



Breathing Life into Shapes

Alec Jacobson
Columbia University

Shape articulation transforms a lifeless geometric object into a vibrant character. Given a sufficient shape deformation interface, the animator or sculptor injects this character with emotion, symbolism, and purpose. Traditional animation and design tools rely on a trained artist's mastery of a small set of basic instruments: pencil and paper, spinning wheel and knife. Computers enrich this toolset dramatically. They not only endow artists with the power to manipulate virtual 2D and 3D scenes, but they also eliminate tedium and expedite prototyping, freeing artists to focus on creative aspects.

With such power comes a temptation to lean entirely on the computer. Computationally intensive animation systems sacrifice real-time feedback for physical accuracy. Offline methods common in film production are perennially too slow for videogames or virtual reality systems. For design tasks, creativity is severely hampered when slow computation short circuits the user out of the shape deformation loop.

My thesis¹ asks, how can we leverage modern computational power to create the best possible shape deformations while maintaining real-time performance as a mandatory invariant? This article summarizes efforts to answer this, culminating in a deformation system with the quality of slow, nonlinear optimization, but at lightning speed. These solutions invite new interfaces for more convenient and expressive user interactions with virtual 2D and 3D shapes. This has immediate impacts on raster and vector graphics editing, 3D character posing and animation, interactive

medical image registration, real-time physically based elastic simulation, and fabrication-aware geometric modeling for 3D printing. The same mathematical machinery has unexpected impacts, too: automatic transfer of anatomical template geometry, improved human pose detection in computer vision, and smooth color interpolation.

After overcoming these challenges, we are at a unique vantage point to scope out the future of real-time animation and shape editing research. The landscape of open problems continues to excite future generations of researchers.

Real-Time Shape Deformation

We anchor our research to a well-studied and popular framework for real-time shape deformation: linear blend skinning.² LBS determines changes to a shape's geometric embedding via a simple formula. Each point \mathbf{v} of a shape's surface (or skin) is given a new position \mathbf{v}' as a weighted combination of affine transformations:

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

In its traditional form, the transformations \mathbf{T}_j are specified by an animator via forward kinematics for each bone of a skeletal hierarchy inside the shape. The associated weights w_j are painted by a trained rigging artist (see Figure 1). LBS enjoys near ubiquity at film and game studios because of its simplicity. Its artifacts are known and predictable.³ Each point's deformation depends only on its weights and the small set of transformations; LBS is embarrassingly parallel. A typical GPU implementation deforms a surface just-in-time before rendering, barely affecting performance. Yet, by relying so heavily on expert users, LBS does not harness modern computer capabilities and power.

Editors' Note

Alec Jacobson received a Eurographics Award for Best PhD Thesis 2014.

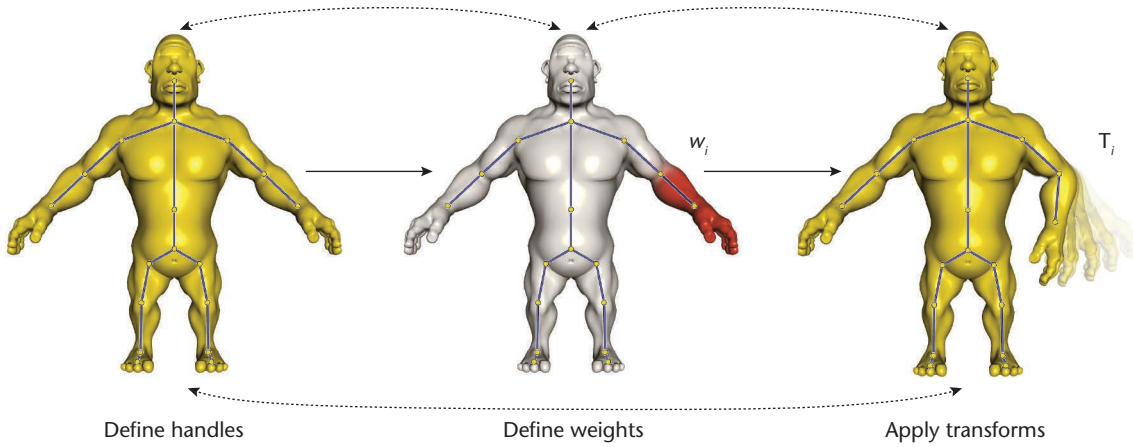


Figure 1. Real-time shape deformation with linear blend skinning. Although skinning decomposes into three major steps, a professional may iterate between each until satisfied. Our research automates weight computation and equips animators with better interfaces.

