Fast Automatic Skinning Transformations

Alec Jacobson[1] Ilya Baran[2] Ladislav Kavan[1] Jovan Popović[3] Olga Sorkine[1]
[1]ETH Zurich [2]Disney Research, Zurich [3]Adobe Systems, Inc.

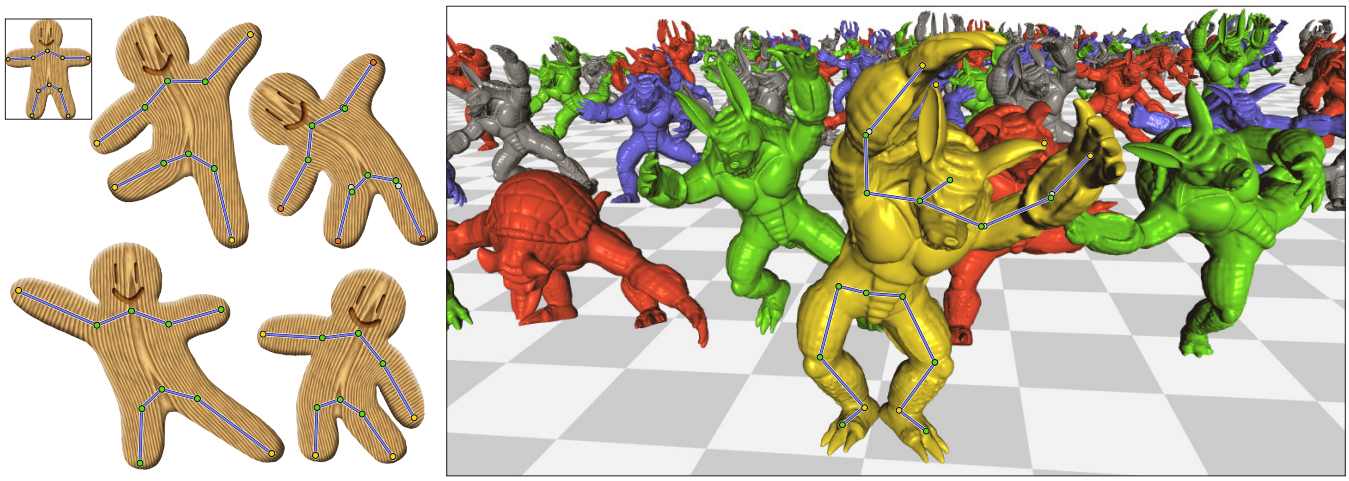

Figure 1: *We present a method to automatically determine 2D and 3D skinning transformations from a sparse set of controls. We achieve high quality deformations by minimizing a nonlinear energy function, while keeping our algorithm extremely fast: skinning transformations for 100 individually animated armadillos (86k triangles each) are computed at 30fps on a single CPU core.*

Abstract

Skinning transformations are a popular way to articulate shapes and characters. However, traditional animation interfaces require all of the skinning transformations to be specified explicitly, typically using a control structure (a rig). We propose a system where the user specifies only a subset of the degrees of freedom and the rest are automatically inferred using nonlinear, rigidity energies. By utilizing a low-order model and reformulating our energy functions accordingly, our algorithm runs orders of magnitude faster than previous methods without compromising quality. In addition to the immediate boosts in performance for existing modeling and real time animation tools, our approach also opens the door to new modes of control: disconnected skeletons combined with shape-aware inverse kinematics. With automatically generated skinning weights, our method can also be used for fast variational shape modeling.

Keywords: skinning, shape modeling, variational methods, as-rigid-as-possible.

Links: DL PDF WEB VIDEO

1 Introduction

Articulation adds life to geometric shapes in two steps. The rigging stage establishes parameters that provide intuitive control of shape geometry. Once these parameters are specified, the deformation stage computes the actual shape to generate poses. One popular choice of these parameters are joint angles for hierarchies of rigid transformations because they match the skeletal structure of humans, animals, and other characters. However, the best set of articulation parameters depends on the task so free-form deformations, blend shapes, cages, and others are also used, often at the expense of tedious manual tuning.

As shape deformation needs to be computed at every animation frame for every character, this process must be extremely fast, especially in applications such as computer games and haptics. The most common method is linear blend skinning (LBS), whose articulation parameters are affine transformations, each associated with a *handle*. Originally, handles were just bones, but previous work has explored the use of point handles, region handles, cage vertices, and even abstract handles not associated with any well-defined shape and perhaps not directly controlled by the user.

Handle transformations that drive LBS are typically obtained from motion capture, physical simulation, or manual posing and keyframing. In many cases, however, it is desirable to only specify a subset of their degrees of freedom. For example, a motion capture sequence may not have enough data to fully define the spine configuration, or an artist may want to specify only a subset of controls and have the rest inferred in a reasonable way. In some cases, it is useful to specify only the translational component of a handle transformation.

This paper presents a method for deforming a shape as naturally as possible when only a subset of the degrees of freedom are specified. Our method computes the unspecified degrees of freedom by minimizing an elastic energy over the shape [Chao et al. 2010]. Our method is extremely efficient, allowing it to be integrated into the

Figure 2: *Traditional animators sometimes sketch bones only in limbs, using them to infer the remaining shape [Blair 1994].*

skinning pipeline, with computational complexity often dominated by LBS. Using a single CPU thread, our approach can compute missing degrees of freedom for one hundred armadillos each with 17 handles and 86k triangles at 30 frames per second (see Figure 1).

To attain our high performance, we speed up the energy minimization in two ways. First, we express the energy only in terms of handle transformations. Second, we use the input skinning weights to determine parts of the shape that are likely to be transformed similarly and simplify the deformation energy accordingly. These two optimizations complement each other: the error incurred by simplifying the energy function tends to be orthogonal to the subspace of a good LBS rig. In fact, we show this reduction often visually improves the deformations by regularizing the energy function. Moreover, our method guarantees deformations as smooth as the input skinning weights.

The ability to leave some handle transformations unspecified opens up new possibilities for both posing and rigging. For posing, computing missing degrees of freedom results in behavior that is similar to inverse kinematics, but shape-aware. For rigging, the user can deliberately specify extra handles in areas that require additional flexibility without having to worry about their transformations. Additionally, extra handles can offset some undesirable behavior of linear blend skinning and allow our method to be applied to general shape deformation. Given manipulation handles, weights automatically generated by previous works can be augmented with additional extra weights attached to abstract handles. These additional weights are generated with a simple algorithm. With them our method produces results on par with high-quality nonlinear shape deformation methods [Botsch et al. 2007], but running orders of magnitude faster.

2 Related work

Inverse kinematics. IK infers missing degrees of freedom for deformation under some constraints. Traditionally, IK is performed on the skeleton of the character, irrespective of the shape itself, and cannot handle disconnected skeletons or surface handles. MeshIK methods [Sumner et al. 2005; Fröhlich and Botsch 2011] use deformation examples to navigate in the space of plausible poses while interpolating prescribed surface region handles. To speed up the MeshIK optimization, a reduced model similar to ours has been presented [Der et al. 2006]. Unlike our model, it requires examples to infer a reduced model and the energy objective. At runtime, they must factor a dense matrix at every iteration, whereas in our method this can be pre-computed, since the system matrix remains unchanged.

