SIGGRAPH Course 2014 — Skinning: Real-time Shape Deformation

Part II: Automatic Skinning via Constrained Energy Optimization

Alec Jacobson

Columbia University and ETH Zurich

This document is meant to be a *living* document (though, perhaps not an immortal one). This version was compiled on 8 December 2015. If you find clumsy typos or audacious mistakes, please email alecjacobson@gmail.com.

1 Introduction

The traditional character skinning pipeline is labor intensive. We may break the pipeline into three main steps to see where professional riggers and animators spend the most time, and consequently where modern automatic methods will take over or assist. Let us recall the linear blend skinning formula so we may track how each step affects a character's deformation (see Figure 1). The new position of a point $\mathbf{v}'$ on the shape is computed as the weighted sum of handle transformations applied to its rest position $\mathbf{v}$:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}. \tag{1}$$

In the first step, a professional rigger must build a control structure inside, on or around the character. In the most traditional case, this control structure is a hierarchy of rigid bones (directed line segments) referred to as a skeleton. We will see how automatic skinning methods invite new control structures such as cages, loose points, or selected rigid regions. We will refer to the set of bones, points, cage vertices etc. as *handles*. The set of n handles defines the range of the sum in Equation (1).

Second, the rigger paints skinning weights for each bone. This process is iterative, combining expertise with trial and error. Riggers paint by adding or subtracting weight values, check the effect on a set of canonical pose, adjust the weights, smooth the weights, and repeat. Furthermore, this process is typically conducted using a 2D mouse interface over a perspective projection on a 2D display: many view adjustments interrupt and prolong the painting process. These weights are defined for each handle j at any point $\mathbf{v}$ on the shape and enter inside our summation as $w_j(\mathbf{v})$ in Equation (1).

Finally, the animator poses the character by applying transformations to each handle. In the traditional case of a skeleton, these transformations are often stored as relative rotations relating the change of each bone from its "rest" or "identity" pose to its current pose in the basis of that bone's parent. The absolute transformation may be recovered by following the skeleton's forward kinematics tree. In our skinning formula, the absolute transformation is the affine matrix $\mathbf{T}_j \in \mathbb{R}^{3\times 4}$ that takes handle j from its rest pose in world space to its current pose.

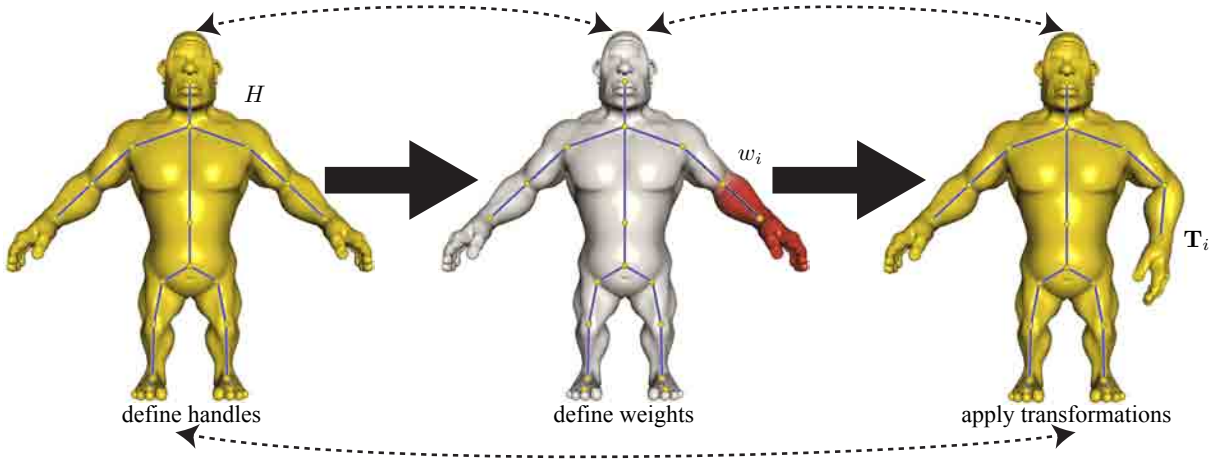


Figure 1: *Though skinning may be broken into three major steps, a professional may iterate between each until satisfied. Modern research seeks to automate each step.*

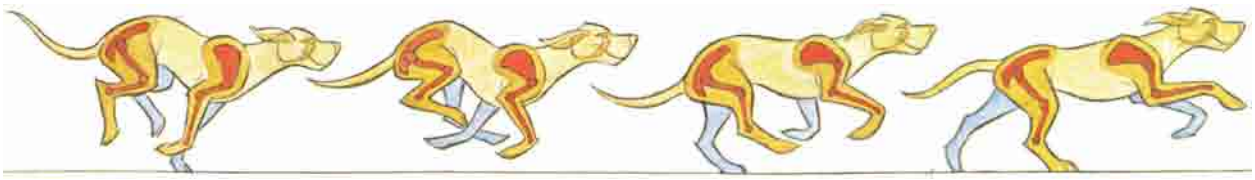


Figure 2: *Traditional animators first sketch an internal skeleton to find a pose, sometimes sketching only in limbs as they imply the remaining shape [Blair 1994].*

In practice, the pipeline is not so serial. Often new poses expose issues with the skinning weights, requiring iterations of adjustments. These adjustments may even reach back to the construction of the control structure. A desired pose requires more degrees of freedom than previously imagined, or the center of rotation at a joint may need to be altered to improve a pose.

Automating each of these stages offers a lower barrier of entry for perspective animators and rapid prototyping opportunities for professional artists. Automation also enables fully automatic procedural animations, where no human labor is required: useful, for example, in crowd simulation (e.g. [Prazak et al. 2010]) or physically-based simulations ([Faure et al. 2011]).

In this chapter, we will see how modern research benefits each major step in the skinning pipeline. We will focus on automatic methods, but also touch on methods offering interactive, semi-automatic systems which trade full autonomy for increased control. The second stage, defining skinning weights, has received the most attention in recent years and therefore will dominate this chapter. A common vein flowing through each stage will be the design of *energies* and *constraints* which translate aesthetic qualities or physical notions into mathematic quantities. The powerful paradigm of constrained energy optimization proves useful for modeling the goals achieved by human riggers and animators in a manner amenable to automatic computation.

2 Automatic skeletons and cages

The locomotion of humans and many other animals depends on their internal skeletons. The arrangement of a creature's bones provides a first-order approximation of its outward appearance. Effects due to muscles beneath the skin (e.g. flexing bicep) [Neumann et al. 2013] and internal pressures (e.g. chest inhaling [Tsoli et al. 2014]) should not be ignored, but for the sake of our discussion we will treat them as secondary. It is no surprise then that the *skeleton*, as a control structure, is nearly as old as computer animation itself (e.g. [Magnenat-Thalmann et al. 1988]). Indeed, even traditional animation books instruct students to first sketch a character's skeleton to find its general pose (see Figure 2) [Blair 1994].

Traditional riggers construct a 3D skeleton as a hierarchy of directed line-segments. This forms a graphical tree, where nodes are dubbed *joints* connecting two or more *bones*. The parent-child relationship is maintained as the skeleton is constructed. Later it will be used to define a forward kinematics tree when determining pose transformations from relative bone rotations.

The problem now is to automatically compute a skeletal tree given only a description of the shape's surface. The shape's geometry and topology imply the geometry and topology of its inner skeleton.

We can separate automatic skeleton construction into two (not quite independent) tasks: determining the skeletal topology and the skeletal embedding. The topological problem asks how many bones make up the skeleton and how they are connected to each other. The embedding problem asks where bones are situation in space, or, equivalently, where bone joints are located within the shape.

Skinning generalizes to other handle types besides skeletons. We will also touch on how cages can be computed automatically, and briefly mention ideas for automatic selection of control points, curves and regions.

2.1 Skeleton extraction

Given a shape—represented for example by its surface in 3D—we would like to find an inner skeleton that captures the geometric and topological features of the shape. More accurately, the skeleton should capture the geometric and topological features of the shape's *articulation*.

However, the articulation is yet unknown, so taken as a subproblem, we can investigate methods that consider only the shape in its *rest* state as input. As output, we expect a tree of line segments embedded in the object.

Subfields of computer vision are dedicated to registering (full or partial) human skeletons to (color or depth) images (e.g. [Shotton et al. 2013]). These methods rely heavily on the prior knowledge that the skeleton has a known topology and even a known general pose. We instead focus on general methods, in particular, those which utilize geometry.

Since an inner skeleton should be *deep inside* the shape, a natural instinct is to look toward the medial axis: the set of points having more than one closest point on the shape's surface [Bloomenthal and Lim 1999]. In 2D, a non-degenerate medial axis is one-dimensional. However, in 3D the medial axis will contain surface-like 2D features. Thus, direct use of the medial axis as a skeleton is not possible. It must be post-processed to recover a one-dimensional skeleton. In practice, medial axis computation is further obstructed by its susceptibility to noise.

Older methods attempted to approximate the medial axis while ensuring curve-like features (e.g. [Wade and Parent 2002]). Liu et al. [2003] connect nearby local minima of electrical force magnitude field defined inside the shape. Cornea et al. [2005] use a similarly motivated electrical force field and find both critical points and points of negative divergence. These methods voxelize the domain and cast a trade-off between computational complexity and accuracy.

An interesting alternative is to look at the skeleton from a topological point of view [Aujay et al. 2007]. Here the skeleton tree is derived from the Reeb graph of a harmonic function seeded by user-selected root and extremity points. The Reeb graph of a function connects a maximum (root node) to saddles (internal nodes) and then to minima (extremity nodes) [Aujay et al. 2007]. Aujay et al. design a harmonic function to interpolate user constraints and form a Reeb graph, but it is too sparse and its geometric interpretation is not meaningful, so heuristics are used to subdivide and embed the graph as a skeleton inside the shape.

Modern attempts at automatic skeleton extraction branch into two main directions: flow-based methods and segmentation-based methods.

Surface flow-based methods A stream of recent methods takes advantage of surface flows to converge on an internal *curved* skeleton. The prototypical method of [Au et al. 2008] notices that a surface undergoing mean-curvature flow tends to degenerate cylindrical parts of the shape into thin lines (see Figure 3). Mean curvature flow simply moves a point $\mathbf{v}$ on the surface opposite its normal direction by an amount proportional to mean curvature at that point:

$$\mathbf{v}' = \mathbf{v} - H\hat{\mathbf{n}} = \mathbf{v} - \Delta\mathbf{v}, \tag{2}$$

where $\mathbf{v}'$ is the point's new position, H and $\hat{\mathbf{n}}$ are the mean curvature and unit normal at $\mathbf{v}$, and Δ is the Laplace-Beltrami operator.

In the discrete setting, we can solve for new vertex positions $\mathbf{V}' \in \mathbb{R}^{n\times 3}$ by solving the linear system:

$$\mathbf{V}' = \left(\mathbf{I} - \mathbf{M}^{-1}\mathbf{L}\right)\mathbf{V} \text{ or } \mathbf{M}\mathbf{V}' = \left(\mathbf{M} - \mathbf{L}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{I}, \mathbf{M}, \mathbf{L} \in \mathbb{R}^{n\times n}$ are the sparse identity, mass (area), and discrete Laplacian matrices.

One can understand mean curvature flow as the gradient of Dirichlet energy of the shape's positions. That is to say that mean curvature flow minimizes surface area. Thus, we expect the shape to *shrink* inward. The linear system solve, is then understood as an implicit integration in this gradient direction.

If we continued flowing indefinitely, a sphere-topology surface would converge to a single point (e.g. zero area). Further, standard discretizations of the Laplacian will become non-singular for badly shaped or scaled triangles. To recover a meaningful skeleton and avoid numerical problems, Au et al. propose regularizing the flow and interrupting each iteration with a clustering stage which isolates near one-dimensional features of the surface and identifies "joints" where more than two such curves converge.

The modified mean curvature flow of [Au et al. 2008] is not guaranteed to *stay inside* the shape. This is of course important for defining an *inner* skeleton. Modern extensions of this method will converge to the medial axis [Tagliasacchi et al. 2012]. As a network of one-dimensional objects, their skeleton is a *subset* of the two-dimensional medial axis. By definition the medial axis, is inside the shape so we can be guaranteed that the skeleton is actually inside and even well-centered.

Various extensions of flow-based skeleton extraction methods have appeared, for example for point clouds [Cao et al. 2010; Huang et al. 2013]. Wang & Lee [2008] propose a flow ostensibly similar to [Au et al. 2008], albeit with a different intuition: balance between uniformly shrinking the shape's volume and pulling its surface back toward its previous positions. The relationship to a continuous flow such as mean curvature is not obvious, but a similarity in formulation and results is clear.

In any case, taken directly, "curve-skeleton extraction" is a fundamentally different problem than rigid skeleton extraction for skinning. Though ostensibly similar to animation skeletons, curve skeletons are meant as a compact, abstract representation of the shape. Thus, they require *post facto* processing to build a *rigid* skeleton of straight line segments common to animation, or requires switching to non-standard skinning methods accommodating curve skeletons [Forstmann and Ohya 2006; Yang et al. 2006; Forstmann et al. 2007; Jacobson and Sorkine 2011; Öztireli et al. 2013].

Figure 3: *Successive Laplacian smoothing and joint clustering converges to an internal skeleton [Au et al. 2008].*

Segmentation-based methods Another class of methods finds internal skeletons via shape segmentation. Katz and Tal segment an input shape based on a heuristic to find cuts between segments near "deep concavities," then skeletal joints are simply placed at cut loop centroids and connected with straight line segments to form a tree [2003]. There is no guarantee that skeletal segments (or even joints for that matter) will lie *inside* the shape.

More recently, Bharaj et al. present a special-case segmentation-based method, which assumes the input model is made up of many small multi-connected components [2012]. In this case, the input is first segmented according to connected components with respect to the triangle mesh's dual graph (two triangles sharing an edge are connected). Then each component becomes a node in a *contact graph*, in which two components' nodes are connected if the components are in contact (i.e. their spatial intersection is nontrivial). In practice, "contact" is determined by bounding box overlap and checking if any samples of one component are within some ϵ of any samples of the other. Nodes are placed geometrically at the centroids of each component.

In general, this graph will be too dense to be used directly as a control skeleton. Bharaj et al. propose an edge-collapse cost function similar to the quadratic error metrics used in mesh simplification. For a potential edge collapse, the cost includes a term to penalize collapses that place new joints far from neighbors and a term to penalize collapses that create long bones.

Feasibility constraints (a.k.a infinite costs) are added to ensure a well formed skeletal tree. An edge ij is *collapsible* if the joints at i and j are valence > 2 (to prevent collapses along chains), if there exist more than one path from i to j (to ensure a single skeletal tree), if the shape components at i or j are very small or if the length of ij is small (to encourage collapsing small bones corresponding to small parts of the shape).

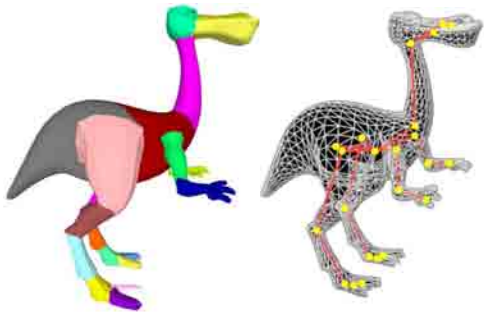


Figure 4: *Classification into rigid parts implies a skeleton [Katz and Tal 2003].*

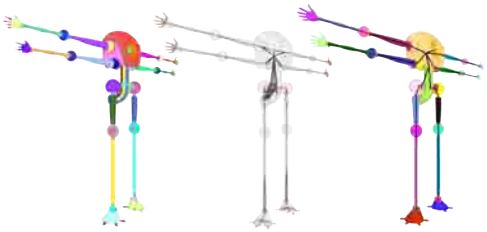


Figure 5: *The connected components form a natural segmentation (left), and a graph representing inter-contacts (middle) may be collapsed to a more suitable skeleton (right) [Bharaj et al. 2012].*

This optimization mirrors methods in image processing and computer vision based on oversegmentation (e.g. [Taguchi et al. 2008]). In those works, the hope is that by oversegmenting the image into a manageable number of patches, a structure (e.g. depth layers, object classification, object pose) can be recovered by analyzing and often joining neighboring patches. In [Bharaj et al. 2012], the assumption is that the components used to model the shape already provide a suitable oversegmentation. Adapting this technique for single component shapes or for shapes that do not meet this assumption is an open problem.