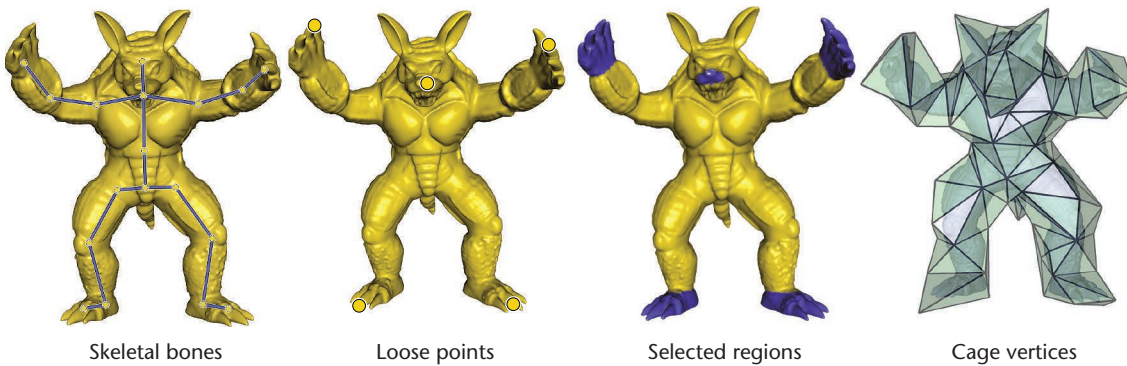


Figure 2. Skinning generalizes to a variety of handle types.

We leverage LBS as the workhorse of our real-time deformation system, thereby guaranteeing real-time performance. The remaining question is how to improve quality and user experience by automating the definition of the LBS degrees of freedom: weights and handle transformations.

Automatic Skinning Weight Computation

Traditionally, LBS deforms 3D shapes via an internal skeleton, but the formula is general. Weights and transforms could belong to a variety of handle types: skeletal bones, loose points on the shape, selected regions of the shape, or vertices of an ex-

terior “cage” (see Figure 2). The deforming shape could be a 3D surface or a 2D cartoon cutout.

Previous methods employ and even advocate for a particular handle type, but this fundamentally limits the user’s interaction with the shape. Users of cage-only techniques suffer when trying to open the piranha’s mouth in Figure 3, whereas a skeleton interface makes the task easy. Similarly, point-based methods lack the precise volume control a cage exerts on the vacuum. A few rotations at points flexibly bend the worm, whereas a skeleton requires many rigid bones. The alligator is best deformed by a combination of different handle types.

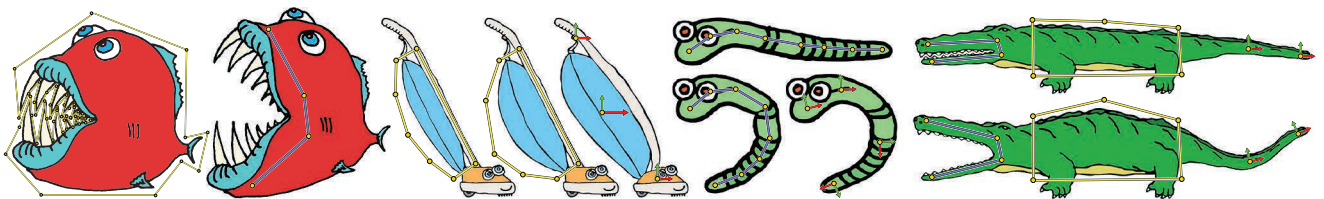


Figure 3. Unifying handle types under a common framework. Handle types are more than just different modeling metaphors; they are interfaces to the creative space of deformations. Each has appropriate and inappropriate uses. Some shapes require multiple handle types acting simultaneously.

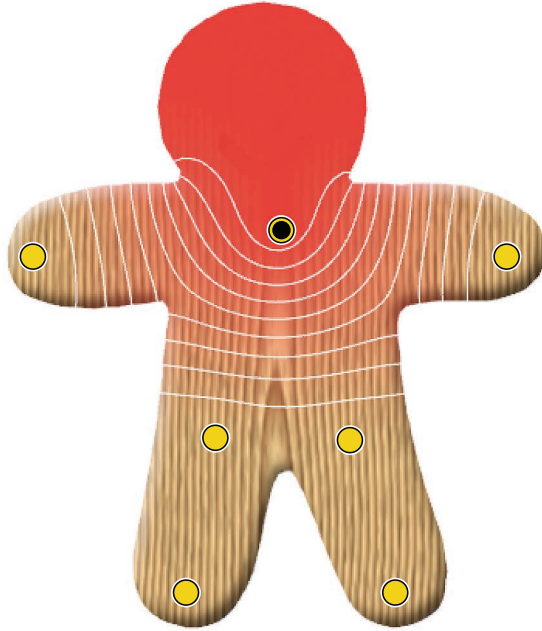


Figure 4. Smooth deformations. Weights should diminish smoothly.

By automating LBS weight computation, we unify these handle types under a common framework. The user can then employ the correct interface or combination for the specific task at hand.

We achieve unification by posing weight computation as a constrained energy minimization problem. Handle types appear as different fixed-value constraints. Energy terms or additional constraints enforce other important properties:

- *Interpolation.* The shape directly beneath or within a handle should be deformed exactly by that handle’s transformation; weights should reach values of one for points at that handle and zero at all other handles.
- *Partition of unity.* If all handles receive the same transform, we expect the entire shape to deform by that transform; weights should sum to one.
- *Smoothness.* To produce smooth deformations, the weights should vary smoothly (See Figure 4).
- *Locality.* Handles should only effect nearby regions not occupied by other handles; weights should be localized.
- *Shape awareness.* Locality should be measured by distances inside or on the shape, rather than through the ambient space around the shape.
- *Boundedness.* All weights should be nonnegative because negative weights lead to unintuitive behavior. Translating a handle to the right will translate regions with negative weights to the left.
- *Monotonicity.* Spurious local minima and maxima in a handle’s weights give the impression that part of the shape is “lagging behind” or that

its influence is “creeping up again.” Weights should be monotonic and free of local extrema.