Improving skinning quality. Our weights enrichment strategy is related to previous work aimed at improving skinning quality. Several works used example shapes for this purpose [Lewis et al. 2000; Wang et al. 2007]; similarly to us, Mohr and Gleicher [2003] enriched the space of the original weights, but using shape examples. Nonlinear skinning techniques improve skinning deformations quality without additional data [Kavan et al. 2008], but the nonlinear nature makes such skinning inconvenient as a reduced model. Weights defined for abstract handles can be used to automatically approximate nonlinear skinning methods expressed in closed-form with LBS [Kavan et al. 2009]. Spline skeletons [Forstmann and Ohya 2006; Forstmann et al. 2007; Yang et al. 2006] often produce better skinning deformations, but again, they make skinning nonlinear. Others, [Wang and Phillips 2002; Merry et al. 2006; Jacobson and Sorkine 2011], generalize the weight functions to matrix-valued or vector-valued forms, modifying the basic LBS formulation. Contrary to these approaches, our method is easier to use and fits into the existing animation pipeline since we only rely on the standard LBS, which is of practical importance due to the number of optimized platform-specific LBS implementations.

Deformation. Recent shape deformation research has concentrated on two main fronts: modeling *physically plausible* deformations (typically by means of variational optimization), and exploring very fast and parallelizable closed-form deformations.

variational vs closed-form deformaiton

Many works deal with the question of how a shape should deform given arbitrary (user-defined) modeling constraints, typically prescribed locations for some points or regions on the surface. Several energy functionals that the deformed shape should minimize (under the modeling constraints) were proposed; these energies typically measure a form of elastic shape distortion. Quadratic energies that lead to linear optimization problems are discussed in the survey of Botsch and Sorkine [2008]. They are robust and can be optimized at interactive framerates for moderately sized meshes, but suffer from linearization artifacts. Nonlinear energies, e.g. [Botsch et al. 2006; Au et al. 2006; Sorkine and Alexa 2007; Chao et al. 2010] provide higher-quality deformations but are slower to optimize, and their complexity grows nonlinearly with the size of the mesh. In this work, we take advantage of the special structure of some recently proposed "as-rigid-as-possible" (ARAP) elastic energies [Igarashi et al. 2005; Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010] and modify their formulation so that it can be extremely efficiently minimized in the subspace of skinning deformations.

The second stream of shape deformation research dates back to the FFD framework [Sederberg and Parry 1986] that avoids global variational optimization. Given a set of control objects (handles), such as cages or points on the shape or skeletons, the question is how to define the *influence* of each handle on each point of the shape. Then, the user-defined transformations at the handles are propagated to each point on the shape by simple local weighted combination, using the influence values. In the case of cages, the influences are defined as generalized barycentric coordinates (see e.g. [Ju et al. 2005; Joshi et al. 2007; Lipman et al. 2008; Weber et al. 2009]) and only translations need to be provided for the cage vertices. For skeletal bones several methods exist to define automatic weights [Baran and Popović 2007; Wareham and Lasenby 2008]; full affine transformations need to be provided per bone, and these are linearly combined using the LBS formula. Higher-

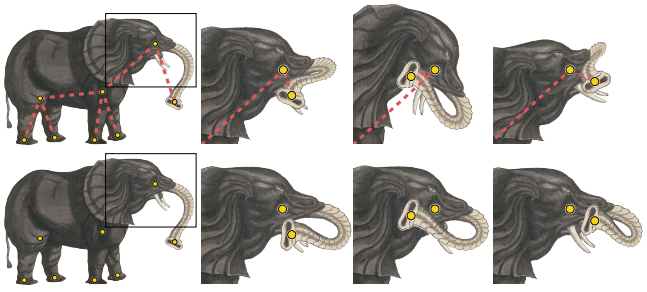

Figure 3: *Elephant controlled by 7 points. The user-provided* pseudo-edges *[Jacobson et al. 2011] (top row) fail to produce the expected buckling (bottom, our method).*

order barycentric coordinates [Langer and Seidel 2008] allow full affine transformations per cage vertex, offering more flexible deformations. Bounded biharmonic weights [Jacobson et al. 2011] can be computed for all common handle types above. MLS deformations [Schaefer et al. 2006] use simple inverse-distance based weights but combine the handle influences in a nonlinear manner by local least-squares optimization.

The above approaches enjoy much better performance than variational methods due to their local, parallelizable nature. However, as mentioned, they either require the user to provide full affine transformations per handle, or manipulate cages, which may be difficult for some desired shapes and deformations. Simple heuristic solutions, such as "pseudo-edges" [Jacobson et al. 2011] do not consider their effect on the resulting shape, leading to unintuive response (see Figure 3).

problem with fast closed-form deformation: need full specification of affine transformation of handles

Reduced models. Methods based on reduced models attempt to combine the benefits of variational and closed-form deformation methods. They minimize some shape energy, but instead of doing so on the shape itself, they employ a much smaller number of degrees of freedom and a simpler (faster) deformation subspace, such as cage-based deformations [Huang et al. 2006; Ben-Chen et al. 2009; Weber et al. 2009; Borosán et al. 2010], LBS with skeletons [Shi et al. 2007], LBS with point/region handles [Au et al. 2007; Sumner et al. 2007], or linear subspace of dominant modes of the deformation energy [Hildebrandt et al. 2011]. Such approaches achieve significant improvements in shape articulation, but struggle to reach rates required by time-critical applications while retaining high quality, flexible deformation, and artistic freedom. Many model-reduction schemes do not guarantee interpolation of the user-specified constraints, e.g., because the degrees of freedom are not sufficient to do so. With the exception of [Au et al. 2007], the reduced models are constructed per shape, irrespective of the user constraints (handles), and the deformations generally have a global nature as a result (when manipulating a handle, shape parts far away from it may move even if nearby handles are constrained). Also note that even heavily parallelized GPU implementations such as [Weber et al. 2009] are still an order of magnitude slower than our method running on a single CPU core.

Among the above-mentioned techniques, [Au et al. 2007; Shi et al. 2007] are closest to ours in terms of the technical setup. Au et al. [2007] take user-provided surface-region handles and construct a reduced model by creating additional handles that are isolines of a harmonic propagation field sourced at the input handles. They then minimize a nonlinear Laplacian energy [Au et al. 2006] on the LBS subspace created by those handles. Contrary to our method, their optimization still requires updates of the entire unreduced mesh in each iteration, making it slow in comparison. MeshPuppetry [Shi et al. 2007] uses a skeletal LBS reduced model to optimize a similar nonlinear Laplacian energy and additional constraints such as the balance of the character. They optimize both the skinning weights and the skeleton transformations simultaneously, which again results in time complexity dependent on the number of primitives. In contrast, we assume fixed skinning weights provided by the user and design deformations that *respect* the artistic intention encapsulated in the weights. This allows us to formulate an algorithm with time complexity independent of the number of primitives of the input model. We also explore new modes of control, such as combination of skeletal and point controls, as well as disconnected skeletons (see Figure 1). Disconnected skeletons are especially interesting as traditional animators often employ them as a starting point for creating lifelike characters (see Figure 2).

Decimating the input mesh, Manson and Schaefer [2011] optimize an ARAP energy [Sorkine and Alexa 2007] at a coarse resolution. They then reintroduce fine details by constrained local optimization. While faster than full-scale optimization, the nonlinear upsampling step is still orders of magnitude slower than our skinning model.