The quality of segmentation-based methods depends heavily on the segmented parts' correspondences to regions of the shape likely to be rigid in the yet unknown deformation. This implies an ideal optimization problem where the quality of a segmentation is measured directly with respect to the rigid motions its implied skeleton would induce [Huang et al. 2009]. This is a strikingly different goal from the usual goal of shape segmentation, which is to extract a high-level understanding of a shape or set of shapes via abstraction (e.g. [Kalogerakis et al. 2010; van Kaick et al. 2014]).

The problem of skeleton extraction is easily classified as ill-posed. The deformation is not known at the time of extraction. In the next section, we consider methods that overcome this by assuming that a skeletal topology is given as input [Baran and Popović 2007a; Bharaj et al. 2012]. Other methods assume that example poses of the input shape are given [Schaefer and Yuksel 2007; Le and Deng 2014]. For this survey, we will exclude discussion of example-based methods and focus on automatic methods with only the rest shape as input.

2.2 Embedding an existing skeleton

Automatic skeleton extraction is necessary in situations when no prior information is known about the input shape. However, in many cases, the type of shape (e.g. human, quadruped, five-digit hand) is known or can be automatically determined. The most common case in computer graphics is—of course—when the shape is a human or humanoid, as is the case for retargeting motion capture data. In other scenarios, it is plausible that the user can quickly determine the topology and some vague geometric information of the skeleton, relying on an automatic method to embed the skeleton inside the shape (e.g. [Miller et al. 2010; Jacobson et al. 2014]). A related problem to automatic rigging is automatic rig transfer, where the input is an *unrigged* shape A and a rig for a similar shape B and the output is a rigging for shape A. Therefore, a common subproblem between automatic rigging and rig transfer is embedding an existing skeleton. Here we will assume the skeleton topology is fixed and only the geometry is variable.

Posed as an optimization problem with joint positions as continuous degrees of freedom in $\mathbb{R}^3$, skeleton embedding is a difficult, perhaps intractable problem. The ideal energy would need to balance many desires. In the case where the input skeleton already has a meaningful geometric pose, Baran & Popovic [2007a] write that this ideal optimization would "compute the joint positions such that the resulting skeleton fits inside the character as nicely as possible and looks like the given skeleton as much as possible."

There are two issues here. How can we translate these desires from English into mathematic quantities amenable to optimization? And even so, how can we tractably optimize over all possible joint locations? Baran & Popović propose solutions to both questions.

They design an energy function composed of a weighted combination of nine independent measures of the quality of a given skeleton, as parameterized by its joint positions $\mathbf{v}_1, \ldots, \mathbf{v}_m$:

$$E(\mathbf{v}_1, \ldots, \mathbf{v}_m) = \sum_{i=1}^{9} \lambda_i E_i(\mathbf{v}_1, \ldots, \mathbf{v}_m). \tag{4}$$

The nine measures attempt to:

1. avoid short bone chains,
2. maintain angles between bones,
3. keep lengths of symmetric bones proportional,
4. avoid overlapping bones,
5. place special "feet" bones below others,
6. avoid zero-length bones,
7. maintain bone orientations,
8. place extremities close to surface, and
9. avoid disparities in Euclidean and bone-path distances [Baran and Popović 2007b].

Even with such measurable quantities it is not obvious how to weigh the importance of one over the other. Baran & Popović propose a practical approach: manually build a dataset of good and bad embeddings and *learn* good weights λ_i. Deriving an intuitive and hopefully more compact model for skeletal embedding remains an open problem. In particular, the measures here are purely geometric heuristics which do not directly account for the embedded skeleton's implied deformations on the shape.

With this composite energy in tow, the task now is to find optimal joint positions. Unfortunately the energy is non-linear and discontinuous: traditional continuous optimization would be intractable. Instead, Baran & Popović opt to limit the search space to a finite set of possible joint locations. Motivated by the observation that the medial axis of a smooth surface represents the set of points where the signed distance function has discontinuous gradient, they build a discrete distance field inside the shape. Locations of approximate derivative discontinuity are found at interfaces between neighboring octree cells with gradients differing by $> 120°$ (see Figure 6).

The number of possible locations is still large. To prune, Baran & Popović sort locations based on distance and greedily keep only locations which fall outside spheres centered at previously kept locations with their distance as radius. This leads to a sparse "sphere packing", whose centers will be possible joint locations.

To efficiently search over these locations, a graph is constructed connecting nearby centers (roughly those with overlapping spheres). Baran & Popović search for the best embedding using a modified A* algorithm over the tree of possible embeddings. They reject possible embeddings early on if the lower bound on the embedding's energy is high. As more of the energies involve high-valence joints, this implies placing high-valence joints first in order to prune bad paths often early on.

Notice that the energies above did not include any general measure of "centeredness" or distance to the medial axis. However, by reducing the search space to only consider locations near the medial axis, the solution is directly constrained to be well centered.

This combinatorially optimized discrete embedding is not ideal. The skeleton often has generally the right shape, but joint locations are *quantized*, snapped to awkward positions. To refine the positions, Baran & Popović propose a continuous optimization using a

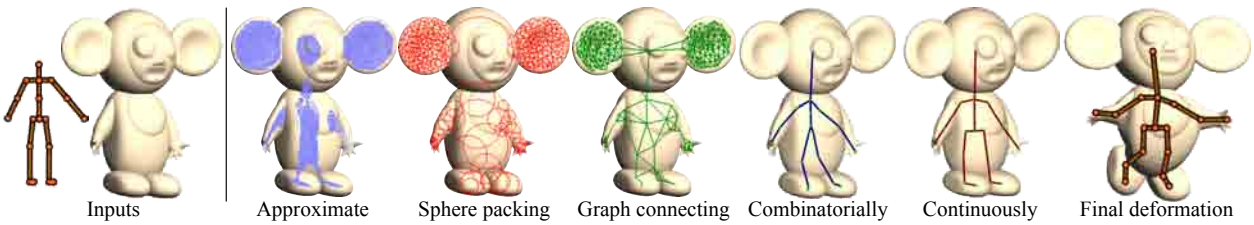


Figure 6: *An input skeleton is embedded in an input character by first optimizing joint positions over a discrete number of positions on the approximate medial axis. Then a continuous optimization refines the placement and the embedded skeleton may be rigged to animate the model [Baran and Popović 2007a].*

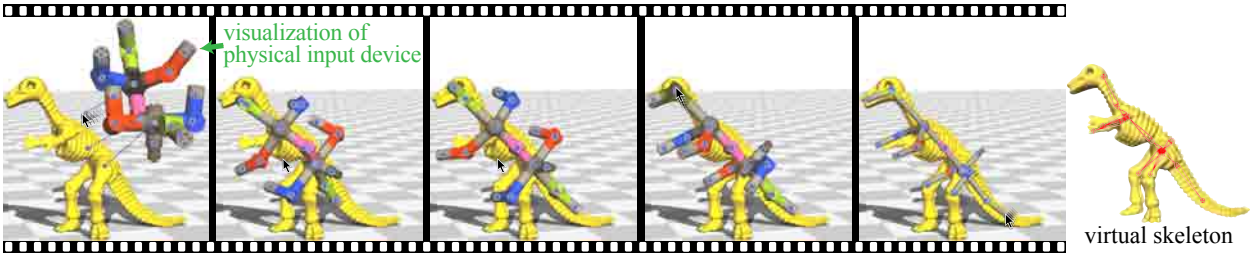


Figure 7: *In a special scenario, a physical input device (which fixes angles and topology but not bone lengths) is bound to shape.*

modified subset of the previously defined energies. This time an energy must be added to pull locations toward the medial axis, as otherwise joints could drift away from it. As this energy is simpler, following gradient-descent leads quickly to a local minimum.

Aggregating heuristics has a tendency to aggregate flaws. There is no reason *a priori* to believe this complicated process is an exception. In particular, two rounds of optimization with different objectives raises the question of which measure is correct?

The pipeline of [Baran and Popović 2007a] indeed produces well embedded skeletons when the input shape closely matches the input skeleton, particularly in the case of a humanoid shape in a *Vitruvian Man* pose: feet flat on the ground, legs straight and arms extended. This is not of little significance as professional modellers tend to create characters in such poses, not coincidentally.

Again, we could easily classify the problem of skeleton embedding as ill-posed. One must either make heavy assumptions (as [Baran and Popović 2007a] or computer vision techniques e.g. [Shotton et al. 2013]) or ask the user for assistance to disambiguate solutions. Nowhere is this more clear than when embedding *partial* skeletons. As input we have a skeletal topology for what should only be a subset of the shape's overall skeleton. Without more information it is difficult to argue for a measure of *best* placement in the shape over many feasible positions, orientations and scales. The same combination of two bones could be used to control a centaur character's left shoulder, right shoulder, front leg, hind leg, neck, or thumb, *inter alia*.

In the case of [Jacobson et al. 2014], such a situation arose from the invention of a skeletal animation input device. The physical hand-held device defines a skeleton topology and skeletal geometry *up to* the scale of each bone and a global rigid transformation. That is, the skeletal hierarchy is fixed, but only the relative orientations between bones are known. To overcome this challenging embedding problem, Jacobson et al. require the user to hint at the locations of the joints. Then a non-linear optimization routine finds the *best* orientation, position and bone lengths (see Figure 7). From the viewpoint of fully automatic methods, this is far from ideal. Requiring less or no user interaction remains an open problem. One possible direction is to leverage metrics depending on the input shape *à la* [Baran and Popović 2007a].

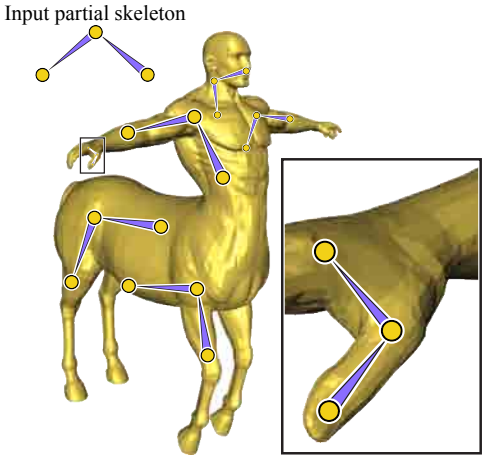


Figure 8: *Embedding a partial skeleton (top inset) is particularly ill-posed: many feasible embeddings exists at different positions, orientations and scales.*

Another special case arises in the automatic rigging pipeline of [Bharaj et al. 2012]. As discussed above, Bharaj et al. extract a skeleton for shapes composed of many connected components. In general, this skeleton will not match those familiar to animators or motion capture output. To accommodate these scenarios, their extraction output and an existing input skeleton are *matched* as graphs. This optimization process minimizes a cost function composed of similar heuristics to those previously mentioned and while satisfying constraints to avoid double assignment. The problem is solved efficiently using dynamic programming.

By leveraging the previously extracted skeleton, the graph matching of [Bharaj et al. 2012] can conveniently work on a reduced problem, no longer directly tied to the original input shape. One could imagine similar pipelines, swapping for example extracted skeletons from [Au et al. 2008] or [Huang et al. 2009]. However, if the goal is to truly embed the input skeleton (rather than just match to it), then ideally an embedded skeleton's quality should be measured with respect to the deformations it implies.

2.3 Automatic cages

Skeletons are not the only control structure. As we shall see in Section 3.2, cage-based deformation is a special case of linear blend skinning. These techniques require a closed polyhedron containing the shape. The cage can be seen as a low-resolution abstraction of the shape. Direct editing of cages entails the user manually moving each cage vertex, so parsimony and "intuitive structuring" is appreciated. We will see that translating "intuitive structuring" into a mathematical quantity is a challenging task.

Generalized barycentric coordinates, the workhorse research field behind cage-based deformation, blossomed in the last decade [Ju et al. 2005; Joshi et al. 2007; Manson and Schaefer 2010]. We will consider these methods in greater detail later (see Section 3.2). For now, it suffices to define a cage-based deformation of a point $\mathbf{v}$ on a shape as:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v})\mathbf{c}'_j, \tag{5}$$

where $\mathbf{c}'_j$ is the deformed position of cage vertex j.

Most works concerning the mathematical construction of generalized barycentric coordinate functions ($w_j(\mathbf{v})$ in Equation (5)), treat the construction of the cage itself as an orthogonal problem. In practice, their cages are constructed manually. In 2D, ease of visualization and simplicity of a shape's (e.g. cartoon's) topology cast manual (and automatic) cage construction as a rather trivial task. In 3D, the "curse of dimensionality" immediately appears and manual cage construction becomes equivalent to 3D shape modelling, albeit without attention to fine details.

We can summarize the desired qualities of a cage with a few criteria. The cage should:

1. adhere closely to the input geometry,
2. contain very few vertices,
3. fully contain the input without interpenetration,
4. respect the topology of the input,
5. contain planar/triangular facets,
6. mirror (intrinsic and extrinsic) symmetries in the input, and
7. place cage vertices to facilitate the desired deformation.

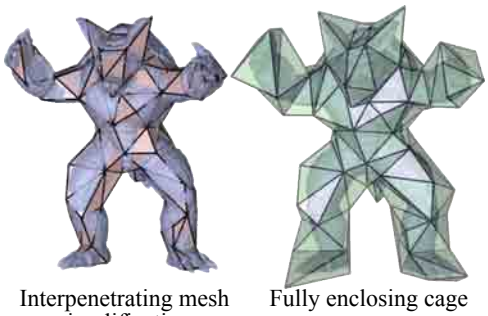


Figure 9: *A simplified mesh might intersect the original, while a cage should fully contain it.*

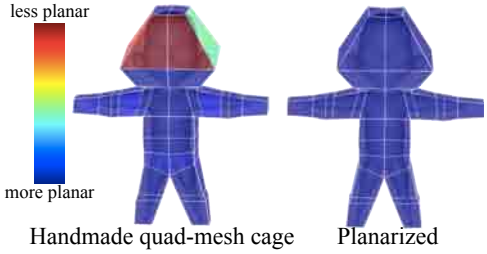


Figure 10: *Some cage-based methods allow quad-mesh cages, but faces must be planar (models courtesy Tony DeRose at Pixar).*

Some criteria are at odds with each other: a coarser cage will necessarily adhere less to the original shape. The last criteria is especially ill-defined. In many cases the desired deformation will not be known beforehand. The best we can hope for is to place cage vertices so that the space of deformations spanned by the cage is meaningful and sufficient for user exploration. This suggests an analog to ideas developed for skeleton extraction, where the skeleton's implied rigid motions were considered [Huang et al. 2009].

Most cage-based deformation methods are defined only for triangular polyhedra (i.e. the cage is a closed, manifold triangle mesh) [Ju et al. 2005; Lipman et al. 2007; Lipman et al. 2008; Hormann and Sukumar 2008; Manson and Schaefer 2010]. In these cases, automatic cage generation methods can leverage general techniques for isocontouring implicit functions [Shen et al. 2004], quadratic-error-metric simplification [Deng et al. 2011], and mesh merging [Xian et al. 2012]. These methods focus on staying close to the input (1) on a specific vertex budget (2) without creating interpenetration (3). Isocontouring an interpolating function such as a signed distance field or the moving least squares function of [Shen et al. 2004] provides means to stay close to the original input. But implicits quickly lose topological and geometric details (see Figure 11). Worse, methods like [Shen et al. 2004] could create *new*, spurious topological features not present in the original input.