Weights as Energy Minimizers

Treating our shape as a surface or volume, we begin with a minimization of a smoothness energy—the integrated squared Laplacian—subject to interpolation constraints at handles. Minimizers of this energy are biharmonic functions, $\Delta^2 w_j = 0$.

Biharmonic weights fall off smoothly from one at a handle to zero at neighboring handles. Unfortunately, they continue into negativity only to swing up again toward zero and beyond at the next “ring” of handle constraints. In general, biharmonic weights oscillate over the entire domain.

Taming Biharmonic Oscillations with Bounds

In contrast to biharmonic functions, harmonic functions $\Delta h = 0$ have a host of well-studied properties including guaranteed boundedness and monotonicity. Harmonic weights are successful for cage-based deformations⁴ but fail in general because they are nonsmooth at handles lying atop the shape, causing tearing. Furthermore, without the surrounding cage, harmonic functions are globally supported.

We want to mimic the monotonicity of harmonic functions but retain the smoothness of biharmonic functions. A first-order attempt at this is to explicitly place bounds on our optimized weights: $0 \leq w_j \leq 1$.

Subject to these inequality constraints, finding optimal weights now requires solving a large sparse quadratic program. Thankfully, modern fast solver algorithms exist, including active set or interior point methods. Adding bound constraints has an interesting, if not surprising, side effect: locality. Combined with the partition of unity constraint, the weights will satisfy an ℓ_1 -norm sparsity-inducing constraint.³

Bounds go a long way. Bounded biharmonic weights achieve nearly all desired properties, yet local extrema may still appear (albeit within the $[0, 1]$ bounds).

Copy-and-Paste Monotonicity

Adding monotonicity as a constraint seems intractable. Topological constraints are notoriously difficult. Fortunately, we can gain insight by looking, again, at harmonic functions.

Harmonic weights are not smooth at handles, but are monotonic. For every vertex on our shape, two neighboring vertices exist such that one’s harmonic weight is smaller and another’s is larger.

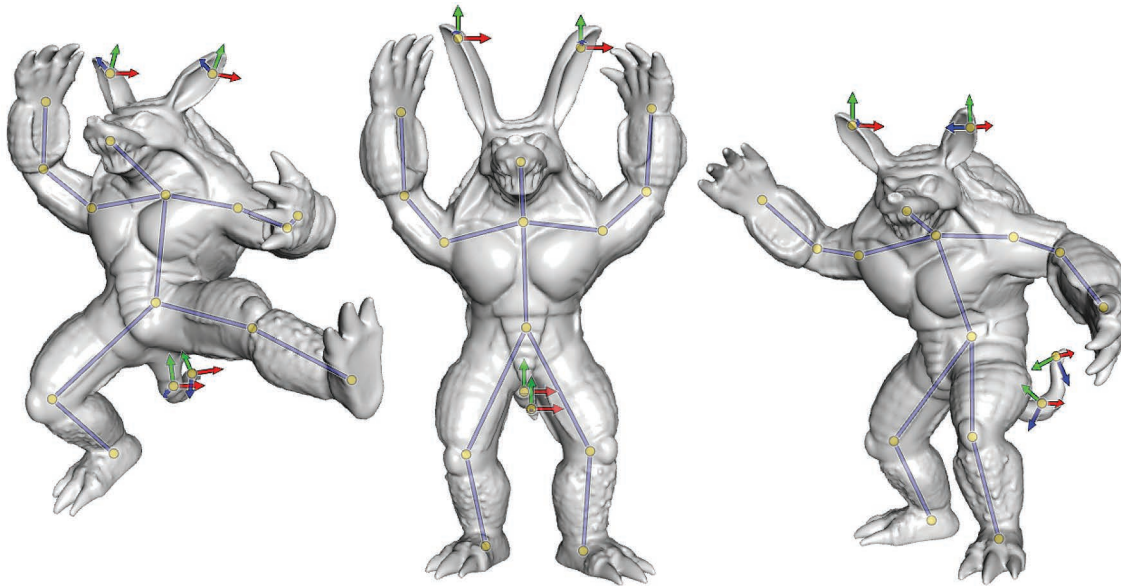


Figure 5. 3D shape modeling and animation. The constrained energy optimization formulation for weight computation easily allows multiple handle types on the same shape (for example, loose points plus an internal skeleton).