Our use of skinning weights to cluster vertices and further simplify optimization is related to previous works that assume rigid patches during example-driven deformation. Pekelny and Gotsman [2008] find clusters to track the motion of piecewise rigid shapes. Huang et al. [2008] optimize the assignment of overlapping rigid clusters to define a locally as-rigid-as-possible deformation for shape registration. Most similar to our method, Der et al. [2006] cluster surface vertices to simplify a nonlinear optimization over a reduced, skinning model. Their method uses example poses to define skinning weights and then clusters vertices according to their k-greatest skinning weights.

Reduced models are also common in physical elastic simulation, typically employing dense subspace models [Barbič and James 2005; An et al. 2008], where they are used to efficiently integrate the equations of motion. As discussed above, skinning provides the advantage of local control and is starting to be used in physical simulation [Gilles et al. 2011; Faure et al. 2011].

3 Method

Denote by $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$ ($d = 2$ or 3) the rest-pose vertex positions of the input mesh $\mathcal{M}$. The user specifies a set of control handles $\mathcal{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_m\}$, which could be bones of a skeleton, points or regions on the shape. To deform the shape, the user must to specify affine transformations $\mathbf{T}_j \in \mathbb{R}^{d \times (d+1)}$ for each handle $\mathbf{h}_j$. We denote the deformed vertex positions by $\mathbf{v}'_1, \ldots, \mathbf{v}'_n$. The handle transformations lead to modeling constraints $\mathbf{h}'_j = \mathbf{T}_j \mathbf{h}_j$.

There are many methods to compute a deformed shape based on the given modeling constraints. The fastest method is linear blend skinning (LBS), which in addition to the inputs above also requires skinning weight functions $w_j : \mathcal{M} \to \mathbb{R}$. For each point $\mathbf{p}$ on the shape, $w_j(\mathbf{p})$ specifies the amount of influence of $\mathbf{T}_j$ on $\mathbf{p}$. The skinning weights are often painted manually by specialized rigging artists, but may also be computed automatically using recent methods. The LBS deformation of $\mathcal{M}$'s vertices is then given by

d: dimension, i.e. 3D
m: number of handles

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

This formula can be equivalently expressed in matrix form:

$$\mathbf{V}' = \mathbf{M}\mathbf{T}, \tag{2}$$

where $\mathbf{V}' \in \mathbb{R}^{n \times d}$ is the matrix whose rows are the deformed vertex positions, $\mathbf{M} \in \mathbb{R}^{n \times ((d+1)m)}$ is the matrix combining rest-pose vertex positions $\mathbf{v}_i$ with vertex weights $w_j(\mathbf{v}_i)$, and $\mathbf{T} \in \mathbb{R}^{((d+1)m) \times d}$ stacks transposed transformation matrices $\mathbf{T}_j$ (see e.g. [Kavan et al. 2010] for details).

3.1 Automatic degrees of freedom

Specifying all the degrees of freedom for all affine transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m$ could be burdensome. It is often easier and more intuitive for the user to specify translations only, dragging the control handles around. However, if the $\mathbf{T}_j$'s consist of translations alone, Eq. 1 leads to unnatural, sheared deformations. Many other linear deformation formulations suffer from the same problem of "translation-insensitivity" [Botsch and Sorkine 2008]: when specifying translations at handles, no local rotations are generated by the deformation, and the shape and its details are distorted as a result. We can also leave some transformations $\mathbf{T}_j$ entirely unconstrained, which corresponds to shape-aware inverse kinematics.

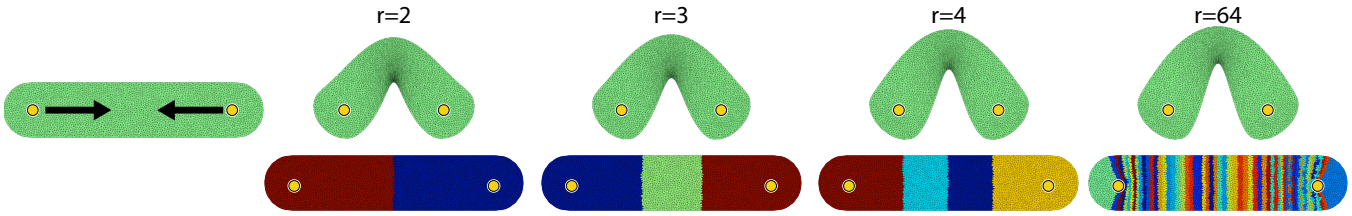

Figure 4: *2D deformation with various numbers of rotation clusters, r. Notice that even if only two rotation clusters are used the deformation is smooth and finds a reasonable shape. Increasing the number of rotation clusters improves this shape with diminishing returns.*

We propose overcoming this problem by *optimizing* the remaining degrees of freedom of the handle transformations $\mathbf{T}_j$ that the user did not specify explicitly. We wish to find such $\mathbf{T}_j$'s that the deformed shape minimizes a deformation energy E under the imposed modeling constraints and the LBS setting (Eq. 2). We can write the user-specified modeling constraints as:

$$\underbrace{\begin{bmatrix} \mathbf{I}_{\text{full}} \\ \mathbf{M}_{\text{pos}} \end{bmatrix}}_{\mathbf{M}_{\text{eq}}} \mathbf{T} = \underbrace{\begin{bmatrix} \mathbf{T}_{\text{full}} \\ \mathbf{P}_{\text{pos}} \end{bmatrix}}_{\mathbf{P}_{\text{eq}}}, \quad (3)$$

T_full is simply same as T

where $\mathbf{I}_{\text{full}}$ are identity blocks corresponding to the fully constrained transformations and $\mathbf{T}_{\text{full}}$ are their user-provided values (as in traditional skinning) and $\mathbf{M}_{\text{pos}}$ are rows of $\mathbf{M}$ corresponding to positional constraints on $\mathcal{M}$'s vertices and $\mathbf{P}_{\text{pos}}$ their values. All constraints together can be concisely written as $\mathbf{M}_{\text{eq}}\mathbf{T} = \mathbf{P}_{\text{eq}}$, where $\mathbf{M}_{\text{eq}} \in \mathbb{R}^{c\times((d+1)m)}$ and $\mathbf{P}_{\text{eq}} \in \mathbb{R}^{c\times d}$ and our construction guarantees they are feasible and full rank.

To obtain the remaining, unconstrained degrees of freedom of $\mathbf{T}_j$'s (i.e., linear components of positionally constrained handles and all components of unconstrained handles), we propose minimizing a shape deformation energy $E : \mathbb{R}^{n\times d} \to \mathbb{R}^{+}$. $E(\mathbf{V}')$ measures the deformation between the rest-pose shape $\mathcal{M}$ and the deformed shape $\mathcal{M}'$. We focus on energies E measuring local deviation from rigidity, advantageous for good detail preservation and intuitive elastic behavior [Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010]. This family of energy functions can be expressed as:

simply ARAP

$$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2, \quad (4)$$

where $\mathbf{R}_1, \ldots, \mathbf{R}_r \in SO(d)$ are local rotations, $\mathcal{E}_1, \ldots, \mathcal{E}_r$ are their corresponding sets of edges (see Figure 5), and $c_{ijk} \in \mathbb{R}$ are weighting coefficients, typically the familiar cotangent weights [Chao et al. 2010]. It is convenient to rewrite Eq. 4 in matrix notation.