Alternatively, Xian et al. [2012] begin by strategically covering the input with oriented bounding boxes. This induces some amount of regularity, but the merging procedure removes any guarantees on the eventual output.

In general, the *freedom* to work with triangle meshes tends to result in more unstructured and denser than necessary cages, especially when compared to man-made quad-dominant meshes.

Some cage-based methods generalize to arbitrary polyhedra [Joshi et al. 2007; Jacobson et al. 2011]. At first glance this flexibility seems to imply that we may now consider cages parameterized by general, polygonal meshes. In the case of manual cages, riggers can design sparse (2) cages using their favorite quad-mesh subdivision tools (e.g. Autodesk's MAYA). However, we require *polyhedra*, that is, shapes with flat faces. Recent research (stemming mostly from architectural modeling) has led to a variety of *planarization* algorithms which find the closest planar-quad mesh to an input quad-mesh [Bouaziz et al. 2012; Poranne et al. 2013; Tang et al. 2014]. These methods efficiently impose planarity constraints, which would be useful for the design of polyhedral cages for shape deformation (see Figure 10).

Similarly, modern advances in symmetry detection and related constraints [Panozzo et al. 2012] should be exploited to ensure that automatic cages respect not only extrinsic symmetry (easier to detect and constrain) but also intrinsic symmetry (i.e. symmetries disguised by the particular pose of the input shape).

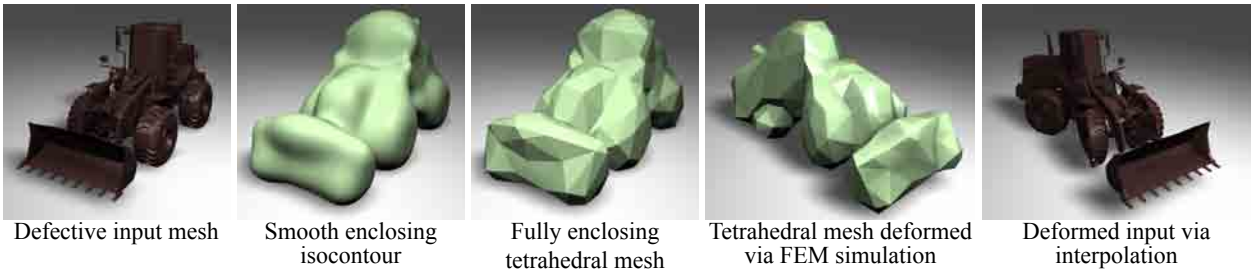


Figure 11: *Isocontouring methods (e.g. [Shen et al. 2004]) may ensure enclosure of the input, but often by compromising either tightness of fit or cage simplicity.*

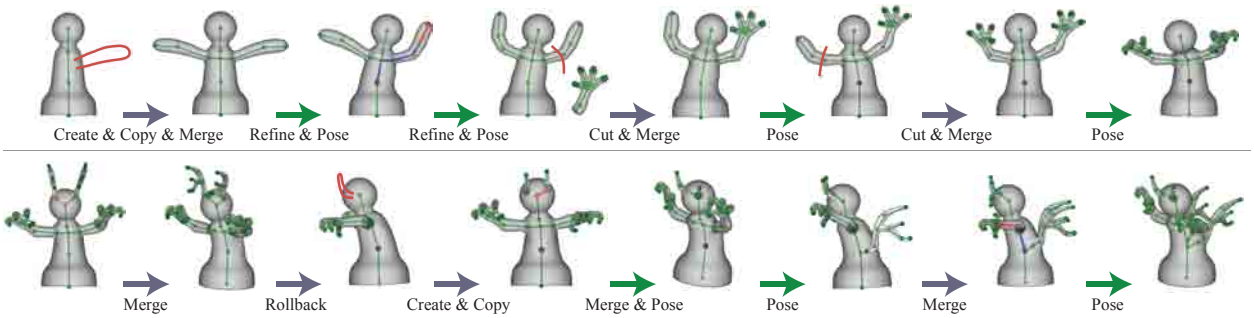


Figure 12: *The woes of skeleton extraction are eliminated when the skeleton is maintained during modeling [Borosan and Jin et al. 2012].*

In general, cages not only reduce the complexity of the deformation description, but also allow flexibility in terms of the representation of the shape *embedded inside* the cage. The shape could be a high-resolution triangle mesh, polygon soup, point cloud or any combination. Some deformation methods use cages for this reason, despite not actually expecting the user to control the cage vertices directly [Ben-Chen et al. 2009b; Ben-Chen et al. 2009a]. The cage is truly just a convenient mathematical discretization of the space transformation's domain. In 2D, if the user-interface related criteria are dropped, we may just take a simplification of the cartoon's alpha channel boundary as the cage [Weber et al. 2009; Weber and Gotsman 2010]. In 3D, designing a reasonably tight-fitting cage remains a tricky problem.

Ben-Chen et al. [2009a] construct cages by simplifying an offset surface. This provides control over how close the cage is to the original surface (i.e. via offset *thickness*), but little control over the number of cage vertices. This trade-off is less critical when the cage is not the user-interface. And perhaps even less critical, is when a "cage" is constructed merely to serve as the boundary of a tetrahedral mesh for finite-element method physically-based simulations [Shen et al. 2004; Jacobson et al. 2013a; Xu and Barbič 2014]. Here, the immediate goal is to create an input mesh for simulation out of *unclean* shape representations (e.g. self-intersections, holes, non-manifold edges, etc., see Figure 11).

The practical impact of cage-based deformation still pales in comparison to skeletal techniques. Part of this inability to transcend academia into industry is due to the difficulty of creating cages. Designing satisfying automatic or semi-automatic methods to alleviate this remains an open problem. Perhaps we can look toward recent advances in sketch-based modeling and quad-mesh design [Takayama et al. 2013].

2.4 Automatic handles during modeling

Many of the problems arising during automatic skeleton or cage construction can be avoided if the control structure is created at the same time as the shape is created. This is a daunting task as it requires maintaining a skeleton or cage throughout all editing operations.

In some sense, subdivision control cages are analogous to those of cage-based deformation. However, as they are tied directly to the finest details of the shape, they are not suitable for immediate use as low frequency deformation cages.

Recently, a full solution was presented in [Borosan and Jin et al. 2012]. Their system is built on top of sketch-based modeling software. Each modeling operation is followed by an update to the skeletal topology and geometry. Skeletons are extracted for new appendages sketched onto the shape. When two shapes are merged, their skeletons are also automatically merged (see Figure 12). By maintaining the skeleton rather than extracting it anew, operations are fast and localized ensuring interactive rates. The importance of interactive performance is paramount—as to not disturb the modeling experience. However, maintaining high framerates comes at the cost of simplifying the overall objective. As a result, skeletons tend to be dense or awkwardly positioned compared to manually

constructed rigs. Nonetheless, this direction is *very* promising and [Borosan and Jin et al. 2012] invites many interesting open problems in simultaneous rig and model creation. Two recent works have demonstrated the advantages for modeling with a skeleton (albeit curved) as a first-class shape representation [Thiery et al. 2012a; Bærentzen et al. 2014].

2.5 Automatic curves, regions and points

There are other useful handle types beyond skeletons and cages. While these see more limited use in traditional skinning applications (i.e. animation and posing), it is worth mentioning a few methods that define handles automatically.

In design, curves define boundaries of smooth parts and intersections with other parts. Curves are a natural handle for *editing* features of given shape to generate a new instance of the same class of shapes [Gal et al. 2009; Schemali et al. 2012]. We may define editing to be, perhaps subtly, distinct from *posing*, which seeks to rearrange an existing instance as if animated with life. For man-made objects, curves may be easily extracted from computer aided design (CAD) shape descriptions. If not available, Gal et al. [2009] identify sharp creases along polygonalized surface edges with dihedral angles $> 40°$, suggesting more robust techniques could be needed for more complicated shapes [Lee and Lee 2002; Ohtake et al. 2004]. Curves play an important role in our perceptual understanding of shapes [Cole et al. 2008], and Schemali et al. leverage this to find relevant curve handles for skinning-like deformation [2012]. Identifying curve handles correctly for editing tasks is seemingly tantamount to reverse engineering the geometric shape itself [Li et al.]. In which case, perhaps a more meaningful parameter set would emerge.

Points are perhaps the easiest handles to place manually and not surprisingly receive the least attention. Yücer et al. use skinning as a subspace reduction for image registration, generating weights around seed points, placed using a variant of farthest point sampling [Yücer et al. 2012]. However, these points are not directly controlled by the user. For points as a user interface, perhaps *Schelling points* (points people tend to focus on in the absence of information) give a reasonable starting point for automatically selecting good deformation point handles [Chen et al. 2012].

3 Automatic weights

Returning to our skinning formula

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}, \tag{6}$$

we have just surveyed various automatic methods for automatically constructing a set of handles. We now assume that the input shape is accompanied by an input handle set (either manually or automatically defined) and turn our attention to the automatic computation of skinning weight functions w_j.

In this section, we will traverse previous methods and maintain a categorization of the desirable qualities for weight functions. This categorization will pave a direct route to automatic computation via constrained energy optimization. Each quality will imply an energy functional or constraint.

Before considering weights resulting from the powerful framework of constrained energy optimization. Let us consider direct, closed-form solutions. These promise far fewer qualities but benefit from fast computation and analytic expression.

3.1 Euclidean-distance based weights

The most basic property we can ask for is interpolation: the weights w_j of handle j should be exactly one along/at handle j and zero along/at all other handles. Intuitive, a *handle*, as a user interface, should directly control the region of the shape it occupies. By satisfying interpolation, we may began to talk in user-interface terms about *direct manipulation* [Shneiderman 1997].

Rigid The simplest automatic weights to satisfy interpolation are so-called *rigid* weights. These weight functions are one at points which are closer to its corresponding handle than all others:

$$w_j(\mathbf{v}) = \begin{cases} 1 & d(\mathbf{v}, h_k) < d(\mathbf{v}, h_k) \; \forall j \neq k, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

where $h_k \subset \mathbb{R}^3$ is the space occupied by handle j and $d(\mathbf{v}, h_k)$ measures the (for now orthogonal Euclidean) distance from $\mathbf{v}$ to the closest point in/on/along handle k.

These weights amount to the piecewise-constant characteristic (a.k.a. binary indicator) functions of the cells of the Voronoi diagram defined by the handle sets. These functions are obviously not smooth, not even C^0, and ignore the input shape. Nonetheless, this method appears in software packages (e.g. MAYA's `Rigid Bind`). Bharaj et al. employ a variant of these rigid weights, where distances are determined *per connected component* of the input model. They assume the input model is made of many small pieces. By assigning each component to a single bone, they may ensure that each piece deforms rigidly under linear blend skinning (although it becomes superfluous to actually evaluating linear blend skinning in the case of rigid weights: there's no blending!).

Inverse distance It follows from an assumption of smoothness that a handle's weight function should then decay from interpolated value of one along that handle toward zero at other handles. Naturally this decay is related to the distance from the evaluation point on the shape to the handles.

We *could* consider linear blend skinning as an instance of a much more general problem of scattered data interpolation. The user (or some procedure) has determined transformations at a sparse set of handles, the goal is then to interpolate this over the rest of the shape. Ignoring any prior information relevant to the shape deformation application (such as the shape itself or that handles may be directly controlled by a user), we could directly apply any *off-the-shelf* scattered data interpolation method.

Indeed, many commercial softwares (e.g. MAYA's `Smooth Bind > Closest Distance`) provide automatic weights based on inverse Euclidean distances. These methods are variants of Shepard interpolation [Shepard 1968].

The basic weights are inversely proportional to a point's distance to the handle:

$$\hat{w}_j(\mathbf{v}) = \frac{1}{d(\mathbf{v}, h_j)^p}. \tag{8}$$

where p controls the smoothness near the handle and the sharpness of decay. Notice, as Shepard first did, that as $p \to \infty$ then $\hat{w}_j$ approach the Voronoi cell characteristic functions above.

If p is finite (usually $p = 2$), then the weights $\hat{w}_j$ must be *normalized* to partition unity (sum to one):

$$w_j(\mathbf{v}) = \frac{\hat{w}_j(\mathbf{v})}{\sum_{k=1}^{m} \hat{w}_k(\mathbf{v})}. \tag{9}$$

In some sense, this normalization voids the intuition behind inverse distance weights. If the original motivation was that weights decay as we consider farther away points, this is no longer true. After normalization, the notion of "farther" becomes relative to the other handles, allowing for unintuitive behavior. Problems occur in regions of the shape that intuitively *belong* to one handle, but are technically distant to all handles (see Figure 13). Far away regions—no matter how well guarded by handles—are in the end similarly distant to all handles: all $\hat{w}_j \to 0$ at the same rate and $w_j \to \frac{1}{n}$. This is a problem not just of locality as it can occur even on local scales: it is also a problem of *monotonicity*. Cutting a 1D cross-section in Figure 13, the inverse distance weight of the right point handle first decreases toward the left handle then increases toward its own handle location then decreases again. Each handle is a global maxima or minima, but there are also local extrema at either end of the cigar. These topological feature signal an issue with these weights: they do not *block* the influence of one another.

There are a wide variety of inverse-distance functions that qualitatively behave similarly to Shepard's original weights. Sumner et al. [2007] truncate inverse squared distances by considering only the k-nearest handles, this ensures sparsity, but does not resolve lack of monotonicity and introduces fairness issues. Dionne & de Lasa [2013] swap approximate geodesic distance for Euclidean distance, making the inverse-distance weights shape-aware. Similarly, [Zhu and Gortler 2007] use a geodesic-inspired distance in the moving least squares deformations of [Schaefer et al. 2006]; and [Levin and Levi 2014] in the radial basis function paradigm of [Botsch and Kobbelt 2005]. Again, the lack of montonicity is not resolved, and even exact geodesics would introduce derivative discontinuities (geodesic distances are only C^0 at *cusps*).

Natural neighbor Figure 13 demonstrates the issue of monotonicity (or the lack thereof). Handles should *own* parts of the shape inaccessible from other handles. This implies that weight functions should be localized and monotonic. These notions are not unique to skinning and see first exploration again in general scattered data interpolation.

Sibson defines natural neighbor coordinates based directly on the Euclidean Voronoi diagram [1980]. First, construct the Voronoi diagram of the handles (assume all handles are points $h_j = \mathbf{c}_j$). The weights at a point $\mathbf{v}$ corresponds to the volume *stolen* from each handle's cell when $\mathbf{v}$ is temporarily added as a new seed in the Voronoi diagram:

$$\hat{w}_j(\mathbf{v}) = \int_\Omega \mathbf{1}_\mathbf{v} \mathbf{1}_j dV, \tag{10}$$

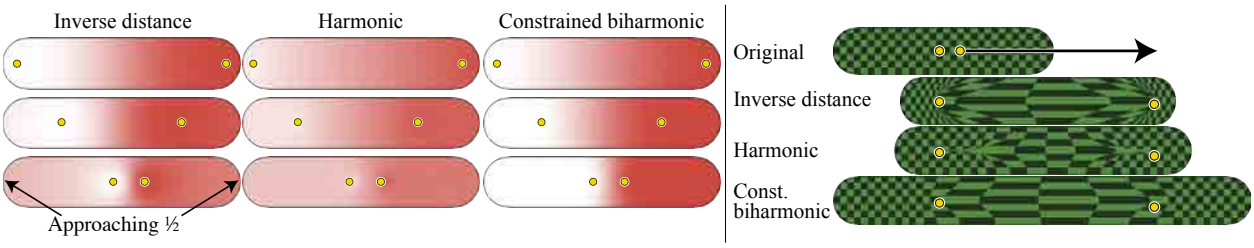


Figure 13: *Left: placing points handles (yellow) progressively closer to each other highlights that inverse distance weights eventually regress toward an average value ($w = 1/2$) in regions distant to all handles. Harmonic weights also show this behavior, while constrained biharmonic weights do not. Right: (using the weights from the bottom row) translating the right point handle unintuitively effects the shape* behind *the left handle when using inverse distance or harmonic weights.*

where $\mathbf{1}_\mathbf{v}$ is the piecewise-constant characteristic function of the Voronoi cell around $\mathbf{v}$ when considered with all handles $\mathbf{c}_1, \ldots, \mathbf{c}_n$ and similarly $\mathbf{1}_j$ indicates the Voronoi cell around $\mathbf{c}_j$ when considered with all other handles $\mathbf{c}_{j \neq i}$.

