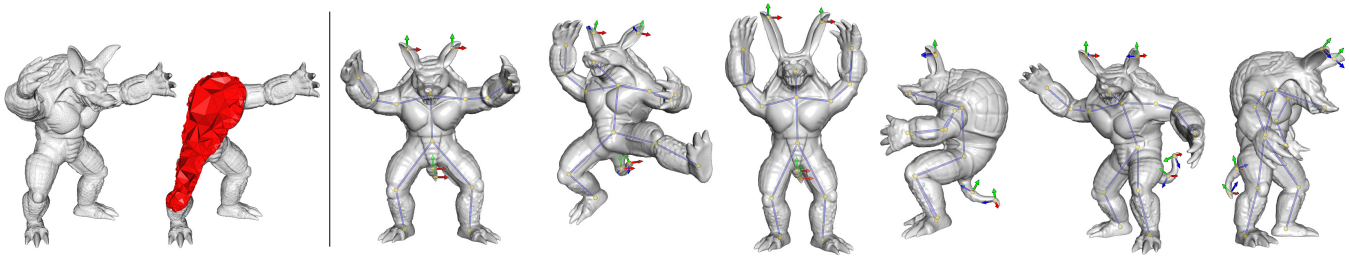

Figure 13: *This example uses a human skeleton embedded in the* Armadillo*. The skeleton does not control the tail or the ears, but points are easily added to control them for additional expressiveness in each pose. Left: a cutaway of the* Armadillo *shows the graded tet mesh produced by TetGen. The inner tetrahedra are much larger than the surface ones, which keeps the discretization complexity reasonable.*

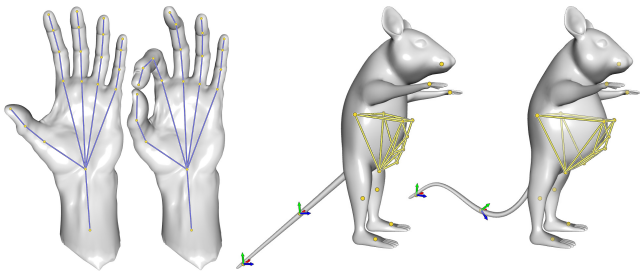

Figure 14: *Cages that fully envelop 3D objects such as the* Hand *are difficult to setup. Skeletons are often easier to embed and manipulate. When cages are needed for precise volume control, our scheme makes them easier to use by allowing partial cages that only overlap parts of the object.*

parison for translational deformation between the linearly-precise Harmonic Coordinates [Joshi et al. 2007] and our cages is shown in Fig. 17 with a rectangular image.

We have only experimented with linear blending deformations based on Eq. (1) in this paper; however, our weights could be used with more advanced methods of transformation blending, such as Dual Quaternions [Kavan et al. 2008]. When blending rigid motions this way, the result remains a rigid motion for a fixed set of weights. Among other advantages, this will cause the regions painted for shape-preservation (Sec. 3.3) to move rigidly, while currently we only obtain similarity or affine transformation there.

5 Conclusion

We have shown how to unify all popular types of control armatures for intuitive design of real-time blending-based deformations. This allows users to freely choose the most convenient handles for every task and relieves them from the burden of manually painting blending weights.

In future work, we would like to optimize the efficiency of both bind- and pose-time of our deformations. Aside from looking at alternative discretizations and numerical approaches, further analysis will help to reduce the dimensionality of the quadratic program: the observed locality property implies that the weights vanish on significant portions of the domain, which could thus be removed from the minimization. It would also be interesting to apply the weight reduction technique in [Landreneau and Schaefer 2010].

Skinning-based deformations are prone to foldovers and self-intersections, as the deformation mapping is not always injective. Our method is no exception. Building a reduced model with our weights for simulation and contact handling is worth exploring in this context. We also plan to study the mathematical properties of the bounded biharmonic weights to determine the necessary conditions for which the observed locality and maximum principle hold.

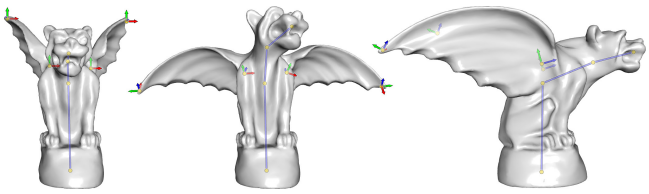

Figure 15: *The* Gargoyle *is deformed using an internal skeleton and point handles.*

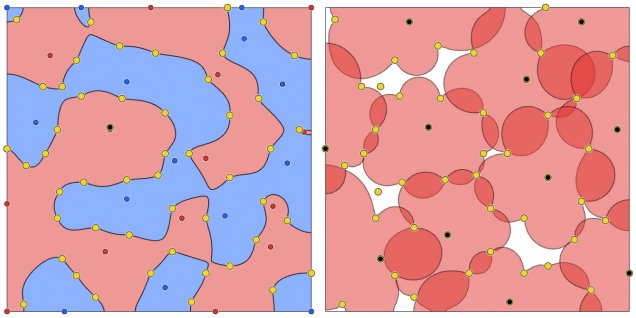

Figure 16: *Fifty point handles (black and yellow) are randomly placed in a square domain. Left: the sign for the black handle's* unbounded *biharmonic weight (red for positive and blue for negative regions). The local maxima and minima are shown as red and blue dots, respectively. Right: the support regions for bounded biharmonic weights of the black handles. In this and all other tested examples, the weights are local and have no spurious local maxima.*

Acknowledgements

We are grateful to Jaakko Lehtinen, Bob Sumner and Denis Zorin for illuminating discussions and to Annie Ytterberg for her narration of the accompanying video. This work was supported in part by an NSF award IIS-0905502 and by a gift from Adobe Systems.