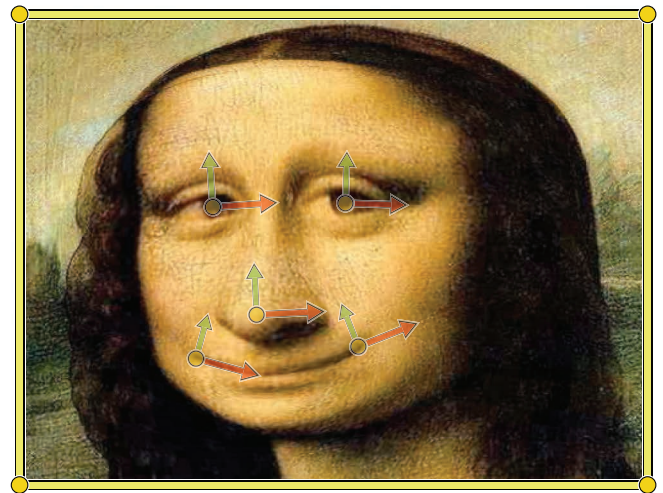
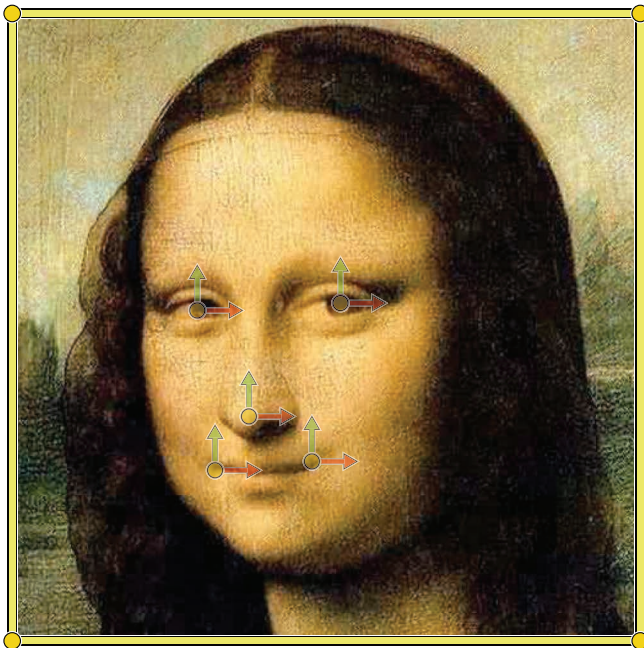


Figure 6. Image editing. A cage alters the image aspect ratio, while point handles sculpt a new expression.

We copy this monotonicity onto our unknown weight functions by requiring that, over a set of edges covering the shape, the weights match the inequality relations of the harmonic weights.

The resulting weights are monotonic and may be reinjected into the optimization as the function from which to copy monotonicity. Thus we can iteratively optimize our weights for further improvement.⁵

The resulting optimization produces high-quality weights, attaining all the qualities listed earlier. They are immediately useful for 3D shape modeling and animation (see Figure 5), 2D cartoon animations, and image editing (see Figure 6). The formulation is purely intrinsic; it does

not depend on the shape's spatial embedding but rather only its metric. Weights can be computed for high-dimensional manifolds. For example, we can lift an image rectangle from \mathbb{R}^2 to \mathbb{R}^5 , augmenting coordinates with RGB values. Biharmonic weights optimized over this "image surface" are now content-aware, condensing variance at image edges. Such bases are useful for a variety of image processing and computer vision tasks, such as colorization (see Figure 7) or registration.⁶

Volumetric Discretization via Winding Numbers

If handles are placed inside rather than on the shape, then we must discretize the shape's inner volume. In 2D, efficient and robust tools create

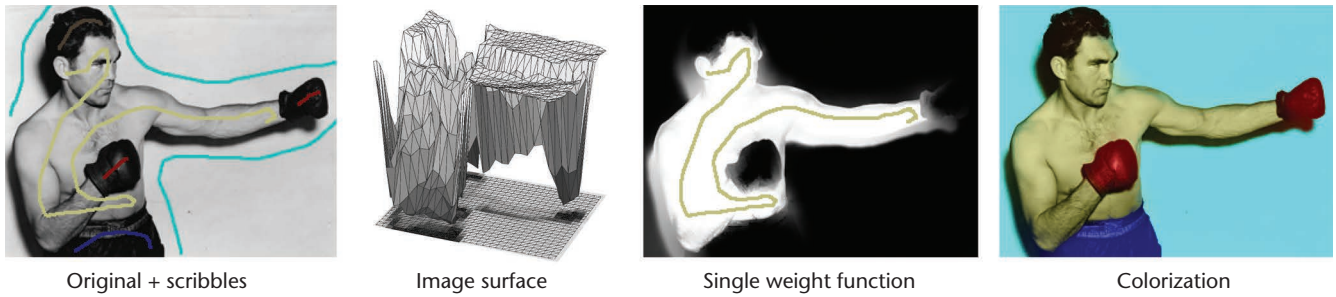


Figure 7. Colorization. By lifting the image rectangle according to its pixel colors to a high-dimension surface, we may compute content-aware weight functions for each scribble. Assigning a color to each and blending creates a colorization.