Again, we normalize $\hat{w}_j$ to w_j as in Equation (9). However, this time many properties emerge, rather than slip away. Sibson shows that weighted sum of the handle locations is equal to the area-weighted evaluation point:

$$\left(\int \mathbf{1}_\mathbf{v} dV \right) \mathbf{v} = \sum_{j=1}^{m} \hat{w}_j(\mathbf{v}) \mathbf{c}_j, \tag{11}$$

$$\sum_{j=1}^{m} \hat{w}_j(\mathbf{v}) \mathbf{v} = \sum_{j=1}^{m} \hat{w}_j(\mathbf{v}) \mathbf{c}_j, \tag{12}$$

$$\mathbf{v} = \sum_{j=1}^{m} w_j(\mathbf{v}) \mathbf{c}_j, \tag{13}$$

or, more familiarly, that these weights are generalized barycentric *coordinates*. In other words, these weights can reproduce any affine function in space, a useful property for scatter data interpolation. For deformation, these weights are a form of "cage-free" barycentric coordinates.

Sibson shows that these coordinates are C^∞ except only C^0 on top of handles and only C^1 on the boundary of dual element circumspheres in the corresponding Delaunay triangulation. Various attempts to improve smoothness exist [Sibson 1981; Hiyoshi and Sugihara 2000], but two major hurdles remain for employing such weights in skinning applications.

First, it is already expensive to compute Voronoi diagrams in $\mathbb{R}^3$. Computing *geodesic* Voronoi diagrams inside arbitrary shapes accurately and efficiently is a daunting open problem in computational geometry [Kunze et al. 1997; Peyré and Cohen 2006]. There exist methods to approximate geodesic distances via a high-dimensional Euclidean embedding (e.g. [Lipman et al. 2010]). But Voronoi diagram complexity scales badly with respect to dimension d: conservatively for a p-manifold $O(n^{(d-1)/p})$ [Amenta et al. 2007]. The approximation power of such embeddings is also limited by the genus of the shape.

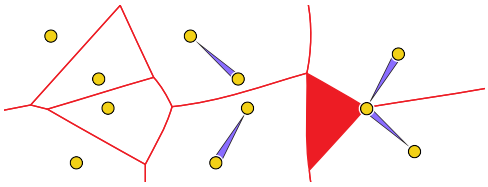


Figure 14: *For points, Voronoi interfaces are linear pieces. For disjoint bones, interfaces are well-defined, but quadratic. For connected bones, "interfaces" may have codimension 0 (red region).*

Second, it is not obvious how to incorporate other handle types, most notably, skeletons of bones. Voronoi diagrams are well-defined for arbitrary seed locations. Whereas for points, the interfaces between Voronoi cells are piecewise-planar, for *disjoint* line segments the interfaces are piecewise-parabolic. This complicates computation, but matters are even worse when considering that in a typical skeleton bones overlap at common joints. When this happens the interface changes co-dimension. The interface between the Voronoi cells of the two overlapping bones (i.e. the set where distances to either are equal) now spans a volume in $\mathbb{R}^3$ (or an area in $\mathbb{R}^2$, see Figure 14).

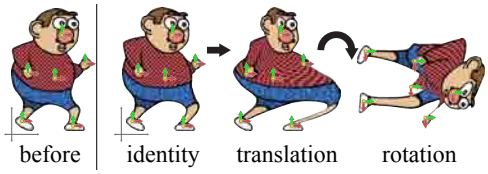


Figure 15: *Lack of partition of unity in weights forfeits affine invariance: even the identity transformation is not reproduced.*

Affine precision and partition of unity Why did we need to normalize inverse distance weights if doing so destroyed their intuitive decay? Partition of unity is a *mandatory* property of skinning weights. This ensures *affine invariance*, in the sense that if we apply the same affine transformation to each handle the result is applying that transformation to the entire shape directly:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v}) \mathbf{T} \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}, \tag{14}$$

$$\mathbf{v}' = \left(\sum_{j=1}^{m} w_j(\mathbf{v}) \right)^{1} \mathbf{T} \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}, \tag{15}$$

$$\mathbf{v}' = \mathbf{T} \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}. \tag{16}$$

Note this even applies if the identity transformation is prescribed $\mathbf{T} = (\mathbf{I}\ \mathbf{0})$. If the weights *do not* sum to one, then applying the identity transformation amounts to scaling each point $\mathbf{v}$ away from the origin by $\sum_{j=1}^{m} w_j(\mathbf{v})$ (see Figure 15). This reveals that partition of unity not only ensures reproduction of global transformations, but also makes skinning independent of the choice of origin (i.e. independent of world coordinates).

The term "affine invariance" is also used in the context of generalized barycentric coordinates, which—by definition—not only satisfy partition unity but also coordinate reproduction (e.g. see Equation (13)). Unlike general linear blend skinning weights, generalized barycentric coordinates can reproduce any affine transformation simply by translating handles. More on this in the next section.

In summary, inverse distance weights suffer from a number of shortcomings: dependence on Euclidean metric, lack of monotonicity, and elusive definitions for arbitrary handle types. We now turn our focus to the concept of *shape-awareness* and see how it can be achieved with simple closed-form weights *without* explicitly computing geodesic distances.

3.2 Generalized barycentric coordinates

Barycentric coordinates in a triangle, tetrahedron or other simplex provide a unique and natural way of interpolating data living at corners to the interior. These coordinates as weight functions reveal a host of convenient properties: e.g. smoothness, non-negativity, coordinate reproduction, monotonicity, ubiquitous domain and closed-form definition. The problem of generalized barycentric coordinates is to devise a similar definition for arbitrary polytopes while retaining as many of these properties as possible or as desired. This subject is quite old [Möbius 1827], but the last decade has prompted an emergence of the use of generalized barycentric coordinates for shape deformation (e.g. [Ju et al. 2005]).

Figure 16: *Repositioning (translating) cage vertices deforms the* Armadillo *embedded inside [Ju et al. 2005].*

In the context of shape deformation, the input model is enclosed in a polyhedral *cage*. The vertices of this cage become the deformation handles. We will define "coordinates" as a special case of "weights" which reproduce their evaluation points:

$$\mathbf{v} = \sum_{j=1}^{m} w_j(\mathbf{v})\mathbf{c}_j, \tag{17}$$

where $\mathbf{c}_j$ is the jth cage vertex in its rest position.

Generalized barycentric coordinates $\subset$ linear blend skinning Cage-based deformations using generalized barycentric coordinates can be understood as a special case of linear blend skinning where weights retain this special coordinate reproduction property of Equation (17) and handle transformations are restricted to translations. Recall, LBS deformation is written:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v})\mathbf{T}_j\mathbf{v} \tag{18}$$

$$= \sum_{j=1}^{m} w_j(\mathbf{v}) \left(\mathbf{L}_j\mathbf{v} + \mathbf{t}_j\right), \tag{19}$$

where the rest position of a point $\mathbf{v}$ is deformed to a new position $\mathbf{v}'$ according to a sum of affine transformations $\mathbf{T}_j$ (composed of linear and translation parts $\mathbf{L}_j$ and $\mathbf{t}_j$) applied to $\mathbf{v}$ and weighted by a spatially varying weight function $w_j(\mathbf{v})$. Barycentric coordinates may also be used to create deformations:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v})\mathbf{c}'_j, \tag{20}$$

where $\mathbf{c}'$ are the deformed positions of the control polygon (see Figure **??**). These deformations may be described as a restricted form of LBS. Substituting the identity in Equation (17) into Equation (20) and rewriting to match Equation (19) with $\mathbf{L}_j = \mathbf{I}$ and $\mathbf{t}_j = \mathbf{c}'_j - \mathbf{c}_j$, we have:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v})(\mathbf{c}'_j + \mathbf{c}_j - \mathbf{c}_j) \tag{21}$$

$$= \mathbf{v} + \sum_{j=1}^{m} w_j(\mathbf{v})(\mathbf{c}'_j - \mathbf{c}_j) \tag{22}$$

$$= \sum_{j=1}^{m} w_j(\mathbf{v})(\mathbf{I}\mathbf{v} + \mathbf{c}'_j - \mathbf{c}_j), \tag{23}$$

assuming that $\sum_{j=1}^{m} w_j(\mathbf{v}) = 1$, which is typically the case as otherwise affine invariance of the barycentric coordinates is lost. In this way we see barycentric coordinates is a special case of LBS. It is restricted in two ways: first, the transformations $\mathbf{T}_j$ are restricted to translations; and second, the weight functions $w_j(\mathbf{v})$ have the unique property of being *coordinates*. This second quality *helps* alleviate the first restriction in the sense that deformations retain affine invariance, however, the restriction on the weight function prohibits other uses. We discuss derivative control here, but locality and sparsity also become an issue.

"Higher order barycentric coordinates" We also quickly show that the higher order barycentric coordinate deformations of [Langer and Seidel 2008] are equivalent to linear blend skinning. Written as a special case of LBS, barycentric coordinate deformation clearly lacks control of derivatives. Simply consider identity translations. With barycentric coordinates, by setting translations to zero, all degrees of freedom are exhausted and the deformation is forced to be the identity transformation. With general LBS the linear part $\mathbf{L}_j$ of each transformation remains, allowing such control. Noticing and exploiting this was the contribution of [Langer and Seidel 2008]. Their derivation is conducted outside the context of linear blend skinning. We will now show that their "higher order barycentric coordinate" deformations are in fact equivalent to LBS with an extra restriction that the first derivatives of the weight functions vanish at each control point: $\nabla w_j = \mathbf{0}$.

In [Langer and Seidel 2008], an interpolatory function is introduced of the form:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v}) \left(\mathbf{c}_j' + \mathbf{D}_j(\mathbf{v} - \mathbf{c}_j) \right), \tag{24}$$

where $\mathbf{D}_j$ "are the linear functions (usually represented as matrices) which specify the derivatives at the" $\mathbf{c}_j$. We may rewrite our linear blend skinning formula in Equation (19) substituting $\mathbf{t}_j = -\mathbf{L}_j\mathbf{c}_j + \mathbf{c}_j$:

$$\mathbf{v}' = \sum_{j=1}^{m} w_j(\mathbf{v}) \left(\mathbf{L}_j\mathbf{v} - \mathbf{L}_j\mathbf{c}_j + \mathbf{c}_j \right), \tag{25}$$

$$= \sum_{j=1}^{m} w_j(\mathbf{v}) \left(\mathbf{c}_j' + \mathbf{L}_j(\mathbf{v} - \mathbf{c}_j) \right), \tag{26}$$

or equivalently reading from right to left, $\mathbf{v}$ is first translated to the rest position of the control point $\mathbf{c}$, where the linear (derivative) terms are applied (often just a rotation) and then it is translated *back* to the deformed position. Written in this way it is clear that higher order barycentric coordinate deformations are equivalent to the usual LBS deformations. What is interesting now, is how do the weights proposed by [Langer and Seidel 2008] fair as skinning weights. Unfortunately, they fair rather poorly. Their scheme is to apply a simple filter of the existing barycentric coordinate methods which were not originally designed to be good skinning weights. The filtered harmonic coordinates [Joshi et al. 2007] are promising, but we know these functions to be too globally supported (see Figure 13). The "axioms" of [Langer and Seidel 2008] turn out to be not quite all that we desire in skinning weights. Indeed we want interpolation, partition of unity and zero-gradient at handles, but we also want shape-awareness, locality, non-negativity, monotony, and smoothness. In addition, the notion of "coordinates" is lost after the filtering of [Langer and Seidel 2008]. In fact, zero-gradients combined with interpolation mutually excludes the reproduction property. There is no apparent gain in having the input to their filter be *coordinate* functions in the first place, beyond the fact that they satisfy the interpolation property.

GBC feature chart Rather than wax and wane about the merits of various generalized barycentric coordinates, we can *coarsely* summarize the state of the art in terms of which qualities are obtained by which coordinate types (see Table 1).

The columns in the table ask whether a given set of coordinates obtains the following *features*.

N-GON	Defined for cages with $n > d + 1$ vertices $\mathbb{R}^d$?
CONCAVE	Defined for concave cages?
SHAPE	Do the coordinates respect the *shape* of the input vertices as defined by the facets (edges in $\mathbb{R}^2$, polygons in $\mathbb{R}^3$) between them?
C^1	Are the coordinates at least C^1 inside the cage?
≥ 0	Are the coordinates always non-negative?
LAG.	Do the coordinates interpolate vertex values? That is, are $w_j(\mathbf{c}_j) = \delta_{ij}$?
COORD.	Do the coordinates truly behave as coordinates by reproducing affine functions, in particular the input positions: $\mathbf{v} = \sum_{j=1}^{m} w_j(\mathbf{v})\mathbf{c}_j$?
CLOSED	Can coordinate be computed using a closed-form expression?
MONO.	Extrema (local or global) appear only at cage vertices?
OUT	Well-defined outside of the cage?
LOCAL	Are coordinates non-zero only *near* their respective vertices and zero everywhere else?
POLY	In $\mathbb{R}^3$, are coordinates defined for polyhedral cages with (flat) facets of more than three sides (e.g. planar quad meshes)?

In Table 1, the rows are various definitions of generalized barycentric coordinates or similar weighting functions. Let us quickly review these coordinates in order to list proper references for a more detailed review. *Barycentric* simply refers to the usual barycentric coordinates defined in a $(d + 1)$-simplex in $\mathbb{R}^d$: a triangle in $\mathbb{R}^2$, a tetrahedron in $\mathbb{R}^3$, etc. These are, of course, not

Method	N-GON	CONCAVE	SHAPE	≥ 0	C^1	LAG.	CLOSED	MONO.	OUT	LOCAL	POLY	COORD.
Barycentric	X	X	•	•	•	•	•	•	•	X	X	•
Wachspress	•	X	•	X	•	•	•	X	?	X	X	•
Natural Neighbor	•	•	X	•	X	•	•	•	•	•	•	•
Mean Value	•	•	•	X	•	•	•	X	•	X	X	•
Green and others	•	•	•	X	•	X	•	X	X	X	X	•
Positive Mean Value	•	•	•	•	X	•	X	X	X	X	X	•
Harmonic	•	•	•	•	•	•	X	•	X	X	•	•
Maximum Entropy	•	•	•	•	•	•	X	?	X	X	•	•
Total Variation	•	•	•	?	X	•	X	?	X	•	•	•
Const. Biharmonic	•	•	•	•	•	•	X	•	X	•	•	X

Table 1: *Generalized barycentric coordinates feature chart. All methods fall short on one property or another.*

generalized but are included here to provide comparison. *Wachspress* coordinates [Wachspress 1975; Meyer et al. 2002] enjoy a greater history and literature especially in the finite-element method community, though find little use in deformation as they are not defined for concave cages. Similarly, *Natural Neighbor* or Sibson's coordinates [Sibson 1980; Sibson 1981; Hiyoshi and Sugihara 2000] enjoy success in general data interpolation notably for their Voronoi-based local supports. But their reliance on Euclidean distance prohibits directly use for shape deformation.

Mean Value coordinates [Floater 2003; Ju et al. 2005; Floater et al. 2005; Hormann and Floater 2006] pioneered the use of generalized barycentric coordinates for shape deformation. They set an example and inspired many variants (e.g. [Manson and Schaefer 2010; Li and Hu 2013]). Mean value coordinates have a simple closed-form expression but lack many desirable qualities. In particular, coordinates should be non-negative. Attempts at obtaining this came with sacrifices. *Positive Mean Value* coordinates are based on visibility and are consequently unsmooth [Lipman et al. 2007]. *Harmonic* coordinates resolve this in theory and promise other properties, but rely on numerical optimization in practice [Joshi et al. 2007]. Similarly, *Maximum Entropy* coordinates are non-negative by construction but rely on numerical optimization [Hormann and Sukumar 2008].