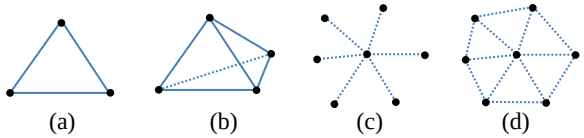

Figure 5: *Typical edge sets: triangle (a), tetrahedron (b), spokes (c), spokes and rims (d).*

We start by separating terms quadratic and linear in $\mathbf{V}'$:

$$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}E_2(\mathbf{V}') - E_1(\mathbf{V}', \mathbf{R}) + \text{const}, \quad (5)$$

$$E_2(\mathbf{V}') = \sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}(\mathbf{v}'_i - \mathbf{v}'_j)^{\mathsf{T}}(\mathbf{v}'_i - \mathbf{v}'_j), \quad (6)$$

$$E_1(\mathbf{V}', \mathbf{R}) = \sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}(\mathbf{v}'_i - \mathbf{v}'_j)^{\mathsf{T}}\mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j), \quad (7)$$

The quadratic term can be written in matrix form as:

$$E_2(\mathbf{V}') = \sum_{k=1}^{r}\text{tr}(\mathbf{C}_k\mathbf{A}_k^{\mathsf{T}}\mathbf{V}'\mathbf{V}'^{\mathsf{T}}\mathbf{A}_k), \quad (8)$$

where $\mathbf{A}_k \in \mathbb{R}^{n\times|\mathcal{E}_k|}$ is directed incidence matrix corresponding to (arbitrarily oriented) edges $\mathcal{E}_k$ and $\mathbf{C}_k \in \mathbb{R}^{|\mathcal{E}_k|\times|\mathcal{E}_k|}$ is a diagonal matrix with weights c_{ijk}. Due to the properties of the trace, we can rewrite this as:

$$E_2(\mathbf{V}') = \text{tr}\left(\mathbf{V}'^{\mathsf{T}}\left(\sum_{k=1}^{r}\mathbf{A}_k\mathbf{C}_k\mathbf{A}_k^{\mathsf{T}}\right)\mathbf{V}'\right) = \text{tr}(\mathbf{V}'^{\mathsf{T}}\mathbf{L}\mathbf{V}'),$$

where we denoted the middle sum as $\mathbf{L} \in \mathbb{R}^{n\times n}$, which for all energies discussed in the literature [Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010] is the standard symmetric, cotangent Laplacian matrix (up to a constant scale factor). Similarly, the linear term can be written as:

$$E_1(\mathbf{V}', \mathbf{R}) = \sum_{k=1}^{r}\text{tr}(\mathbf{C}_k\mathbf{A}_k^{\mathsf{T}}\mathbf{V}'\mathbf{R}_k\mathbf{V}^{\mathsf{T}}\mathbf{A}_k) \quad (9)$$

$$= \text{tr}\left(\left(\textstyle\sum_{k=1}^{r}\mathbf{R}_k\mathbf{V}^{\mathsf{T}}\mathbf{A}_k\mathbf{C}_k\mathbf{A}_k^{\mathsf{T}}\right)\mathbf{V}'\right) = \text{tr}(\mathbf{R}\mathbf{K}\mathbf{V}'), \quad (10)$$

where $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_r)$ and $\mathbf{K} \in R^{dr\times n}$ stacks differential rest pose coordinates $\mathbf{V}^{\mathsf{T}}\mathbf{A}_k\mathbf{C}_k\mathbf{A}_k^{\mathsf{T}}$. We therefore obtain:

$$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}\text{tr}(\mathbf{V}'^{\mathsf{T}}\mathbf{L}\mathbf{V}') - \text{tr}(\mathbf{R}\mathbf{K}\mathbf{V}') + \text{const} \quad (11)$$

In this form, it is easy to formulate our reduced-order optimization by plugging in the linear blend skinning formula $\mathbf{V}' = \mathbf{M}\mathbf{T}$:

$$\begin{aligned} \underset{\mathbf{T},\mathbf{R}}{\text{argmin}} \quad & \frac{1}{2}\text{tr}(\mathbf{T}^{\mathsf{T}}\tilde{\mathbf{L}}\mathbf{T}) - \text{tr}(\mathbf{R}\tilde{\mathbf{K}}\mathbf{T}) \\ \text{subject to} \quad & \mathbf{M}_{\text{eq}}\mathbf{T} = \mathbf{P}_{\text{eq}}, \quad \mathbf{R} \in SO(d)^r \end{aligned} \quad (12)$$

where $\tilde{\mathbf{L}} = \mathbf{M}^{\mathsf{T}}\mathbf{L}\mathbf{M}$ and $\tilde{\mathbf{K}} = \mathbf{K}\mathbf{M}$.

To solve this optimization problem, we follow the local-global approach of Sorkine and Alexa [2007]. First we fix $\mathbf{T}$ and solve for $\mathbf{R}$ (local step). Then we fix $\mathbf{R}$ and solve for $\mathbf{T}$ (global step).

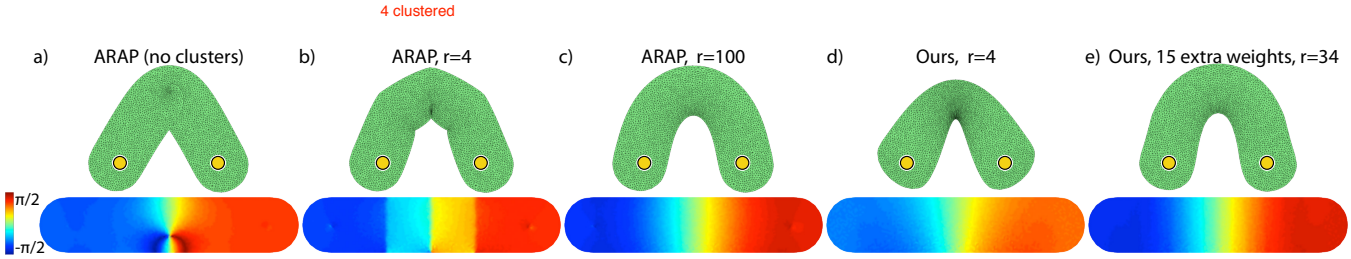

Figure 6: *Left to right: unreduced per-triangle ARAP introduces a singularity in the rotation field (visualized below). Per-triangle ARAP energy with $r = 4$ prevents this singularity; the non-smooth transitions are no longer noticeable with $r = 100$. Our method (using smooth weights) is always smooth, even with $r = 4$. After enriching our deformation subspace with 15 additional weights we obtain the desired result.*

Local step. For fixed $\mathbf{T}$, we are left maximizing $\mathrm{tr}(\mathbf{RS})$, where $\mathbf{S} = \tilde{\mathbf{K}}\mathbf{T}$ is constant. This amounts to maximizing each $d \times d$ block $\mathrm{tr}(\mathbf{R}_i\mathbf{S}_i)$ individually. It is well known [Sorkine and Alexa 2007] that the rotation maximizing the trace is $\mathbf{R}_i = \mathbf{Q}_i^\mathsf{T}\mathbf{U}_i^\mathsf{T}$, where $\mathbf{S}_i = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{Q}_i$ is the singular value decomposition. For $d = 3$ we employ the optimized SVD routines by McAdams and colleagues [2011] that avoid reflections, i.e., guarantee $\det(\mathbf{R}_i) > 0$.