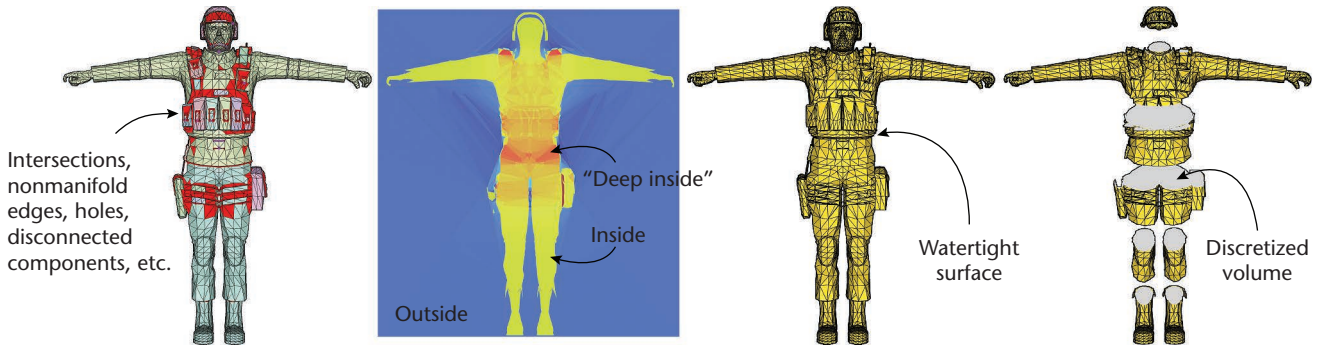


Figure 8. Volumetric meshing. The S.W.A.T. man’s implied solid shape is difficult to mesh as a result of surface artifacts. The generalized winding number measures “insideness,” so the interior can be carved out.

triangle meshes given a boundary polygon.

In 3D, the toolset is less developed. We bootstrap existing constrained Delaunay tetrahedral meshing techniques that maintain the exact input surface and reduce the number of auxiliary inner vertices. However, tetrahedralizers will fail on most character meshes found in the wild, which are littered with self-intersections, open boundaries, nonmanifold edges, and disconnected components. Without a better volume discretization tool, our automatic weights would be limited to squeaky clean “academic” models.

We derived a generalization of the winding number to facilitate volumetric meshing (see Figure 8). The winding number for watertight surfaces maps points inside the shape to one and points outside to zero. We may reformulate this function as a surface integral in spherical coordinates and, in turn, a sum over solid angles subtended by each facet.

We immediately ask, what will this sum compute if the mesh is not closed? If there are overlaps? Or nonmanifold flaps? The answer is a well-behaved—even harmonic—smooth function, corresponding to how much inside the evaluation point is with respect to the given mesh. This generalized winding number may then guide a tetrahedralization.

Tet-meshing and skinning weight optimization⁷ are precomputed, performed once when the user loads a shape and places handles. Then the shape is deformed in real time according to the linear

blend skinning formula. These automatic methods not only eliminates tedious weight painting, but also open the door to new user interfaces for shape deformation that combine different handle types.

Good Weights Are Not Enough

Even with ideal weights, the quality and expressive of linear blend skinning is limited. My thesis addresses two important limitations: supplying only affine transformations spans a small space of deformations, and it is tedious to explore choices of handle transformations.

Expanding the Skinning Shape Space

Our automatic weights allowed us to experiment with a variety of handle types on the same shape. In particular, we noticed how loose points are particularly adept at stretching and twisting actions and how skeletal bones excel at bending rigid parts about a common joint. Conversely, point handles lack the articulation required for bending. Stretching bones leads to scaling artifacts and twisting of bones is packed at joints, where weights must overlap for good bending.

Deconstructing the handle transformations, we isolate the twisting and stretching components responsible for poor behavior. The root of the problem becomes obvious: skinning weights reveal to which bones a point is attached and by how much, but not where along the bone. Armed with



Figure 9. Computing skinning weights for point handles. Standard skinning leads to blow-up artifacts when stretching bones (hands and head) and pucks twisting near joints (elbows). Enriching the space of skinning with an extra set of weights alleviates these problems.

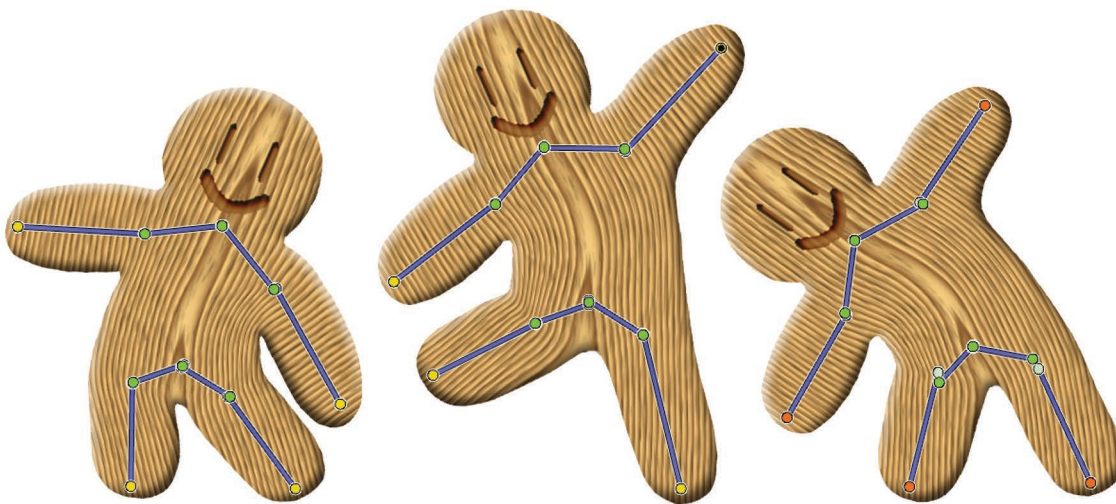


Figure 10. Nonlinear optimization. The user specifies positional constraints at the yellow points and the bones transform automatically. Motivated by traditional animation textbooks, disjoint skeletons articulate limbs while the body falls into place naturally.

a method to compute skinning weights for point handles, we define an auxiliary set of weights for each bone’s endpoints. Intuitively, these map points on the shape to locations along the bone’s line segment. Bones now interpolate twisting and stretching along their lengths, while bending is still controlled by the original skinning weights (see Figure 9).