Green and others refers to the Green Coordinates [Lipman et al. 2008] and other "complex barycentric coordinates" [Weber et al. 2009; Weber and Gotsman 2010; Weber et al. 2012; Weber et al. 2011] that loosen the Lagrange or interpolation property and define weight functions as complex-valued (rather than scalar-valued) functions. This provides an interesting theoretical playground, especially in $\mathbb{R}^2$. However, the lack of interpolation makes the cage less meaningful as a user-interface and user direction can only be satisfied in a least squares sense. Extensions to $\mathbb{R}^3$ are elusive.

Total Variation refers to the recent energy-minimizing coordinates [**?**]. The choice of energy itself encourages sparsity, at the cost of ensuring smoothness. Their formulation includes a parameter to encourage smoothness, but at the cost of locality and monotonicity: the parameter manages a trade-off between smoothness and locality.

Const. Biharmonic refers to the general framework for computing skinning weights via constrained optimization [Jacobson et al. 2011; Jacobson et al. 2012b]. This optimization will be discussed in detail later in this chapter. As these functions do not reproduce the rest shape, I refuse to label them "coordinates". Their presence in the table merely provides comparison.

In summary, generalized barycentric coordinates represent an interesting field of research with applications beyond (and perhaps more appropriate than) shape deformation. The most reasonable methods require a cage as a user interface, which severely limits its application and use in practice. Cages are not only difficult to design (see Section 2.3), but also to pose. Indeed, many methods use cages and generalized barycentric coordinates as a subspace reduction relying on easier to use user-interfaces like points [Ben-Chen et al. 2009b; Weber et al. 2009; Weber and Gotsman 2010; Borosán et al. 2010; Thiery et al. 2012b] or skeletons [Ju et al. 2008] to position the cage and indirectly the embedded shape.

3.3 Connection to variational modeling

Let us now turn our attention from restrictive cage-based methods and consider a class of deformation techniques that are not traditionally classified as skinning: variational surface and solid modeling. Handle-based variational modeling techniques *minimize* an energy which measures the deformation quality (or rather lack of quality) subject to constraints ensuring handle interpolation. A generic optimization might look like:

$$\underset{\mathbf{V}'}{\text{argmin}}\ E(\mathbf{V}'), \tag{27}$$

$$\text{subject to } \mathbf{v}' = \overline{\mathbf{v}},\ \forall \mathbf{v} \in H, \tag{28}$$

where $H \subset \Omega$ is the subset of the shape covered by the handles and $\overline{\mathbf{v}}$ is the prescribed position of a point on the shape according to the handle covering it.

It is well outside the scope of this course to survey variational modeling techniques.[1] Besides the obvious generality of the choice of energy, the variational formulation neatly incorporates arbitrary handle types as fixed value equality constraints (a.k.a. boundary conditions). Traditionally, variational modeling methods focus on selected regions (e.g. [Botsch and Kobbelt 2004]), isolated points (e.g. [Sorkine and Alexa 2007]), and for articulated curves (e.g. [Zimmermann et al. 2007]). Each handle itself need not undergo a uniform affine transformation: the curves of [Zimmermann et al. 2007] are redraw by the user to take any shape. However, user interfaces for specifying a single 3D rigid transformation are straight-forward and most methods simply assume isolated points are repositioned (translated) and region handles are rigidly deformed.

Linear variational modeling In general, the optimization in Equation (27) is parameterized by the space spanned by all possible boundary conditions: that is, all possible choices for new positions of vertices lying in the handle set. For a dense mesh, the number of vertices in the handles or even along the boundary of handles could be quite large. However, as Botsch & Kobbelt noticed [2004], if each handle j undergoes a uniform affine transformation T_j, then the effective parameter space is much smaller.

This is particularly poignant in the case where the deformation $E(\mathbf{V}')$ is quadratic [Botsch and Sorkine 2008]. The solution operator is linear, revealing that the deformed positions $\mathbf{V}'$ are a linear function of the small number of handle transformations:

$$\mathbf{V}' = \mathbf{M}\mathbf{T}, \tag{29}$$

where $\mathbf{M} \in \mathbb{R}^{n \times m(3+1)}$ contains the solution operator projected onto the space spanned by the constant affine handle transformations stack (transposed) in $\mathbf{T} \in \mathbb{R}^{m(3+1)\times 3}$.

As we can write linear blend skinning as a matrix multiplication with the same dimensions (see e.g. [Kavan et al. 2010; Jacobson et al. 2012a]), it is tempting to declare that handle-based linear variational modeling is an instance of linear blend skinning. However, standard linear blend skinning weights are scalar and apply to each coordinate the same way, implying redundancies in the columns of $\mathbf{M}$. The columns of $\mathbf{M}$ resulting from linear variational modeling will in general shown no redundancies. This indicates that linear variational modeling is an instance of *animation space* deformation [Merry et al. 2006], a generalization of linear blend skinning and in turn a special case of the more general *multi-weight enveloping* [Wang and Phillips 2002].

Quadratic smoothness energies Though linear variational modeling resulting from quadratic energy minimization is *not quite* the same as linear blend skinning, we may still gain insights by examining these techniques. Variational shape modeling seeks to measure directly the resulting deformation, searching for the *best* choice of parameters (i.e. handle transformations). The energy $E(\mathbf{V}')$ employed defines what is meant by best. In the absence of any prior information of a shape's physical materials, a common assumption is that the best deformations are *fair*. Fairness is an abstract aesthetic quality, not intended to be defined concretely [Botsch et al. 2010]. Intuitively, fair deformations are smooth and avoid *unnecessary* oscillations (cf. Runge's phenomenon).

Common quadratic fairness energies are associated with poly-harmonic partial differential equations:

Common name	Least squares description	Energy	PDE	Boundary continuity
Dirichlet	f should be as constant as possible	$\int_\Omega \|\nabla f\|^2 dV$	$\Delta f = 0$	C^0
Laplacian	f should be as harmonic as possible	$\int_\Omega (\Delta f)^2 dV$	$\Delta^2 f = 0$	C^1
Laplacian gradient	Δf should be as constant as possible	$\int_\Omega \|\nabla \Delta f\|^2 dV$	$\Delta^3 f = 0$	C^2
...				

This table represents a rather cursory overview of these and related energies. Specific boundary conditions play a very important role: for example, the Laplacian energy $\int_\Omega (\Delta f)^2 dV$ and the thin-plate spline energy $\int_\Omega \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial xy}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 dV$ are both minimized by biharmonic functions $\Delta^2 f = 0$. The difference between the two energies may be reduced to a term measured along the boundary of the domain.

We also list the "least squares description" as it helps build an intuition for the minimizers of these energies and also reveals what types of functions f will *perfectly* measure zero energy.

Finally, we list the level of continuity minimizers may be expected to achieve at constraint locations (in our case at the handles, see Figure 17). As we will only consider the discrete case where these energies are discretized using finite-element method over triangle

[1] Though a modern, in-depth survey of non-linear techniques is long overdue.

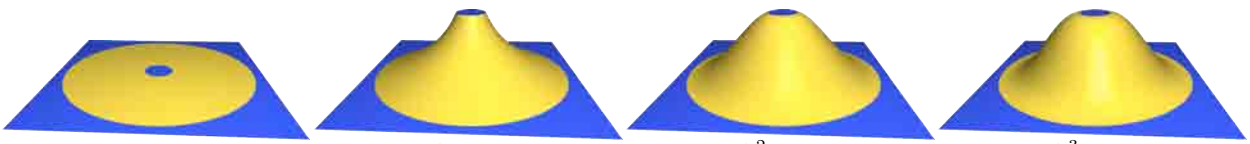


Figure 17: *k-harmonic solutions show C^{k-1} continuity at constraints.*

and tetrahedron meshes, these continuity levels are only seen in the limit [Botsch and Kobbelt 2004; Jacobson et al. 2010]. Further, these levels are bounds on the continuity near constraints. Away from constraints, the limit solutions for all are C^∞.

All energies are *intrinsic* in the sense that they are determined solely by the metric inside or on the shape, and not the shape's particular embedding in Euclidean space. This ensures that resulting deformations are shape-aware.

Traditionally these energies are applied directly to the unknown surface position coordinate functions $\mathbf{V}'$ [Botsch et al. 2006] or to unknown displacements $(\mathbf{V}' - \mathbf{V})$ [Botsch and Kobbelt 2004; Sorkine et al. 2004]. Working directly with positions is problematic, since the original surface is likely not a minimizer so identity (unmoved) handle constraints will result in *popping* to a smoothed out surface. Working with displacements ensures that original details are not smoothed away, and shifts the conversation away from measuring the fairness of surface to measuring the fairness of the *displacement field*.

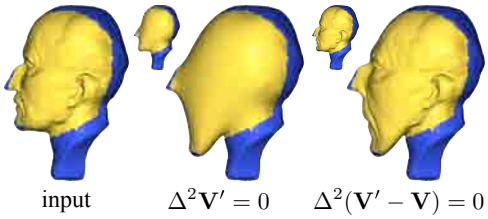


Figure 18: *Variational modeling techniques either directly optimize surface positions (details smoothed away) or optimize* displacements*, where identity handle constraints (inset) show no popping.*

If we directly apply these energies to our skinning weight functions (e.g. $\int_\Omega \|\nabla w_j\|^2 dV$), we again shift the conversation, now to the fairness of the weight functions. We must convince ourselves that fair skinning weight functions will produce fair deformations. In general, the displacements induced by linear blend skinning with biharmonic weights ($\Delta w_j = 0$) will not be biharmonic. In fact, these two deformation paradigms will only coincide for purely translated handles (see Section 4.7 in [Jacobson 2013]). So, we should divorce ourselves from the notion that the only fair deformations are those produced by displacements minimizing certain energies.

These same energies *are* of interest for computing skinning weights because they simultaneously achieve: 1) smoothness, 2) shape-awareness, and 3) support for various handle types. We will see how additional optimization constraints facilitates other desirable features.

3.4 Desirable qualities for skinning weights

It is high time to list desirable qualities of skinning weights. These qualities not only guide our choice of energies and constraints, but also provide a lens through which we may examine existing methods (see Table 2).

Method	SPEED	POINT	BONE	CAGE	CURVE, REGION	SHAPE	≥ 1	MONO.	LOCAL	LAG.	C^1
Inv. Euclidean	closed form	•	•	X	•	X	•	X	X	•	•
Inv. geodesic	Dijkstra	•	•	X	•	•	•	X	X	•	X
Natural neighbor	Voronoi	•	X	X	X	X	•	•	•	•	•
Harmonic	linear solve	•	•	•	•	•	•	•	X	•	X
Bone heat	visibility	•	•	X	X	•	•	X	X	•	X
Biharmonic	linear solve	•	•	•	•	•	X	X	X	•	•
Compressed modes	convex, ADMM	X	X	X	X	•	X	X	•	X	?
Const. biharmonic	conic program	•	•	•	•	•	•	•	•	•	•
Elasticity inspired	non-convex	X	•	X	X	•	X	X	X	•	X

Table 2: *Skinning weights feature chart: features are realized as optimization constraints resulting in more involved solving routines.*

Interpolation (LAG.) and partition of unity are mandatory properties to ensure handle constraints can truly act as a user interface. If the deformation should be smooth then weights must be smooth, typically at least derivative continuity (C^1), but in some cases higher-order curvature continuity is required for controlling highlights [Tosun et al. 2007]. Recall, that *smoothness* is just one component of the more general requirement of *fairness*. While skinning is traditionally controlled by BONES, an ideal automatic weights method should form a unified framework to support all common handle types: POINTS, CAGES, CURVES, and REGIONS.

We can vaguely define shape-aware weights (SHAPE) as those that decay with respect to geodesic distance on/within the shape as opposed to Euclidean distance and the ambient space around the shape. Shape-awareness ensures that handles do not affect geodesically (and therefore semantically) distant parts the shape (see Figure 19).

Even perfectly smooth weights may oscillate wildly, causing unintuitive responses to a user's handle manipulations. Weights that *dip* below zero, cause parts of the shape to respond in the exact opposite of the user's edit: the user translates to the right and rotates counterclockwise, but the part of the shape with negative weights will translate left and rotate clockwise (see Figure 20). Thus, at a bare minimum we should require weights to be non-negative (≥ 0). Combined with partition of unity this implies that weights are also bounded above by one. Otherwise regions may be too strongly affected by a user's edit: moving left 10 pixels causes regions with weights of two to move 20 pixels.

An even stricter condition is to banish oscillations all together, requiring weights to be monotonically decreasing away from one at their respective handle and toward zero at other handles (MONO.). Weights for the tail handle of the *Lobster* in Figure 20 oscillate even within $[0, 1]$. The local maximum in the right claw is the harbinger of the unintuitive response the claw receives when moving the tail.

Many shader implementations of skinning mandate an upper bound on the number of non-zero weights influencing a vertex of the shape. This implies that weights should be sparse: weights should be zero over most of the shape. Support regions (with non-zero weights) should be localized around a handle (LOCAL).

Finally, automatic methods are meant to replace tedious manual weight painting. In some sense, computation time is easily discarded: it is precomputation done once and for all. In reality, even with automatic methods in the loop rigging is still an interactive process. A user may be unsatisfied with the current skeleton configuration and need to recompute new weights. The ability to compute quality weights quickly (SPEED) would also open doors to new applications (e.g. physically based simulation with contacts and fracture, cf. [Faure et al. 2011; Jacobson 2013; Harmon and Zorin 2013]).

In Table 2, I list speeds in terms of bottleneck subroutines. For example, a common subroutine is to solve a linear system with a Poisson-like sparsity pattern.

3.5 Unconstrained optimization

Inverse distance methods boast closed-form expressions that do not rely on spatial discretization or even a specific shape description. However, these methods are very limited. In order to retain smoothness while achieving shape-awareness, Baran & Popović propose a two step approach to compute automatic weights for skeletal bones [2007a]. Their method is often called "bone heat" as it loosely relates smooth weight computation to heat diffusion. I will first give an alternative understanding of this method as an instance of variational data smoothing. Then we will proceed to other similar unconstrained optimizations.[2]

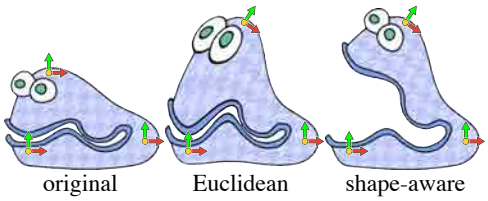


Figure 19: *Euclidean distance-based weights* bleed *across the* Clam*'s open mouth, unaware of the shape's boundary.*

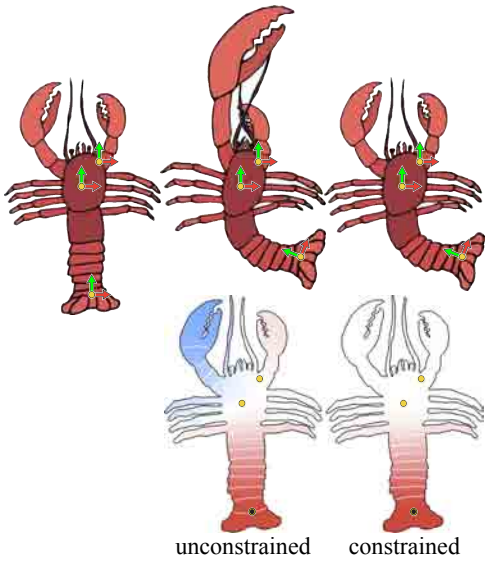


Figure 20: *Unconstrained optimization allows non-negative (blue), non-monotonic (local minimum and maximum in claws) weights, leading to unintuitive response.*

Bone heat First each point on the shape is rigidly attached to its closest bone, using the usual Euclidean distance:

$$\hat{w}_j(\mathbf{v}) = \begin{cases} 1 & d(\mathbf{v}, h_j) < d(\mathbf{v}, h_j), \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

We can think of $\hat{w}_j$ as a "data prior," a term familiar to image segmentation. This data is *noisy* in the sense that it does not account for the shape's boundary and it is piecewise constant instead of smooth. To fix the shape awareness issue, consider an energy measuring adherence to this data according to some *confidence* function $k(\mathbf{v})$. If the closest bone to a point $\mathbf{v}$ is not visible, then set $k(\mathbf{v})$ to zero (completely not confident). Otherwise, set $k(\mathbf{v})$ to $d(\mathbf{v}, h_j)^{-2}$, where h_j is the closest bone to $\mathbf{v}$. This way confidence quickly decreases as we move farther away from the bone. To fix the lack of smoothness, consider a second energy which measures smoothness over the shape's surface, namely the Dirichlet energy. Combining these two energies results in quadratic energy which balances between reproducing the rigid closest visible bone (data) and minimizing gradients over the surface (smoothness):

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \|\nabla w_j(\mathbf{v})\|^2 + k(\mathbf{v}) \left(w_j(\mathbf{v}) - \hat{w}_j(\mathbf{v})\right) \, dV. \tag{31}$$

There are no constraints, not even boundary conditions, as it is assumed that bones are buried inside the shape and we are only solving over the surface.