Global step. For fixed $\mathbf{R}$, Eq. 12 turns into a quadratic minimization problem with linear equality constraints. The constraints can be handled by introducing a matrix $\mathbf{\Lambda} \in \mathbb{R}^{c\times d}$ of Lagrange multipliers, arriving at the Lagrangian:

additional constraint by bounded biharmonic weights

$$\mathscr{L} = \frac{1}{2}\mathrm{tr}(\mathbf{T}^\mathsf{T}\tilde{\mathbf{L}}\mathbf{T}) - \mathrm{tr}(\mathbf{R}\tilde{\mathbf{K}}\mathbf{T}) + \mathrm{tr}(\mathbf{\Lambda}^\mathsf{T}(\mathbf{M}_{\mathrm{eq}}\mathbf{T} - \mathbf{P}_{\mathrm{eq}}))$$

Recalling standard matrix calculus identities, we differentiate:

$$\frac{\partial\mathscr{L}}{\partial\mathbf{T}} = \tilde{\mathbf{L}}\mathbf{T} - \tilde{\mathbf{K}}^\mathsf{T}\mathbf{R}^\mathsf{T} + \mathbf{M}_{\mathrm{eq}}^\mathsf{T}\mathbf{\Lambda} \quad (13)$$

$$\frac{\partial\mathscr{L}}{\partial\mathbf{\Lambda}} = \mathbf{M}_{\mathrm{eq}}\mathbf{T} - \mathbf{P}_{\mathrm{eq}} \quad (14)$$

where we exploit the symmetry of $\tilde{\mathbf{L}}$. Setting these derivatives to zero, we obtain:

$$\begin{bmatrix} \tilde{\mathbf{L}} & \mathbf{M}_{\mathrm{eq}}^T \\ \mathbf{M}_{\mathrm{eq}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{T} \\ \mathbf{\Lambda} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{K}}^\mathsf{T}\mathbf{R}^\mathsf{T} \\ \mathbf{P}_{\mathrm{eq}} \end{bmatrix} \quad (15)$$

With enough constraints to determine all translational degrees of freedom of $\tilde{\mathbf{L}}$, the system matrix is non-singular. We can thus precompute its inverse $\mathbf{\Pi}$ and express the solution as:

$$\mathbf{T} = [\mathbf{\Pi}_1\ \mathbf{\Pi}_2] \begin{bmatrix} \tilde{\mathbf{K}}^\mathsf{T}\mathbf{R}^\mathsf{T} \\ \mathbf{P}_{\mathrm{eq}} \end{bmatrix} = \mathbf{\Gamma}_{\mathrm{solve}}\mathbf{R}^\mathsf{T} + \mathbf{\Phi}_{\mathrm{solve}},$$

where $\mathbf{\Gamma}_{\mathrm{solve}} \in \mathbb{R}^{(d+1)m\times dr}$ and $\mathbf{\Phi}_{\mathrm{solve}} \in \mathbb{R}^{(d+1)m\times d}$ can be precomputed.

Final algorithm. One iteration of the reduced alternating optimization can be summarized as follows:

1. $\mathbf{S} = \tilde{\mathbf{K}}\mathbf{T}$
2. Turn $\mathbf{S}$ into $\mathbf{R}$ using SVDs
3. $\mathbf{T} = \mathbf{\Gamma}_{\mathrm{solve}}\mathbf{R}^\mathsf{T} + \mathbf{\Phi}_{\mathrm{solve}}$

The only nonlinear step is 2, consisting of r SVDs of $d \times d$ matrices. While the linear steps 3 and 1 could be combined together, it is typically more efficient to use two $dr \times (d+1)m$ and $(d+1)m \times dr$ matrices rather than one $dr \times dr$ matrix. The complexity is independent of the number of vertices n, however, with r on the order of number of primitives, we still cannot guarantee real-time framerates. In the following section we discuss how to select representative rotations so that we need only $r = O(m)$ while obtaining results similar to much higher r.

3.2 Rotation clusters

Linear blend skinning constrains the resulting deformations to a small linear subspace, where the motions of neighboring vertices are typically highly correlated. Therefore, it seems unnecessary to estimate local rotations at each edge set. Instead, we propose clustering vertices undergoing similar deformations. Instead, we cluster the vertices into $\mathcal{V}_1, \ldots, \mathcal{V}_r \subseteq \{1, \ldots, n\}$ so that their best-fit rotations can stand in for rotation of each edge set.

Linear blend skinning deformations are governed by vertex weights w_i and vertices with similar weights undergo similar rotations (regardless of spatial proximity). Therefore, we cluster vertices into r clusters based on their Euclidean distance in *weight space*, where for each vertex $\mathbf{v}_i$ we assign a vector of weights $(w_1(\mathbf{v}_i), \ldots, w_m(\mathbf{v}_i))$. In this representation rigid components comprised of vertices with identical weights collapse to a single point and performing k-means clustering achieves the desired grouping.

The number of clusters r can be used to trade off quality for speed. However, we observed that typically $r = O(m)$ is sufficient (we often simply set $r = 2m$), and higher numbers do not result in further visual improvements (see Figure 4). Interestingly, rotation clusters often improve deformation quality even with a unreduced formulation where each vertex position is an unknown. The clusters act as a regularization term that helps avoid singularities, though with small r the unreduced optimization suffers from lack of smoothness (see Figure 6b). Let us consider smoothness of discrete weight functions or deformation fields in an informal perceptual sense (as is often the case for manually painted weights) or in terms of convergence to a C^1 function [Jacobson et al. 2011]. Then, because our final deformation is dictated by linear blend skinning, it is always as smooth as the input skinning weights w_i. This smoothness is thus guaranteed at cluster boundaries and also near handle boundaries, where unreduced variational methods typically produce sharp discontinuities (see Figures 6 and 7).

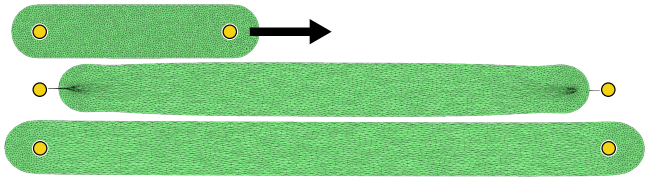

Figure 7: *The cigar shape (top) is stretched using unreduced per-triangle ARAP which introduces a $\mathcal{C}^1$ discontinuity at each point handle (middle). Our subspace is restricted to smooth deformations, so our result is also smooth (bottom).*