Automatic Skinning Transformations

Choosing the affine transformation for each bone can be a daunting task for users: e.g. a modest 20-bone skeleton spans a 240-dimensional parametric space. User interfaces alleviate some agony with on-screen rotation widgets linked to forward kinematics hierarchies, but the problem still remains that a convincing, lively animation of the shape requires touching every handle. Ideally, the user specifies a sparse set of high-level

constraints: “turn the head like this,” or “point the foot here.” Then a savvy algorithm assists by inferring appropriate handle transformations to meet the user’s constraints while maintaining a high-quality deformation.

Inverse kinematics assumes a hierarchical skeleton and does not measure the bones’ effect on the deforming shape. Meanwhile, recent sophisticated methods measure mesh deformation quality in terms of local rigidity (aka elasticity), but their nonlinear optimizations are far from real time.

We combine these efforts and propose a reduced optimization. By projecting the search space to that of skinning deformations spanned by the small number of handle transformations, the nonlinear optimization becomes not only tractable but completely decouples computational complexity from shape discretization. The optimization runs at rates far faster than rendering.

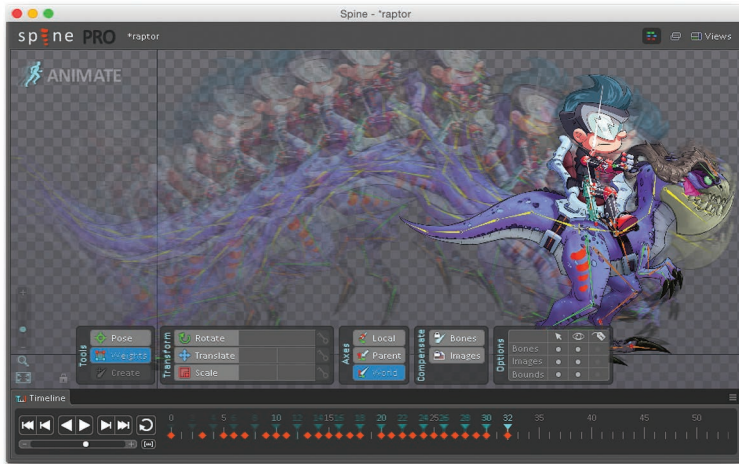


Figure 11. SPINE program from Esoteric Software. SPINE utilizes bounded biharmonic weights in its sophisticated 2D animation software. (Composite screen capture with permission.)

Our optimization acts as a deformation-aware inverse kinematics system. This enables disjoint skeletons, where the constraints on one component influence the other implicitly via their combined effect on the shape (see Figure 10).

Impact and Applications

Advanced real-time deformation tools are now common in 3D animation and 2D image editing software. The skinning weight optimization, subspace reduction, and volumetric discretization techniques presented in my thesis have seen numerous applications and improvements.

Cartoon Animation and Image Editing

GIMP, Adobe Photoshop, and Adobe Aftereffects have integrated interactive handle-based cartoon deformation tools into their traditional image- and video-editing tool suites. The new SPINE program from Esoteric Software—a MAYA-like tool specialized for 2D cartoons—implements bounded biharmonic weights as an automatic skinning feature for sophisticated skeletal animation (see Figure 11).

Our investigation of poly-harmonic functions encourages the continued use of solutions to higher-order partial differential equations in graphics and image processing, when continuity at constraints is important. Biharmonic replaces harmonic blending, to expand “diffusion curves” to include isolated point constraints.

The content-aware weights computed over images lifted to higher dimensions (see Figure 7) led to the development of large-baseline, nonrigid image registration and multiimage editing system, patented by Disney. Skinning-based image deformation also provides a basis for real-time registration of medical images, where interactivity is

critical so physicians can efficiently inject their domain-specific expertise.

Real-Time 3D Deformation

By clearly outlining the list of desirable properties of skinning weights in mathematical terms and then attaining them through a unified framework, our work sets a high standard for future automatic rigging methods. Autodesk’s MAYA has significantly improved its automatic skinning method for internal skeletons, graduating to a volumetric approach. For robustness it opts for voxelization over tetrahedralization, but exploits generalized winding numbers to help classify voxels inside nonwatertight models.

My thesis focuses on deforming triangle meshes (possibly textured). Applying these techniques to other popular shape representations—for example, Bézier splines or Catmull-Clark subdivision surfaces—is challenging. The deformation of a subdivision limit surface is, in general, no longer the limit surface of some subdivision cage. Nonetheless, research has shown how skinning deformations can be applied to subdivision surfaces and splines, via an efficient energy minimization.⁸ This enables LBS to fit into animation and editing workflows without altering the core shape representation.