This problem is discretized using finite-element method on a triangle mesh. Differentiating with respect to each $\mathbf{w}_j$ and setting the result to zero reveals the following linear system of equations:

$$(\mathbf{L} + \mathbf{K})\mathbf{W} = \mathbf{K}\hat{\mathbf{W}}, \tag{32}$$

[2]Technically, fixing values at/along/on handles are constraints as boundary conditions. I write "unconstrained" to indicate the absence of other constraints (e.g. inequalities).

where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the sparse discrete Laplacian matrix [Meyer et al. 2002], $\mathbf{K}$ is a diagonal matrix with $K_{ii} = k(\mathbf{v}_i)$, and $\mathbf{W}, \hat{\mathbf{W}} \in \mathbb{R}^{n \times m}$ contain the weight functions for each handle as columns.

By construction, these weights partition unity (constant functions are in the null-space of $\mathbf{L}$). The weights are smooth (see Figure 21).[3] Because the original data $\hat{\mathbf{W}}$ is bound between $[0, 1]$, the smoothed solution $\mathbf{W}$ will be, too.

Monotonicity is often observed in practice, but not guaranteed. Intricate concavities and tunnels can lead to arbitrarily disconnected components in the raw weights $\hat{\mathbf{W}}$.

Perhaps the weakest quality is locality. The surface diffusion runs rampant in areas where no strong data term is active. We will see that this is a natural consequence of employing the Dirichlet energy which prefers constant functions, thus weights tend toward a globally average value rather than zero.

Part of the success of [Baran and Popović 2007a] can be attributed to the avoidance of heavy spatial discretization. It is unlikely that character surface triangle meshes *in the wild* are "quality" from a FEM standpoint, but far more unlikely that a suitable volumetric tetrahedra mesh can be computed (though gains are being made in this direction, see [Si 2003; Jacobson et al. 2013a; Xu and Barbič 2014]). Thus, much less preprocessing is require to use the bone heat method on a given model. Hence, its adaptation in software packages such as MAYA and BLENDER.

However, reluctance to discretize the solid shape *as a solid* comes at a cost. Visibility modulated distance measure is crude replacement for geodesic distance and diffusion over a surface is very different from diffusion through a volume (see Figure 22). The following progression will assume an available and appropriate shape discretization. Then we can isolate and resolve each problem in a principled manner.

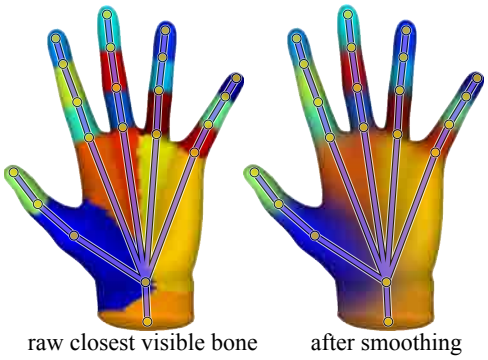


Figure 21: *Visibility as a shape-aware proxy for internal, geodesic proximity is treated as a raw signal (left) and smoothed over the surface (right).*

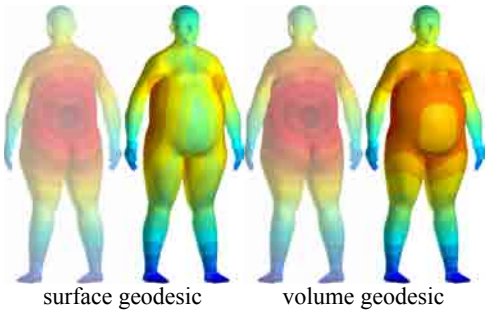


Figure 22: *Surface geodesics must travel around the* Fat Man*'s belly; volume geodesics may travel through it.*

Unconstrained poly-harmonic weights Why does the bone heat method of [Baran and Popović 2007a] need the closest visible bone "data term"? Shouldn't the smoothness term with appropriate boundary conditions be enough to produce smooth shape aware weights? The knee-jerk response is to blame the lack of a volumetric discretization. The data term is a way to *attach* the interior bones to the surface where smoothing can be conduct. This response is valid, but masks another issue: the choice of smoothness energy.

Let us now consider directly optimizing the Dirichlet energy subject to boundary conditions which support points, bones, curves, regions and cages:

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \|\nabla w_j(\mathbf{v})\|^2 \, dV, \tag{33}$$

$$\text{subject to } w_j(\mathbf{v}) = \varphi_j(\mathbf{v}) \; \forall \mathbf{v} \in H, \tag{34}$$

where φ_j is a generalized hat function defined over our handle set H. For $\mathbf{v}$ lying on the ith points, cage vertices, bone segments, regions or curves, then $\varphi_j(\mathbf{v}) = 1$ if $i = j$ and $\varphi_j(\mathbf{v}) = 0$ if $i \neq j$. If $\mathbf{v}$ lies on a cage simplex $i, k, \ldots$ then $\varphi_j(\mathbf{v})$ is the barycentric coordinate function if $j \in i, k, \ldots$ and $\varphi_j(\mathbf{v}) = 0$ otherwise.[4] We will reuse these fixed value constraints through the following sections.

Now that we are possibly dealing with cages, we interpret the domain of our integrated energy Ω to be the union of the possibly solid input shape and any closed cages around or partially around it. Smoothness energies built from differential quantities integrated over Ω will be shape-aware by construction.

If our handles were only a single enclosing cage, then this formulation is equivalent to harmonic coordinates[Joshi et al. 2007]. In this case the smoothness issues of Dirichlet energy solutions near fixed values may be ignored because the cage is outside, away from the shape. Similarly, the cage combats the global support of harmonic functions. We know that Dirichlet energy will prefer

[3] So long as handles are not incident on the surface, in which case corresponding entries in $\mathbf{K}$ tend toward infinity implying hard constraints, probably converging to something only C^1.

[4] Fixing the weights to be equal to barycentric coordinates along cage simplices is in some sense arbitrary. It ensures linear behavior on that simplex (e.g. for stretching), but by construction introduces derivative discontinuities at cage vertices with non-trivial cage valence. It is perhaps only important that the cage handles incident share control over the simplex, while preventing other handles. Thus, we could only enforce $w_j(\mathbf{v}) = 0$ for $\mathbf{v}$ on a cage simplex $i, k, \ldots \ni j$ [Manson and Schaefer 2010].

constant functions. This is at odds with our goals of locality: in general solutions will flatten out near an average before reaching zero. In the case of cages, the entire boundary of domain except cage facets incident on cage vertex j will be enforcing $w_j = 0$. This does not technically induce locality: harmonic functions are *strictly monotonic* reaching minimal values (zero) only at the domain boundary. However, in practice the weights w_j become negligible far from handle j.

The story for points, bones, curves and regions lying directly on or in the shape is very different. Lack of derivative continuity near handles is immediately apparent (see Figure 23 and recall Figure 17). Without an enclosing cage, we default to implicit Neumann boundary conditions $\frac{\partial w}{\partial n} = 0$ on the boundary of Ω. These do not encourage sparsity and indeed we see global support (see Figure 23). The global support of harmonic weights prevents us from simply smoothing the weights *post facto* [Joshi et al. 2007].

Instead, we may try to enforce smoothness at handles directly by switching to a higher-order smoothness energy. Namely, the Laplacian energy:

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \left(\Delta w_j(\mathbf{v})\right)^2 dV, \tag{35}$$

$$\text{subject to } w_j(\mathbf{v}) = \varphi_j(\mathbf{v}) \; \forall \mathbf{v} \in H. \tag{36}$$

Indeed, we now see C^1 continuity at handles, but we have lost the maximum principle. Biharmonic weights oscillate wildly over the domain even outside the range $[0, 1]$. Negative, non-monotonic weights induce unintuitive responses to user edits (see Figure 20).

We seem to have satisfied one property only to lose many others. We could continue to increase the order of our energy (triharmonic, tetraharmonic, etc.) and we would see similar effects: continuity at handles is increased but oscillations become wilder and wilder [Jacobson et al. 2012b] (see Figure 24).

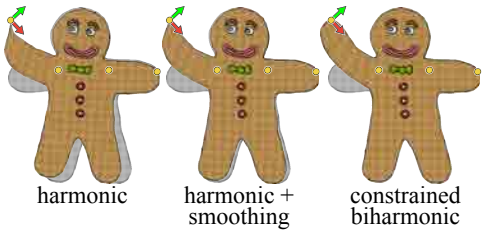


Figure 23: *Harmonic weights are globally supported and not smooth at handles.* Post facto *smoothing helps but doesn't fix their global influence (original shape is shown behind in grayscale).*

Harmonic (as an extension of [Joshi et al. 2007]) and biharmonic (as an extension of [Botsch and Kobbelt 2004]) skinning weights are in some sense *strawmen* in an informal fallacy. No serious previous work has directly claimed these would be good weights to use for general skinning. However, many methods are based on similar unconstrained energy optimizations and exhibit similar artifacts. It is interesting to consider them in the context of skinning weights, because we may examine weight functions as scalar fields defined by simple boundary conditions. Here their topological properties lay open for inspection. It is via this analysis that Jacobson et al. could formulate the following constraints to sculpt meaningful and useful automatic skinning weights [Jacobson et al. 2011; Jacobson et al. 2012b].

3.6 Constrained biharmonic weights

We have seen that minimizing the Laplacian energy (resulting in biharmonic weights) produces smooth, shape-awawre weights even near handles placed in or on the surface of the shape. The lingering desirable qualities may now be tackled with additional constraints.

We must tame the oscillations of the biharmonic weight functions. One attempt is to approach this symptomatically. The presence of negative weights seems tied to the presence of oscillations. We could therefore try to reduce oscillations by prohibiting negative weights. By adding, bound (box) constraints to our optimization we change our linear optimization problem to a nonlinear (but still convex) one. We therefore need to also enforce partition of unity explicitly:

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \left(\Delta w_j(\mathbf{v})\right)^2 dV, \tag{37}$$

$$\text{subject to } w_j(\mathbf{v}) = \varphi_j(\mathbf{v}) \; \forall \mathbf{v} \in H, \tag{38}$$

$$w_j(\mathbf{v}) > 0, \tag{39}$$

$$\sum_{j=1}^{m} w_j(\mathbf{v}) = 1. \tag{40}$$

This problem is called a quadratic program, a special case of convex optimization with efficient solvers. In practice, interior point solvers such as MOSEK [Andersen and Andersen 2000] and MATLAB's `quadprog` are the fastest ways to achieve useful weights (pointwise accuracy $< 1e-4$). When evaluating supports and topological properties as we will do next, it is important to use much more conservative tolerances for interior point solvers (which severely diminishes performance) or to switch to an active set solver (e.g. LIBIGL's `igl::active_set` [Jacobson et al. 2013b]).

Imposing bounds has an interesting immediate side effect. Not only are oscillations dampened, but weights are also suddenly locally supported.

This can be understood *intuitively*. The weight functions must smoothly decay from one at their respective handle to zero at other handles. Upon reaching zero, the weights are prevented from continuing to swing below zero by the imposed bounds. The only choice for increasing the non-zero support would be to reverse and swing upward again in the range $[0, 1]$ beyond the other handles. The Laplacian energy prefers harmonic functions which avoid unnecessary oscillations. The weights had negative derivative in the direction toward the other handles. To oscillate upward when continuing in that direction, the gradient must change to be positive. Laplacian energy punishes exactly this change, hence it is avoided.

This line of reasoning does seem to explain the observed *a posteriori*, but has a tendency to anthropomorphize the energy: "the energy wants weights to be local!"

A more principled explanation is drawn from [Rustamov 2011], who employs very similar constraints on the Laplacian energy to design multi-scale kernels. The least absolute shrinkage and selection operator (LASSO) encourages sparsity [Tibshirani 1996]. The ideal sparsity constraint may be written in terms of the ℓ_0-norm:

$$\sum_{j=1}^{m} |w_j(\mathbf{v})|_0 < t \; \forall \mathbf{v} \in \Omega, \tag{41}$$

where $|x|_0 = 1$ if $x \neq 0$ and $|x|_0 = 0$ only if $x = 0$. Such constraints are called cardinality constraints and result in non-convex optimization: they are not efficiently enforced.

Tibishirani shows, however, that replacing this ℓ_0-norm constraint with a looser ℓ_1-norm constraint *often* has the same effect:

$$\sum_{j=1}^{m} |w_j(\mathbf{v})|_1 < t \; \forall \mathbf{v} \in \Omega, \tag{42}$$

where $|x|_1$ is the usual absolute value of x. The sparsity effect is stronger with larger m, i.e. with more handles (as is our want).

Rustamov is not optimizing skinning weights. Instead, he wants smooth, locally supported functions centralized around a selected point. However, his optimization problem is ostensibly similar to ours:

$$\underset{f}{\text{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \left(\Delta f(\mathbf{v})\right)^2 dV, \tag{43}$$

$$\text{subject to } f(\mathbf{v}_{\text{selected}}) = 1, \tag{44}$$

$$\frac{1}{V} \int_\Omega |f(\mathbf{v})|_1 dV \leq t, \tag{45}$$

where V is the total volume of Ω.

Rustamov observes that if f is positive, the integral of the function's absolute value is simply the integral of the function's value:

$$f(\mathbf{v}) \geq 0 \longrightarrow \frac{1}{V} \int_\Omega |f(\mathbf{v})|_1 dV = \frac{1}{V} \int_\Omega f(\mathbf{v}) dV. \tag{46}$$

This allows him to substitute the difficult-to-manage ℓ_1 constraint with bound constraints and a linear inequality constraint:

$$\underset{f}{\text{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \left(\Delta f(\mathbf{v})\right)^2 dV, \tag{47}$$

$$\text{subject to } f(\mathbf{v}_{\text{selected}}) = 1, \tag{48}$$

$$f(\mathbf{v}) \geq 0, \tag{49}$$

$$\frac{1}{V} \int_\Omega f(\mathbf{v}) dV \leq t. \tag{50}$$

We now start to see the similarity to our problem.

Our partition of unity constraint effectively enforces:

$$\frac{1}{mV} \int_\Omega \sum_{j=1}^{m} w_j(\mathbf{v}) dV = 1. \tag{51}$$

combined with the non-negativity bounds, this in turn implies that

$$\frac{1}{mV} \int_\Omega \sum_{j=1}^{m} |w_j(\mathbf{v})|_1 dV = 1. \tag{52}$$

This is perhaps slightly less sparsity-inducing than allowing the integrated ℓ_1-norm to be *less than* one. But it is perhaps the best we can hope for from this framework while maintaining partition of unity.