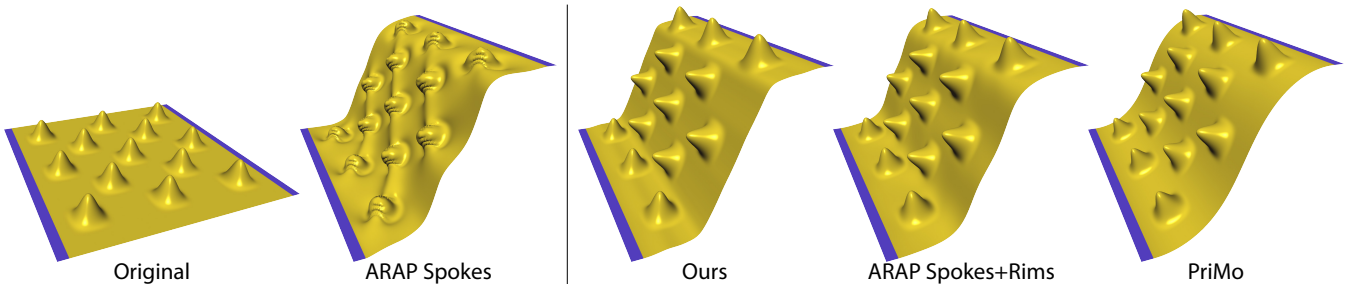

Figure 8: *Left: unreduced spokes-only ARAP energy [Sorkine and Alexa 2007] produces artifacts due to indefinite terms in the energy function. Right: our method utilizing the spokes and rims energy [Chao et al. 2010] fixes these issues and surprisingly accurately models behavior of bumpy rubber, similar to the unreduced solution and to behavior observed in silicone ice cube trays. Meanwhile, PriMo [Botsch et al. 2006] produces a fair wave corresponding to a thin metal sheet, but seems to ignore the bumps.*

3.3 Additional weight functions

Linearly blending handle transformations can be insufficient to create natural deformations. To enrich the space of attainable deformations, we can automatically create a set of abstract handles whose transformations are entirely determined by our optimization. For example, bending a cigar using two handles at its ends will tend to introduce a pinch (see Figure 6d). By generating additional weights functions along the cigar, we allow our optimization to find a more pleasing deformation (see Figure 6e).

In theory, an arbitrary weight function, appended as additional four columns of matrix $\mathbf{M}$, cannot increase the deformation energy because the original deformation subspace is contained in the new one. Still, constructing additional weights that keep the deformation fair is not straightforward. Simply adding additional bounded biharmonic weights [Jacobson et al. 2011] tends to introduce interpolation artifacts similar to Shepard interpolation (see the illustration in [Lewis et al. 2000]). Using low-frequency Laplacian eigenvectors is tempting, but their global support can hurt local control. Suitable additional weights should be smooth, localized, and preserve the nature of the original weights (especially if manually painted).

Partition of unity is usually required of skinning weights to ensure translation invariance. Due to our translation invariant energies, our method (somewhat surprisingly) requires only the original weights to partition unity, the additional weights can be arbitrary. This is because a global translation by vector $\mathbf{t} \in \mathbb{R}^d$ is in our skinning subspace (shifting all of the original transformations by $\mathbf{t}$ and keeping the additional constraints unchanged shifts all vertices by $\mathbf{t}$), and therefore if the user translates all constraints by $\mathbf{t}$, the resulting optimized transformations exactly reproduce this translation. This is a consequence of the fact the value of our energy functions (Eq. 4) remains unchanged when translating all vertices by $\mathbf{t}$.

Also, the scale of the additional weights does not matter because the optimization considers all linear combinations. However, we do require them to be zero at handles, i.e., where one of the original weights has value one. This way we ensure that our constraints are trivially feasible.

To preserve the deformation properties of the original weights, we propose embedding our mesh $\mathcal{M}$ in weight space, the same as when computing rotation clusters. To this end we create smooth isotropic cubic B-spline basis functions in weight space, centered at some seed locations $\mathbf{s}_1, \ldots, \mathbf{s}_p \in \mathbb{R}^m$. Let $\mathbf{e}_1, \ldots, \mathbf{e}_m \in \mathbb{R}^m$ be unit m-simplex vertices, $e_{i,j} = \delta_{i,j}$. To obtain a good distribution of the seed locations, we employ a discrete multi-dimensional version of optimized farthest points [Schlömer et al. 2011]. In each iteration, we choose a new location for seed $\mathbf{s}_i$:

$$\mathbf{s}_i = \operatorname*{argmax}_{\mathbf{x} \in \mathcal{M}} d_i(\mathbf{x}) \quad (16)$$

$$d_i(\mathbf{x}) = \min(\min_{j \neq i} \|\mathbf{x} - \mathbf{s}_j\|, \ \tfrac{1}{2} \min_k \|\mathbf{x} - \mathbf{e}_k\|) \quad (17)$$

i.e., moving $\mathbf{s}_i$ as far as possible from the other seeds and corners. We push $\mathbf{s}_i$ from the corners twice as far to facilitate the requirement of vanishing additional weights at all $\mathbf{e}_i$. Because Delaunay triangulation in higher dimensions is prohibitively expensive, we estimate the argmax in Eq. 16 by random sampling on $\mathcal{M}$ in weight space.

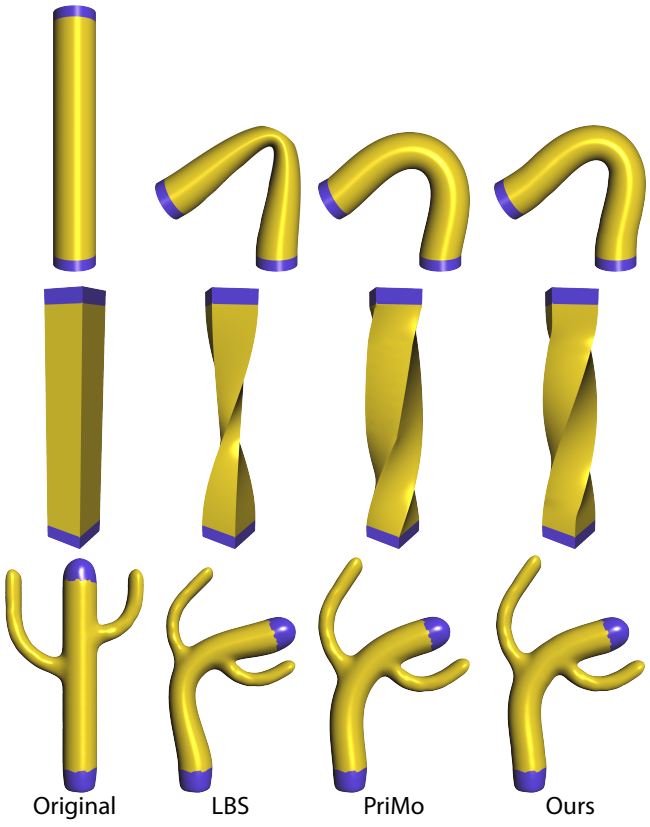

Figure 9: *Our method achieves visually equivalent quality as full nonlinear optimization (PriMo) [Botsch et al. 2006] and runs much faster.*

	Input model				Model reduction			Runtime	Precomputation		
Model	d	n	Triangles	Type	m_{orig}	m_{extra}	r	1 Iter.	Full	Switch	Add. Weights
Gingerbread man	2	2899	5543	Tri	9	0	18	11μs	0.227s	14ms	0s
Bubble man	2	6383	11804	Tri	5	0	10	7μs	0.266s	3ms	0s
Female	3	45659	91314	Sp+rim	17	0	34	19μs	3.359s	16ms	0s
Armadillo	3	47162	86482	Tet	17	0	34	20μs	8.365s	15ms	0s
Octopus	3	149666	299328	Tet	9	0	18	10μs	9.957s	9ms	0s
Cylinder	3	4802	9600	Sp+rim	2	30	64	40μs	0.471s	29ms	4.4s
Cactus	3	5261	10518	Sp+rim	2	30	64	40μs	0.586s	35ms	4.882s
Bar	3	6084	12106	Sp+rim	2	30	64	42μs	0.582s	32ms	4.266s
Ogre	3	28837	52306	Tet	5	15	40	22μs	2.717s	6ms	12.166s
Cross	3	29090	58176	Sp+rim	2	30	64	41μs	3.738s	49ms	10.09s
Bumpy plane	3	40401	80000	Sp+rim	2	30	64	41μs	4.328s	33ms	7.812s
Wiener dog	3	48187	85672	Tet	18	15	66	43μs	12.708s	77ms	11.123s