The locality and smooth behavior of constrained energy-minimizing weights can provide a deformation basis for high-level editing tasks such as optimizing a shape to stand or spin when 3D printed.⁹ LBS with appropriate weights also forms a basis for simulations with physically based dynamics.¹⁰ By adding a simple momentum term to our subspace automatic handle transformation optimization, shapes oscillate under elastic motion over time. The energy term in our optimization amounts to assuming a smoothness prior on the type of preferred deformations. Follow-up works generalize this to consider other priors such as assuming rigid, robot-like components, or organic characters of elastic rubber.

The automation of tedious tasks in the animation and modeling pipelines allows focusing our efforts on improving the user’s direct interface to the deforming character.¹¹ We designed a physical input device that replaces the mouse and keyboard, allowing the user to construct a custom skeleton with joints whose measured angles may be directly applied as skinning transformations (see Figure 12).

Part of the impact of our work is due to our commitment to providing open-source research code. The work summarized here and many related research ideas are implemented in C++ as part of

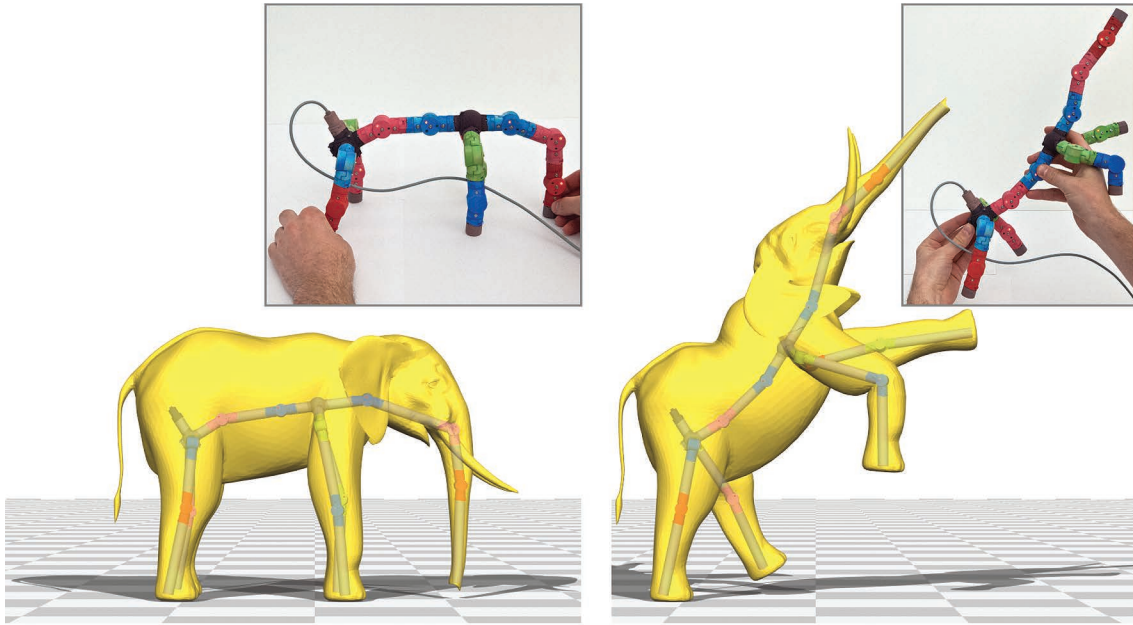


Figure 12. Physical input device. By automating the skinning pipeline, we enable new plug-and-play user interfaces for 3D modeling.

the continuously maintained LIBIGL (<https://github.com/libigl/libigl>) and for Matlab as part of the geometry processing toolbox (<https://github.com/alecjacobson/gptoolbox>).

Open Problems

The impact and applications of new research in real-time shape deformation research are exciting and encouraging for the future. Emerging technologies and accelerating trends continue to motivate the need for high-quality real-time shape manipulation. The democratization of 3D printing is rapidly increasing the population of potential shape modelers. This is no longer a field dominated by trained experts. We will need to examine intelligent algorithms that enable intuitive interfaces for first-time modelers and children. Real-time performance will continue to remain a difficult requirement as high-resolution 2D and 3D data becomes more widespread. Tethering the computation complexity of shape deformation to the shape complexity is out of the question for gigapixel images and micron-detail meshes.

The open problems I find most interesting involve asking deeper, harder questions about the type of shape deformations achievable in real time. I also encourage asking more perceptual questions.

Long Form 2D Cartoon Animation

Modern research has exhausted the possibilities of creating cartoon animations from a single-layer input image. With these tools, it is easy to create a convincing 5 second animation of a gingerbread man waving or an alligator chomping. We are far

from providing the tools needed to facilitate creating long animation sequences or feature films. This is not just a matter of engineering, it is also a matter of designing better algorithms and interfaces. An ideal tool should draw on shared information from animations throughout the long sequence and multiple source images. The recent work on 2.5D cartoon modeling is a step in this exciting direction.¹²

No Free Coordinate Lunch?