References

ANDERSEN, E. D., AND ANDERSEN, K. D. 2000. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization*. Kluwer Academic Publishers, 197–232.

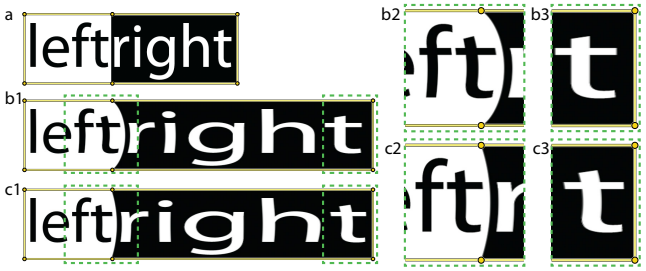

Figure 17: *We show the trade-off between locality and linear precision within a cage. The rest pose of an image of text (a) is stretched horizontally with a cage using Harmonic Coordinates (b1), and our bounded biharmonic weights (c1). A closer look shows that Harmonic Coordinates' deformation has a more global response to the handle translation (b2), whereas our weights are more local (c2). On the other hand, Harmonic Coordinates maintain the vertical lines in letter T near the deformed handles (b3), while our weights reveal their lack of linear precision (c3).*

AU, O. K.-C., FU, H., TAI, C.-L., AND COHEN-OR, D. 2007. Handle-aware isolines for scalable shape editing. *ACM Trans. Graph. 26*, 3, 83.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. 26*, 3, 72:1–72:8.

BEN-CHEN, M., WEBER, O., AND GOTSMAN, C. 2009. Variational harmonic maps for space deformation. *ACM Trans. Graph. 28*, 3, 34:1–34:11.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE TVCG 14*, 1, 213–230.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. PriMo: Coupled prisms for intuitive surface modeling. In *Proc. SGP*, 11–20.

BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. 2007. Adaptive space deformations based on rigid cells. *Comput. Graph. Forum 26*, 3, 339–347.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

DER, K. G., SUMNER, R. W., AND POPOVIĆ, J. 2006. Inverse kinematics for reduced deformable models. *ACM Trans. Graph. 25*, 3, 1174–1179.

FLOATER, M. S. 2003. Mean value coordinates. *Computer-Aided Geometric Design 20*, 1, 19–27.

HORMANN, K., AND SUKUMAR, N. 2008. Maximum entropy coordinates for arbitrary polytopes. *Comput. Graph. Forum 27*, 5, 1513–1520.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

JACOBSON, A., TOSUN, E., SORKINE, O., AND ZORIN, D. 2010. Mixed finite elements for variational surface modeling. *Comput. Graph. Forum (Proc. SGP) 29*, 5, 1565–1574.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3, 71:1–71:9.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KAVAN, L., COLLINS, S., ZARA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4, 105:1–105:23.

LANDRENEAU, E., AND SCHAEFER, S. 2010. Poisson-based weight reduction of animated meshes. *Comput. Graph. Forum 29*, 6, 1945–1954.

LANGER, T., AND SEIDEL, H.-P. 2008. Higher order barycentric coordinates. *Comput. Graph. Forum 27*, 2, 459–466.

LIPMAN, Y., KOPF, J., COHEN-OR, D., AND LEVIN, D. 2007. GPU-assisted positive mean value coordinates for mesh deformations. In *Proc. SGP*, 117–124.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph. 27*, 3, 78:1–78:10.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Graphics Interface*, 26–33.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3, 533–540.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM, 517–524.

SHEWCHUK, J. R. 1996. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 203–222.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3, 81:1–81:10.

SI, H., 2003. TETGEN: A 3D delaunay tetrahedral mesh generator. `http://tetgen.berlios.de`.

SIBSON, R. 1981. *Interpolating multivariate data*. John Wiley & Sons, ch. A brief description of natural neighbor interpolation, 21–36.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. SGP*, 109–116.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3, 80:1–80:7.

WEBER, O., AND GOTSMAN, C. 2010. Controllable conformal maps for shape deformation and interpolation. *ACM Trans. Graph. 29*, 4, 78:1–78:11.

WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Comput. Graph. Forum 26*, 3, 265–274.

WEBER, O., BEN-CHEN, M., AND GOTSMAN, C. 2009. Complex barycentric coordinates with applications to planar shape deformation. *Comput. Graph. Forum 28*, 2, 587–597.