Table 1: *Model statistics and performance. d denotes the dimension, n the number of vertices,* Elements *refers to the type of elements used in the energy formulation (triangles* (Tri, Sp+rim) *or tetrahedra* Tet*);* Sp+rim *refers to spokes and rims energy. m_{orig} is the number of original weights and m_{extra} is the number of additional weights, r is the number of rotation clusters. We report the time for one optimization iteration (*1 Iter.*), the full precomputation time (*Full*), and the precomputation time when switching constraint types at handles (*Switch*).*

The algorithm is guaranteed to converge because the distance between two closest points cannot decrease. For the final $\mathbf{s}_i$, we define the additional weight function simply as $B(\|\mathbf{x} - \mathbf{s}_i\|/(2d_i(\mathbf{s}_i)))$, where $B(t) : [0,1] \to [0,1]$ is a cubic B-spline basis function.

This way of generating additional weights is only one of many possible, but we enjoy its simplicity and speed (variational methods such as Jacobson et al. [2011] are much slower). In our experiments, subspaces enriched with these additional weights support natural deformations with respect to our energy functions. Especially in situations with a small number of original weights we observe significant improvement of the deformation quality (see Figure 13).

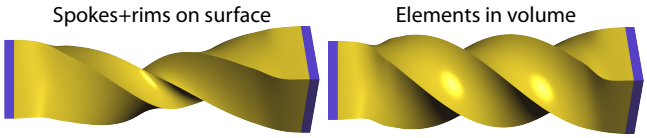

Figure 10: *Our reduced model preserves the nature of different energy functions: spokes and rims (left) behave like thin shells, while tetrahedra (right) correspond to volumetric elasticity.*

4 Results

We experimented with a number of previously discussed ARAP energies. For 2D images, we use the per-triangle energy [Liu et al. 2008]. For 3D shapes, the user's choice of energy controls the desired behavior (see Figure 10). The surface-based spokes and rims energy [Chao et al. 2010] treats models as thin shells while eliminating the artifacts of the spokes energy [Sorkine and Alexa 2007] (see Figure 8). Alternatively, the per-tetrahedron energy [Chao et al. 2010] treats shapes as enclosed volumes. Note that thanks to our reduced-order formulation, the tetrahedra participate in preprocessing only, where internal complexity is folded into our reduced system. At runtime our method considers only the surface.

To assess deformation quality, in Figure 9 we compare our results on the survey benchmark to those of Figure 10 in [Botsch and Sorkine 2008], with our skinning weights automatically generated using bounded biharmonic weights and enriched with additional weights per Section 3.3. As expected, the mesh quality of our reduced nonlinear method surpasses the quality of linear models. Meanwhile, the deformations are comparable to high quality nonlinear methods [Botsch et al. 2006] that are orders of magnitude slower to compute.

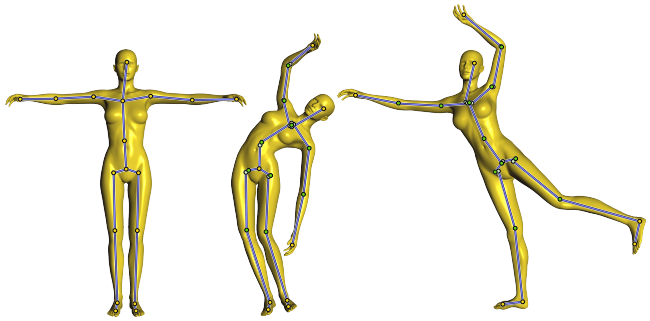

Figure 11: *Shape-aware IK: a female model animated solely by positional constraints at the endpoints of the head, hands and feet.*

The summary of our results can be found in Table 1. Our timings were obtained on a single-core CPU implementation on a laptop with a 2.5GHz Core i7-2860QM. We typically perform 15 iterations of the local-global optimization each at about 10μs to 40μs, resulting in total solve times of about 0.15 to 0.6 milliseconds. With our models, this performance is close to that of an LBS shader (with 4 weights per vertex, running in parallel on a GeForce 560M GPU). In addition to full precomputation times, we also report the time necessary to switch constraints or change their type (i.e. invert the system matrix in Eq. 15). In our figures and the accompanying video we use yellow dots for positional constraints (both at point handles or bone endpoints), green dots for unconstrained points, and red dots when specifying full transformations. Transformations at region handles (blue) are fully specified unless otherwise noted.

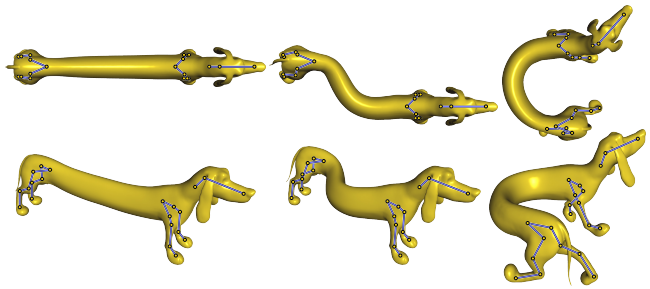

Figure 12: *Three disconnected skeletons control the* Wiener dog. *Additional abstract handle weights are generated along the belly allowing it to deform elastically as the user rearranges the skeletons.*

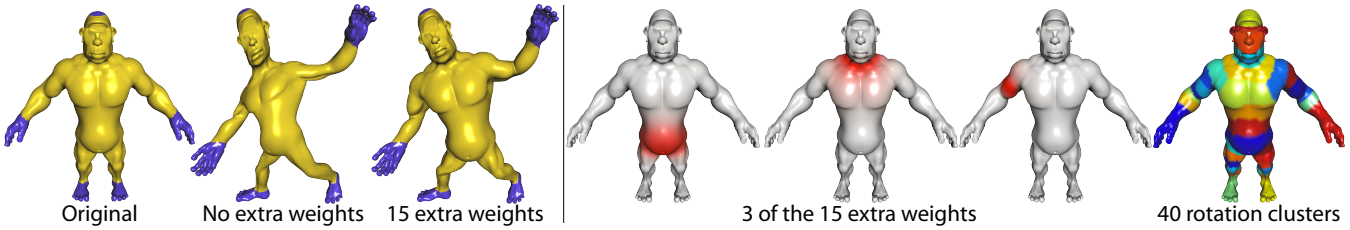

Figure 13: *With five fully constrained region handles, our method reduces to LBS with its volume loss issues. Enriching our subspace with 15 additional weight functions (shown in red) allows our method to find more natural deformations.*

In the realm of real-time animation, our approach provides shape-aware inverse kinematics supporting both classical (Figure 11) and disconnected (Figure 12) skeletons. To facilitate user control, our system also supports traditional hierarchical transformation chains (forward kinematics) even when only a few points are ultimately used as constraints in our optimization. The blue line segments drawn between joints denote this hierarchy and are not intended to visualize the affine transformations at joints, which may be partially or entirely optimized by our system. While most variational methods require constraints on each connected mesh component, our method can easily handle multiple connected components, implicitly "glued together" by manually painted skinning weights (see Figure 14).