Our work is not yet done. Negativity is merely a symptom of oscillations. Preventing negative weights does not prevent oscillations within the range $[0, 1]$. This problem appears most frequently when sparse handle sets are placed on shapes with long, thin extruding parts (see Figure 24). The problem is exacerbated by even higher order smoothness energies.

To explicitly prevent oscillations and force weights to be monotonic (free of local extrema), look back to harmonic weights. The maximum principle of harmonic functions guarantee that harmonic weights are monotonic (whether continuous and discrete [Wardetzky et al. 2007]). Building upon [Weinkauf et al. 2010], Jacobson et al. [2012b] *copy* the monotonicity of harmonic weights and *paste* it as constraints onto the optimization above. The weight at each vertex of the discretized domain is sure *not* to be a local extrema if one of its neighbors' weights is greater and one is less. Determining which neighbor to consider is a combinatorial problem not easily answered. Instead, compute a set of harmonic weights as *representative functions* u_j. These functions represent a monotonic topology, but have suboptimal values (as measured by the *Laplacian energy*). Therefore, optimize for new values w_j which have the same topological structure as u_j. For each vertex i, find the neighbor k with the greatest u_j value and constraint $u_j(\mathbf{v}_i) \leq u_j(\mathbf{v}_k)$. Add a similar constraint for each neighbor with least u_j value. Define a "constraint edge" $\{i, k\}$ in $\mathcal{E}_j$ to indicate the constraint $w_j(\mathbf{v}_i) \leq w_j(\mathbf{v}_k)$. Then we may write our optimization as:

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega \left(\Delta w_j(\mathbf{v})\right)^2 dV, \tag{53}$$

$$\text{subject to } w_j(\mathbf{v}) = \varphi_j(\mathbf{v}) \; \forall \mathbf{v} \in H, \tag{54}$$

$$w_j(\mathbf{v}_i) < w_j(\mathbf{v}_k) \; \forall \{i, k\} \in \mathcal{E}_j, \tag{55}$$

$$\sum_{j=1}^{m} w_j(\mathbf{v}) = 1. \tag{56}$$

This problem is also a quadratic program, albeit with many more constraints than before. This problem can still be efficiently optimized by first converting it to a conic program [Jacobson et al. 2012b].

Recently, it was realized that the resulting w_j are in turn new *representative functions* [Günther et al. 2014]. The optimization can be improved by iteratively solving the problem above, reseting the edge sets $\mathcal{E}_j$ according to each vertex's new greatest and least neighbor. Each edge-set represents a conservative convexification of the feasible space (monotonic functions). Each iteration resets this convexification around the current guess, starting with a guaranteed-monotonic harmonic weights (see Figure 25).

Discretization and implementation It is beyond the scope of this survey to derive the finite-element discretization of our optimization problem above (see e.g. [Jacobson et al. 2010; Jacobson 2013]). Instead, let us assume that our domain Ω (union of shape and any cages) is discretized with an appropriate triangle or tetrahedral mesh with n vertices. Then the two ingredients necessary to set up our problem are the sparse discrete Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ and the discrete diagonal mass (a.k.a. volume) matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$. In a purely discrete setting, our energy is approximated by integrating the pointwise (*unintegrated*) Laplacian

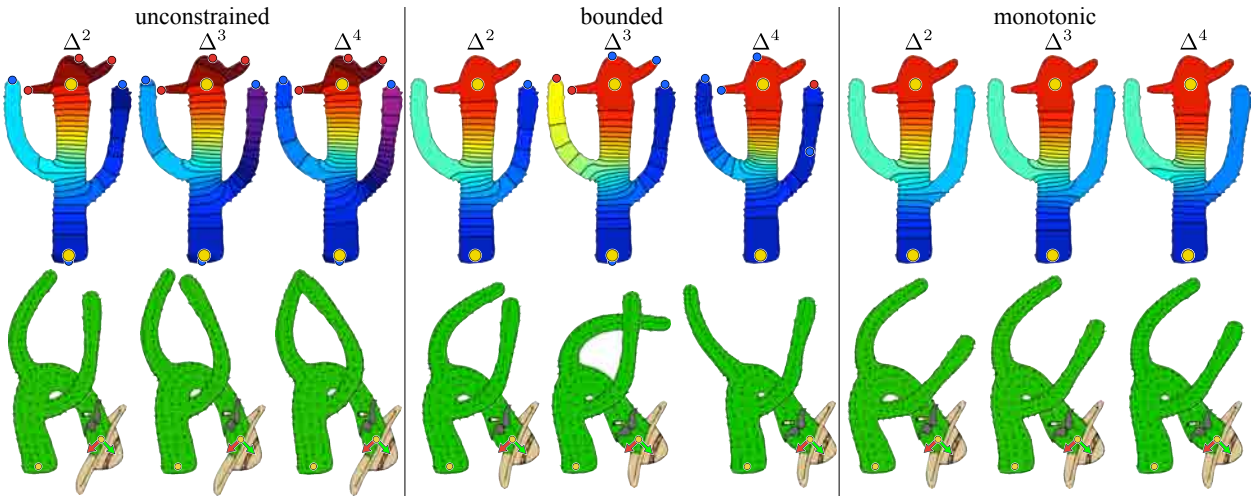
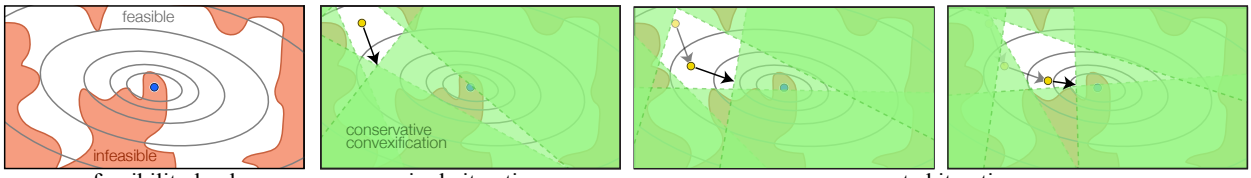


Figure 24: *Unconstrained weights oscillated wildly (red dots: local maxima, blue dots: local minima). Bounds help tame oscillations but do not defeat them. Explicitly montonicity constraints eventually remove all oscillations.*



energy, feasibility landscape | single iteration [Jacobson et al. 2012] | repeated iterations [Günther et al. 2012]

Figure 25: *Illustration: The space of monotonic functions is non-convex and difficult to parameterize (white space). Convexifying around harmonic weights (first yellow dot) allows efficient, conservative optimization. Repeating this improves until convergence.*

$(\mathbf{M}^{-1}\mathbf{L})$ of the unknown weights:

$$\sum_{j=1}^{m} \int_{\Omega} (\Delta w_j)^2 \, dV \approx \text{tr}\left((\mathbf{M}^{-1}\mathbf{L}\mathbf{W})^{\mathsf{T}}\mathbf{M}(\mathbf{M}^{-1}\mathbf{L}\mathbf{W})\right) \tag{57}$$

$$= \text{tr}\left(\mathbf{W}^{\mathsf{T}}\mathbf{L}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{L}\mathbf{W}\right) \tag{58}$$

$$= \text{vec}(\mathbf{W})^{\mathsf{T}}\left(\mathbf{I}_m \otimes (\mathbf{L}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{L})\right)\text{vec}(\mathbf{W}) \tag{59}$$

where $\text{tr}(\mathbf{X})$ computes the algebraic trace of $\mathbf{X}$, $\text{vec}(\mathbf{X})$ *vectorizes* a rectangular matrix into a single column, $\mathbf{I}_m \in \mathbb{R}^{m\times m}$ is the identity matrix, and $\mathbf{A} \otimes \mathbf{B}$ computes the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$.

Though the energy is separable in the columns of $\mathbf{W}$, the n partition of unity constraints tie weights across the m handles together:

$$(\mathbf{1}_m^{\mathsf{T}} \otimes \mathbf{I}_n)\text{vec}(\mathbf{W}) = \mathbf{1}_n, \tag{60}$$

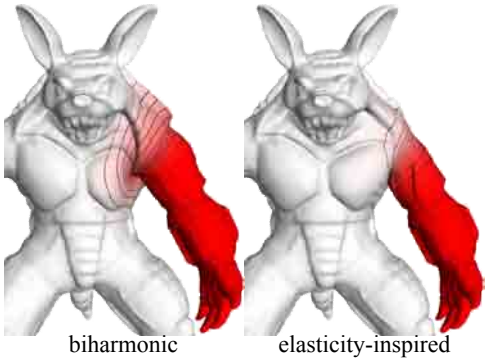
where $\mathbf{1}_k$ is a column of k ones.

These constraints prove to be very frustrating. The problem size, computation time and memory requirements are now tied to the product mn as weights for all handles must be computed simultaneously. Jacobson et al. proposed *dropping* the partition of unity constraints during optimization and normalizing *post facto* [2011]. This truly separates the optimization of each weight and greatly reduces computational complexity. There is strong evidence that doing so does not affect the high-level qualities of the weights, but essentially all claims are theoretically dismissed. Smoothness, interpolation, and non-negativity properties are not tarnished. However, our *principled* explanation for the locality of these weights depends on the partition of unity constraint, and monotonic weights could become non-monotonic after normalization.

It is an open problem to understand *why* constrained biharmonic weights without the partition of unity constraint during optimization and *post facto* normalization are similar to their properly optimized counterparts. A more interesting question might be *when* or *where* may they be expected to be similar? This might lead to more efficient optimization schemes for the full optimization problem.

Choice of energy So far, we have motivated our choice of energy based on desires for smoothness and shape-awareness. In the absence of any other information about the input shape or the way it should deform, these are reasonable *priors*. However, the variational modeling and physically-based simulation literatures have largely converged on the assumption that solid shapes behave elastically, as if made out of homogeneous rubber (see e.g. [Müller et al. 2005; Botsch et al. 2007]).

Kavan & Sorkine [2012] optimize for weights that minimize the "as-rigid-as-possible" energy familiar to variational modeling and cousin to corotational elasticity in physically-based simulation [Chao et al. 2010]. This energy measures the local rigidity of the deformed shape according to the yet-unknown weights for a set of input examples poses. The optimization is non-convex, and the smoothness of the weights is unknown, though probably not C^1 at handles. It is easy to imagine generating reasonable example skeleton poses automatically, but difficult to generalize this to other handle types. As intended, these weights do produce more elastic looking deformations, especially when combined with the modified skinning formula also proposed in [Kavan and Sorkine 2012] (see Figure 26).



biharmonic | elasticity-inspired

Figure 26: *Biharmonic weights produce smooth deformations by minimizing the Laplacian energy, but elasticity-inspired energies may be more appropriate for solid shapes.*

3.7 Localization and sparsity

We have read how the non-negativity and partition of unity constraints could explain the sparsity observed in constrained biharmonic weights.

However, there is no strong claim of sparsity in the sense of *minimizing* an ℓ_1-norm or achieving a bound on the ℓ_0-norm.

Recently, Neumann et al. apply ℓ_1-regularization to the discrete Laplacian eigen-decomposition problem. Doing so produces a sparse orthogonal basis. One potential application of this basis is deformation. The basis is automatically distributed over the shape: each basis vector centralized at a reasonable feature location. In some sense, this means that they simultaneously solve the automatic handle and automatic weights problem. However, interpolation is not guaranteed, so *handles* in this setting are not usable for direct manipulation. Nonetheless, this represents a very promising direction for future research.

Automatically determined locality or sparsity are not enough for shader implementations of skinning that require a maximum of k non-zero weights. If, at the time of optimization, we knew the k influencing weights for each point on the shape, then we could simply force all other weights to be zero via fixed value constraints. Landreneau & Schaefer apply this idea to *sparsify* any given skinning weights (even if manually painted) [2010]. They assume that the k-greatest weights are the *best* choice of k non-zero weights and solve a quadratic program which attempts to match the behavior of the incoming weights by matching its Laplacian:

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2} \int_\Omega (\Delta \hat{w}_j - \Delta w_j)^2 \; dV, \tag{61}$$

$$\text{subject to } w_j(\mathbf{v}) = \varphi_j(\mathbf{v}) \; \forall \mathbf{v} \in H, \tag{62}$$

$$w_j(\mathbf{v}) > 0, \tag{63}$$

$$\sum_{j=1}^{m} w_j(\mathbf{v}) = 1, \tag{64}$$

$$w_j(\mathbf{v}) = 0, \text{ if } j \notin K(\mathbf{v}), \tag{65}$$

where $\hat{w}_j$ is the incoming weight for handle j and $K(\mathbf{v})$ is the list of k-greatest weights at vertex $\mathbf{v}$.

One could imagine trying to iterate on a similar optimization. First, solve while trying to allow only the current k greatest weights at each vertex, then determine a new set of k greatest weights, and then repeat until convergence. This indeed produces slightly more optimal weights than [Landreneau and Schaefer 2010], but seemingly falls quickly into a local minima. Optimizing for truly sparse weights ($\leq k$ non-zeros) while retaining smoothness and other important properties is an exciting open problem.

3.8 Summary

The feature chart in Table 2 is useful as a first-order summary of the discussed methods. However, it would be naive to take only this crude, quantized analysis. For the sake of creating the chart, we need to define each feature as a mathematical test or measurable quantity. Ideally, definitions would adapt to the given application. For example, (unbounded) biharmonic weights oscillate outside the range $[0, 1]$ over the entire domain (e.g. failure of ≥ 1,MONO.,LOCAL). However, this should not be read to imply weights will drastically effect far away regions. In fact, the profile of biharmonic weights (in absolute value) seems to follow the profile of harmonic weights: exponential decay, assuming a domain densely populated with other handles.

Similarly, the feature chart in Table 2 only captures *easy to measure* features. A very important feature of a weight function is its *decay profile*. Intuitively, if we consider weights in a 1D-slice extending (geodesically) between and beyond two handles we hope to see something similar to an *ease* curve (green in Figure 27). This is generally achieved by [Jacobson et al. 2011]. If the slice does not pass directly through the handles, good decay is also achieved by [Baran and Popović 2007a], albeit a bitter slower than desired. This differentiates [Baran and Popović 2007a] from harmonic weights, which according to Table 2 actually obtain *more* desirable features. Harmonic weights have a very undesirable decay profile (red), abruptly approaching the interpolated handles and extending beyond (behind) the other handles. For Euclidean domains, natural neighbor coordinates *nearly* achieve the desired decay profile, but lack derivative continuity at point handle locations. Myopically looking only in the short span between two nearby handles, unconstrained biharmonic weights also achieve an ease-like decay profile, but oscillate beyond the interpolated values, which do not necessary occur as local extrema (blue). The profile of inverse distance based weights (Euclidean or geodesic) is determined by the power in the denominator (d^2 for pink dashed curve in Figure 27). This parameter provides rather crude control. If taken too high, the profile is similar to rounded-out step function. If taken too low, the profile is more reasonable but smoothness and locality begin to suffer. Dionne & de Lasa introduce another parameter to linearly blend absolute distances and squared distances in the denominator [2013]. Inverse distance weights also extend beyond the interpolated values eventually all approaching $1/n$ (rather than one weight dominating). Finally, the lack of monotonicity in these and other weights makes proper analysis of the *decay profile* elusive.

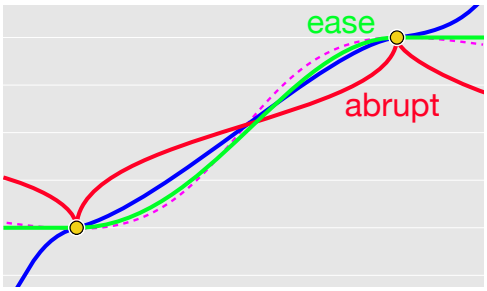


Figure 27: *Illustrations of 1D cross-sections showing various* decay profiles.