The energy minimization framework attempts to satisfy a list of desired properties by adding each as an explicit constraint or energy functional. While our proposed weights go a long way, this question remains to be answered: which properties are strictly at odds with each other? For example, linear precision (the ability of weights to reproduce any linear transformation via translation of alone) seems to be mutually exclusive from smoothness and nonnegativity. Similarly, recent attempts to impose strict constraints on sparsity (at most k nonzero weights everywhere) resulted in nonsmooth functions. Are these constraints impossible to achieve simultaneously?

Local (and Global) Injectivity


A flurry of exciting recent works prevent local shape inversions (where part of a shape becomes flipped inside out or folds over itself). Many methods have beautiful foundations in complex analysis. However, all current solutions are far from real time, requiring nonlinear runtime optimization. For some applications, local injectivity is enough,

but shape modeling and computer animation demand global contact resolution as well. The contact between a character's belly and fingertip is not captured by locally computed differentials. This further complicates optimizations and introduces a collision-detection phase.

Perception and Semantics

During this discussion, I rather cavalierly throw around qualifiers such as “high quality” or “intuitive.” Often during development of a new idea, one can compare techniques on the basis of common sense and discard previous works as “clearly less intuitive.” However, as methods advance, the distinction between opposing methods becomes less clear and it becomes essential to define a metric of success. What is a “high-quality shape deformation”? In modeling or animation, it is the deformation that meets the expectation of the user controlling its parameters. Our colleague Yotam Gingold once quipped about the energy minimization paradigm, “The real energy we're trying to minimize is user surprise.” Perceptual studies on shape deformation and animation will help measure existing methods and give new insights to aspects requiring attention.

Most state-of-the-art nonlinear deformation methods effectively assume shapes are made out of perpetually smooth, homogeneous rubber. While we can derive some semantic understanding of the shape's material properties via the user's placement of handles, why not fire the high-powered cannon of machine learning at the amassing collection of 3D animated shapes? One challenge is to define features suitable for understanding and parameterizing this wide space in the specific context of shape deformation. Another challenge will be to boil down obfuscated learned associations to meaningful wisdom about shape deformation.

Bootstrapping an LBS framework, my thesis provides automatic tools eliminating tedious manual tasks. This not only lowers the barrier of entry of shape deformation to novice and amateur users, but also enables automatic pipelines and invites new interfaces for image editing, 3D shape modeling, and animation. The future of real-time shape deformation is an exciting research frontier, in which I expect fervent activity. 

Acknowledgments

I thank my advisors (Eitan Grinspun, Olga Sorkine-Hornung, and Denis Zorin), coauthors, and colleagues for their help with this work. My dissertation was supported in part by the ERC grant *iModel* (StG-

2012-306877), an SNF award 200021 137879, an Intel Doctoral Fellowship, and a gift from Adobe. Writing this article was supported in part by NSF awards IIS 11-17257 and IIS 14-09286.

References

1. A. Jacobson, “Algorithms and Interfaces for Real-Time Deformation of 2D and 3D Shapes,” PhD dissertation, ETH Zurich, 2013.
2. N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann, “Joint-Dependent Local Deformations for Hand Animation and Object Grasping,” *Proc. Graphics Interface*, 1988, pp. 26–33.
3. A. Jacobson et al., “Skinning: Real-Time Shape Deformation,” *Proc. ACM SIGGRAPH 2014 Courses*, 2014, article no. 24.
4. P. Joshi et al., “Harmonic Coordinates for Character Articulation,” *ACM Trans. Graphics*, vol. 26, no. 3, 2007, article no. 71.
5. D. Günther et al., “Fast and Memory-Efficient Topological Denoising of 2D and 3D Scalar Fields,” *IEEE Trans. Visualization and Computer Graphics*, vol. 20, no. 12, 2014, pp. 2585–2594.
6. A. Jacobson and O. Sorkine, “A Cotangent Laplacian for Images as Surfaces,” tech. report 757, ETH Zurich, 2012.
7. A. Jacobson et al. “Robust Inside-Outside Segmentation via Generalized Winding Numbers,” *ACM Trans. Graphics*, vol 32, no. 4, 2013, article no. 33.
8. S. Liu, A. Jacobson, and Y. Gingold, “Skinning Cubic Bézier Splines and Catmull-Clark Subdivision Surfaces,” *ACM Trans. Graphics*, vol. 33, no. 6, 2014, article no. 190.
9. M. Bäcker et al., “Spin-It: Optimizing Moment of Inertia for Spinnable Objects,” *ACM Trans. Graphics*, vol. 33, no. 4, 2014, article no. 96.
10. Y. Wang et al., “Linear Subspace Design for Real-Time Shape Deformation,” *ACM Trans. Graphics*, vol. 34, no. 4, 2015.
11. A. Jacobson et al., “Tangible and Modular Input Device for Character Articulation,” *ACM Trans. Graphics*, vol. 33, no. 4, 2014, article no. 82.
12. A. Rivers, T. Igarashi, and F. Durand, “2.5D Cartoon Models,” *ACM Trans. Graphics*, vol. 29, no. 4, 2010, article no. 59.

Alec Jacobson is a postdoctoral researcher at Columbia University. Contact him at jacobson@cs.columbia.edu.

Contact department editor Jim Foley at foley@cc.gatech.edu.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.