Reduced variational models are typically associated with a higher energy value and a loss of quality due to the reduction of degrees of freedom. While our model reduction also increases the original energy value, projection into our deformation subspace has a nice regularization side-effect, forcing the solution to more visually pleasing deformations than the non-reduced method. For example, with $\mathcal{C}^1$ weight functions, we can guarantee $\mathcal{C}^1$ deformations, unlike non-reduced ARAP energy minimization (see Figure 7). Similarly, the reduction of the energy term using rotation clusters (Section 3.2) helps to avoid deformation field singularities occasionally produced by the non-reduced method (see Figure 6).

The utility of generating additional weights at abstract handles is illustrated in Figure 13. If all region handles are fully constrained, there is no room for optimization and our method reduces to LBS. However, after adding 15 additional weights, our technique results in more natural, shape-preserving deformations. Additional weight functions were also used in the benchmark examples (see Figure 9) and a cross-shape model (see Figure 15), designed to stress-test fairness of a branching structure's deformation.

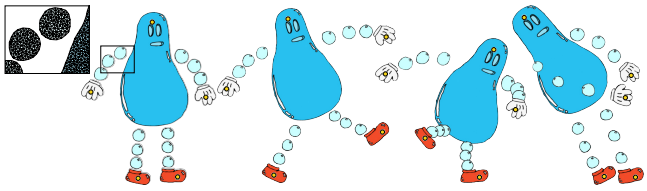

Figure 14: Bubble Man *with 17 connected components and only 5 positional handles. Our deformation subspace (LBS) is designed to prevent the components from falling apart.*

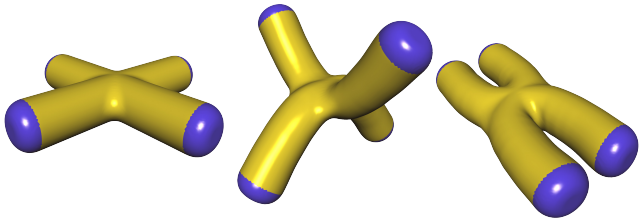

Figure 15: *A cross shape with four region controls is deformed in real-time producing a smooth, high quality shape*

5 Limitations and future work

To guarantee real-time performance, we use a fixed number of iterations in the local-global optimization and therefore, we cannot guarantee that our solution has converged completely. However, by initializing the optimization by the previous frame's transformations, we do not observe any disturbing artifacts even when using only 15 iterations (note that a reduced model typically requires much fewer iterations than an unreduced one). Inverse-kinematics methods often incorporate many other tunable parameters, such as balance and joint limit constraints [Shi et al. 2007]. Incorporating such terms is an obvious direction for improving the scope of our results.

The additional weights employed at abstract handles to bolster our optimization space are simple to construct, but may not be optimal for any given input mesh, linear blend skinning setup, and number of desired additional weights. With many overlapping original weights, our embedding in weight space causes our additional weights to overlap more. This in turn results in the practical disadvantage of having to support a large number of influencing handles in a shader implementation of linear blend skinning. Specifying the maximum number of influencing weights at any given point is an interesting constraint on our additional weight construction that we do not yet consider, even though previous weight reduction methods can be used as an immediate alternative [Landreneau and Schaefer 2010].

We justify clustering vertices in weight space because vertices with similar weights will have similar deformation gradients. This assumption holds for typical weight functions with small gradients, allowing us to use a small number of rotation groups. Robust and efficient handling of abruptly changing weights would be an interesting future direction.

6 Conclusion

Linear deformation methods are still sometimes preferred over nonlinear ones because of their speed and simplicity, resulting in compromises in quality [Botsch and Sorkine 2008]. By reducing both the space of possible deformations and simplifying the elastic energy function, our method avoids this compromise and delivers a method for deformations that are both fast and of high quality. We introduce new modes of control: shape-aware inverse kinematics combined with disconnected skeletons. Our method is simple to implement, with the time-critical inner loop only consisting of dense matrix multiplications and low-dimensional ($d \times d$) singular value decompositions. We believe our method will help to promote nonlinear deformation methods in applications where run-time efficiency is a primary concern.

Acknowledgements

We are grateful to Peter Schröder for an illuminating discussion, Emily Whiting for her narration of the accompanying video, Maurizio Nitti for the *Wiener Dog*, and Eftychios Sifakis for his open source fast 3×3 SVD code. We also thank Bob Sumner, Daniele Panozzo, Sebastian Martin and Bernd Bickel for their feedback. This work was supported in part by an SNF award 200021_137879 and by a gift from Adobe Systems.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5, 165:1–165:10.

AU, O. K.-C., TAI, C.-L., LIU, L., AND FU, H. 2006. Dual Laplacian editing for meshes. *IEEE Trans. Vis. Comput. Graphi. 12*, 3, 386–395.

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

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

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. 24*, 3, 982–990.

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

BLAIR, P. 1994. *Cartoon Animation*. Walter Foster Publishing, Inc., Irvine, CA, USA.

BOROSÁN, P., HOWARD, R., ZHANG, S., AND NEALEN, A. 2010. Hybrid mesh editing. In *Proc. EUROGRAPHICS, Short papers*, 41–44.

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

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

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

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

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

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4, 73:1–73:10.

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

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

FRÖHLICH, S., AND BOTSCH, M. 2011. Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8, 2246–2257.

GILLES, B., BOUSQUET, G., FAURE, F., AND PAI, D. 2011. Frame-based elastic models. *ACM Trans. Graph. 30*, 2, 15:1–15:12.

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.

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.-X., ADAMS, B., WICKE, M., AND GUIBAS, L. J. 2008. Non-rigid registration under isometric deformations. In *Proc. SGP*, 1449–1457.

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

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. 30*, 6, 165:1–165:8.

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.

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

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

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

KAVAN, L., COLLINS, S., AND O'SULLIVAN, C. 2009. Automatic linearization of nonlinear skinning. In *Proc. I3D*, 49–56.

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

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.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. ACM SIGGRAPH*, 165–172.

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

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. *Comput. Graph. Forum 27*, 5, 1495–1504.

MANSON, J., AND SCHAEFER, S. 2011. Hierarchical deformation of locally rigid meshes. *Comput. Graph. Forum 30*, 8, 2387–2396.

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.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3, 562–568.

PEKELNY, Y., AND GOTSMAN, C. 2008. Articulated object reconstruction and markerless motion capture from depth video. *Comput. Graph. Forum 27*, 2, 399–408.

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

SCHLÖMER, T., HECK, D., AND DEUSSEN, O. 2011. Farthest-point optimized point sets with maximized minimum distance. In *Proc. ACM SIGGRAPH Symposium on High Performance Graphics*, 135–142.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proc. ACM SIGGRAPH*, 151–160.

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

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

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3, 488–495.

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

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

WANG, R. Y., PULLI, K., AND POPOVIĆ, J. 2007. Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3, 73.

WAREHAM, R., AND LASENBY, J. 2008. Bone Glow: An improved method for the assignment of weights for mesh deformation. *Articulated Motion and Deformable Objects*, 63–71.

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.

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.