In summary, constrained biharmonic weights satisfy the desirable properties that we have shown are important for intuitive control of skinning deformations. This is not to write that the problem is solved. Long computation times prevent tighter interaction loops

during rig creation. True sparsity ($\leq k$ non-zeros per vertex) is also not directly obtained during optimization.

There is also an interesting possibility of combining the tasks of handle creation and placement with weight computation. Tying the two together in the same optimization and measure quality based on the induced deformations would be the ideal problem to solve.

4 Automatic transformations

In the introduction, we broke the skinning pipeline into three major steps. We have explored automatic handle creation and automatic skinning weight computation. With weights and handles defined, the only remaining degrees of freedom in the skinning formula are the handle transformations. Inverse kinematics is the traditional method to unburden the user from needing to specifying all degrees of freedom in a skeletal kinematics chain. These methods typically only consider the skeleton, ignoring the deformation of the shape the skeleton induces. Modern methods now exist that extend to arbitrary handle types and choose remaining degrees of freedom by directly measuring their effect on the deformed shape according to the skinning formula (see e.g. [Jacobson et al. 2012a]).

Alas, this course is too brief to cover this final step. In future revisions, I hope to give this topic its due coverage. For now, I point the reader to relevant works in mesh-based inverse kinematics [Der et al. 2006; Huang et al. 2006; Au et al. 2007] and linear subspace reductions for physically-based simulation [Barbič et al. 2012; Hildebrandt et al. 2011]. The most glaring open problem in this topic is dealing with contacts and collisions in real time [McAdams et al. 2011; Rodolphe et al. 2013].

Acknowledgements

I am indebted to Olga Sorkine-Hornung and Eitan Grinspun for their advice. I thank my coauthors and colleagues Ilya Baran, David Günther, Jan Reininghaus, Jovan Popovic, Tino Weinkauf, Denis Zorin. This work is supported in part by an Intel PhD Fellowship. The Columbia University Computer Graphics Group is supported by the NSF, Intel, The Walt Disney Company, and Autodesk.

References

AMENTA, N., ATTALI, D., AND DEVILLERS, O. 2007. Complexity of delaunay triangulation for points on lower-dimensional polyhedra. In *SODA*, 1106–1113.

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*. 197–232.

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.

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3.

AUJAY, G., HÉTROY, F., LAZARUS, F., AND DEPRAZ, C. 2007. Harmonic skeleton for realistic character animation. In *Proc. SCA*.

BÆRENTZEN, J. A., ABDRASHITOV, R., AND SINGH, K. 2014. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Trans. Graph. 33*, 4.

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

BARAN, I., AND POPOVIĆ, J. 2007. Penalty functions for automatic rigging and animation of 3d characters. Tech. rep., MIT.

BARBIČ, J., SIN, F. S., AND GRINSPUN, E. 2012. Interactive Editing of Deformable Simulations. *ACM Trans. Graph. 31*, 4 (Aug).

BEN-CHEN, M., WEBER, O., AND GOTSMAN, C. 2009. Spatial deformation transfer. In *Proc. SCA*, 67–74.

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

BHARAJ, G., THORMÄHLEN, T., SEIDEL, H.-P., AND THEOBALT, C. 2012. Automatically rigging multi-component characters. *Comput. Graph. Forum 30*, 2.

BLAIR, P. 1994. *Cartoon Animation*. Irvine, CA, USA.

BLOOMENTHAL, J., AND LIM, C. 1999. Skeletal methods of shape manipulation. In *Proc. SMI*.

BOROSÁN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. 2012. RigMesh: Automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph. 31*, 6, 198:1–198:9.

BOROSÁN, P., HOWARD, R., ZHANG, S., AND NEALEN, A. 2010. Hybrid Mesh Editing. 41–44.

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

BOTSCH, M., AND KOBBELT, L. 2005. Real-time shape editing using radial basis functions. *Comput. Graph. Forum 24*, 3, 611–621.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 213–230.

BOTSCH, M., SUMNER, R., PAULY, M., AND GROSS, M. 2006. Deformation transfer for detail-preserving surface editing. In *Proceedings of VMV*, 357–364.

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*.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum 31*, 5, 1657–1667.

CAO, J., TAGLIASACCHI, A., OLSON, M., ZHANG, H., AND SU, Z. 2010. Point cloud skeletons via laplacian-based contraction. In *Proc. of IEEE Conf. on Shape Modeling and Applications*, 187–197.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4 (July), 38:1–38:6.

CHEN, X., SAPAROV, A., PANG, B., AND FUNKHOUSER, T. 2012. Schelling points on 3D surface meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (Aug.).

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? *ACM Trans. Graph. 27*, 3 (Aug.).

CORNEA, N. D., SILVER, D., YUAN, X., AND BALASUBRAMANIAN, R. 2005. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer 21*, 11.

DENG, Z.-J., LUO, X.-N., AND MIAO, X.-P. 2011. Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology 26*, 3.

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

DIONNE, O., AND DE LASA, M. 2013. Geodesic voxel binding for production character meshes. In *Proc. SCA*, 173–180.

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30* (August), 73:1–73:10.

FLOATER, M. S., KÓS, G., AND REIMERS, M. 2005. Mean value coordinates in 3d. *Comput. Aided Geom. Des. 22*, 7.

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

FORSTMANN, S., AND OHYA, J. 2006. Fast skeletal animation by skinned arc-spline based deformation. In *Proc. Eurographics, short papers volume*.

FORSTMANN, S., OHYA, J., KROHN-GRIMBERGHE, A., AND MCDOUGALL, R. 2007. Deformation styles for spline-based skeletal animation. In *Proc. SCA*, 141–150.

GAL, R., SORKINE, O., MITRA, N., AND COHEN-OR, D. 2009. iWires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28*, 3, 33:1–33:10.

GÜNTHER, D., JACOBSON, A., REININGHAUS, J., SEIDEL, H.-P., SORKINE-HORNUNG, O., AND WEINKAUF, T. 2014. Fast and memory-efficient topological denoising of 2D and 3D scalar fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE VIS) 20*, 12, to appear.

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Trans. Graph. 32*, 4.

HILDEBRANDT, K., SCHULZ, C., TYCOWICZ, C. V., AND POLTHIER, K. 2011. Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 5, 119:1–119:11.

HIYOSHI, H., AND SUGIHARA, K. 2000. Voronoi-based interpolation with higher continuity. In *Proc. SOCG*.

HORMANN, K., AND FLOATER, M. S. 2006. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph. 25*, 4.

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

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

HUANG, Q., WICKE, M., ADAMS, B., AND GUIBAS, L. 2009. Shape decomposition using modal analysis. *Comput. Graph. Forum 28*, 2.

HUANG, H., WU, S., COHEN-OR, D., GONG, M., ZHANG, H., LI, G., AND B.CHEN. 2013. L1-medial skeleton of point cloud. *ACM Trans. Graph. 32*.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. 30*, 6.

JACOBSON, A., TOSUN, E., SORKINE, O., AND ZORIN, D. 2010. Mixed finite elements for variational surface modeling. In *Proc. SGP*, 1565–1574.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4, 78:1–78:8.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4.

JACOBSON, A., WEINKAUF, T., AND SORKINE, O. 2012. Smooth shape-aware functions with controlled extrema. In *Proc. SGP*.

JACOBSON, A., KAVAN, L., AND SORKINE-HORNUNG, O. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.*.

JACOBSON, A., PANOZZO, D., ET AL., 2013. libigl: A simple C++ geometry processing library. http://igl.ethz.ch/projects/libigl/.

JACOBSON, A., PANOZZO, D., GLAUSER, O., PREDALIER, C., HILLIGES, O., AND SORKINE-HORNING, O. 2014. Tangible and modular input device for character articulation. *ACM Trans. Graph. 33*, 4, to appear.

JACOBSON, A. 2013. *Algorithms and Interfaces for Real-Time Deformation of 2D and 3D Shapes*. PhD thesis, ETH Zurich.

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

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

JU, T., ZHOU, Q.-Y., VAN DE PANNE, M., COHEN-OR, D., AND NEUMANN, U. 2008. Reusable skinning templates using cage-based deformations. *ACM Trans. Graph. 27*, 5 (Dec.), 122:1–122:10.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3d mesh segmentation and labeling. *ACM Trans. Graph. 29*, 4, 102.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. 31*, 6, 196:1–196:8.

KAVAN, L., SLOAN, P., AND O'SULLIVAN, C. 2010. Fast and efficient skinning of animated meshes. *Comput. Graph. Forum 29*, 2, 327–336.

KUNZE, R., WOLTER, F., AND RAUSCH, T. 1997. Geodesic voronoi diagrams on parametric surfaces. In *Proceedings of the 1997 Conference on Computer Graphics International*.

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.

LE, B. H., AND DENG, Z. 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph. 33*, 4.

LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. *Comput. Graph. Forum 21*, 3.

LEVIN, D., AND LEVI, Z. 2014. Shape deformation via interior rbf. 1062–1075.

LI, X.-Y., AND HU, S.-M. 2013. Poisson coordinates.

LI, Y., WU, X., CHRYSATHOU, Y., SHARF, A., COHEN-OR, D., AND MITRA, N. J. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph. 30*, 4.

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.

LIPMAN, Y., RUSTAMOV, R., AND FUNKHOUSER, T. 2010. Biharmonic distance. *ACM Trans. Graph. 29*, 3.

LIU, P.-C., WU, F.-C., MA, W.-C., LIANG, R.-H., AND OUHYOUNG, M. 2003. Automatic animation skeleton using repulsive force field. In *Computer Graphics and Applications*, 409–413.

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

MANSON, J., AND SCHAEFER, S. 2010. Moving least squares coordinates. In *Proc. SGP*, 1517–1524.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 37:1–37:12.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4, 1400–1423.

MEYER, M., BARR, A., LEE, H., AND DESBRUN, M. 2002. Generalized barycentric coordinates on irregular polygons. *J. Graph. Tools 7*, 1.

MILLER, C., ARIKAN, O., AND FUSSELL, D. 2010. Frankenrigs: Building character rigs from multiple sources. In *Proc. SCA*.

MÖBIUS, A. F. 1827. *Der barycentrische Calcul*.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers*.

NEUMANN, T., VARANASI, K., HASLER, N., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Capture and statistical modeling of arm-muscle deformations. *ACM Trans. Graph. 32*.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3, 609–612.

ÖZTIRELI, A. C., BARAN, I., POPA, T., DALSTEIN, B., SUMNER, R. W., AND GROSS, M. 2013. Differential blending for expressive sketch-based posing. In *Proc. SCA*.

PANOZZO, D., LIPMAN, Y., PUPPO, E., AND ZORIN, D. 2012. Fields on symmetric surfaces. *ACM Trans. Graph. 31*, 4.

PEYRÉ, G., AND COHEN, L. D. 2006. Geodesic remeshing using front propagation. *Int. J. Comput. Vision 69*, 1.

PORANNE, R., OVREIU, E., AND GOTSMAN, C. 2013. Interactive planarization and optimization of 3d meshes. *Comput. Graph. Forum 32*, 1, 152–163.

PRAZAK, M., KAVAN, L., MCDONNELL, R., DOBBYN, S., AND O'SULLIVAN, C. 2010. Moving crowds: A linear animation system for crowd simulation. In *Proc. I3D*, 9:1–9:1.

RODOLPHE, V., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph.*.

RUSTAMOV, R. M. 2011. Multiscale biharmonic kernels. In *Proc. SGP*, 1521–1531.

SCHAEFER, S., AND YUKSEL, C. 2007. Example-based skeleton extraction. In *Proc. SGP*.

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

SCHEMALI, L., THIERY, J.-M., AND BOUBEKEUR, T. 2012. Automatic line handles for freeform deformation. In *Eurographics 2012 (Short)*.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23*, 3, 896–904.

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

SHNEIDERMAN, B. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proc. IUI*, 33–39.

SHOTTON, J., SHARP, T., KIPMAN, A., FITZGIBBON, A., FINOCCHIO, M., BLAKE, A., COOK, M., AND MOORE, R. 2013. Real-time human pose recognition in parts from single depth images. *Commun. ACM 56*, 1.

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

SIBSON, R. 1980. A vector identity for the Dirichlet tessellation. 151–155.

SIBSON, R. 1981. *Interpolating multivariate data*. 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.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proc. SGP*, 179–188.

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

TAGLIASACCHI, A., ALHASHIM, I., OLSON, M., AND ZHANG, H. 2012. Mean curvature skeletons. *Comput. Graph. Forum 31*, 5.

TAGUCHI, Y., WILBURN, B., AND ZITNICK, C. L. 2008. Stereo reconstruction with mixed pixels using adaptive over-segmentation. *Proc. CVPR*.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 32*, 4, 97:1–97:8.

TANG, C., SUN, X., GOMES, A., WALLNER, J., AND POTTMANN, H. 2014. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph.*.

THIERY, J.-M., BUCHHOLZ, B., TIERNY, J., AND BOUBEKEUR, T. 2012. Analytic curve skeletons for 3d surface modeling and processing. *Comput. Graph. Forum*.

THIERY, J.-M., TIERNY, J., AND BOUBEKEUR, T. 2012. Cager: Cage-based reverse engineering of animated 3d shapes. *Comput. Graph. Forum 31*, 8.

TIBSHIRANI, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*.

TOSUN, E., GINGOLD, Y. I., REISMAN, J., AND ZORIN, D. 2007. Shape optimization using reflection lines. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, 193–202.

TSOLI, A., MAHMOOD, N., AND BLACK, M. J. 2014. Breathing life into shape: Capturing, modeling and animating 3d human breathing. *ACM Trans. Graph. 33*, 4.

VAN KAICK, O., FISH, N., KLEIMAN, Y., ASAFI, S., AND COHEN-OR, D. 2014. Shape segmentation by approximate convexity analysis. *ACM Trans. Graph.*.

WACHSPRESS, E. 1975. *A Rational Finite Element Basis*.

WADE, L., AND PARENT, R. E. 2002. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2.

WANG, Y.-S., AND LEE, T.-Y. 2008. Curve-skeleton extraction using iterative least squares optimization. *Visualization and Computer Graphics, IEEE Transactions on 14*, 4 (July), 926–936.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proc. SCA*, 129–138.

WARDETZKY, M., MATHUR, S., KÄLBERER, F., AND GRINSPUN, E. 2007. Discrete Laplace operators: no free lunch. In *Proc. SGP*, 33–37.

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

WEBER, O., BEN-CHEN, M., GOTSMAN, C., AND HORMANN, K. 2011. A complex view of barycentric mappings. *Comput. Graph. Forum 30*, 5.

WEBER, O., PORANNE, R., AND GOTSMAN, C. 2012. Biharmonic coordinates. *Comput. Graph. Forum 31*, 8.

WEINKAUF, T., GINGOLD, Y., AND SORKINE, O. 2010. Topology-based smoothing of 2D scalar fields with C^1-continuity. *Comput. Graph. Forum (proc. EuroVis) 29*, 3, 1221–1230.

XIAN, C., LIN, H., AND GAO, S. 2012. Automatic cage generation by improved OBBs for mesh deformation. *The Visual Computer 28*, 1, 21–33.

XU, H., AND BARBIČ, J. 2014. Signed distance fields for polygon soup meshes. *Graphics Interface 2014*.

YANG, X., SOMASEKHARAN, A., AND ZHANG, J. J. 2006. Curve skeleton skinning for human and creature characters. *Comput. Animat. Virtual Worlds 17*, 3-4, 281–292.

YÜCER, K., JACOBSON, A., HORNUNG, A., AND SORKINE, O. 2012. Transfusive image manipulation. *ACM Trans. Graph. 31*.

ZHU, Y., AND GORTLER, S. J. 2007. 3D deformation using moving least squares. Tech. Rep. Tr-10-07, Harvard University, Cambridge, MA.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2007. Silsketch: automated sketch-based editing of surface meshes. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, 23–30.