Linear Subspace Design for Real-Time Shape Deformation

Yu Wang[1] Alec Jacobson[2,3] Jernej Barbič[4] Ladislav Kavan[1]

[1]University of Pennsylvania [2]Columbia University [3]ETH Zurich [4]University of Southern California

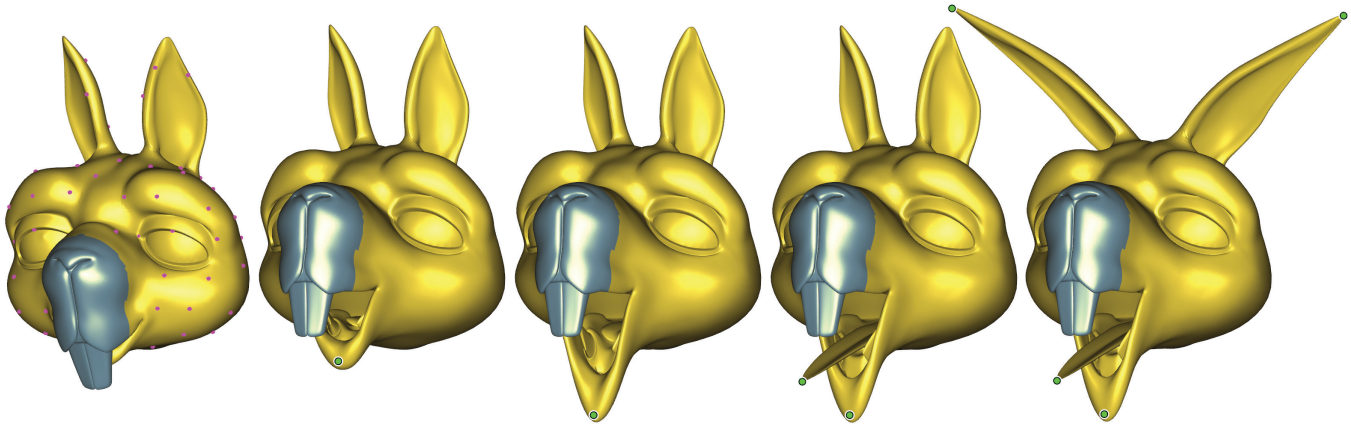

Figure 1: *Our linear subspaces are very fast to compute. This enables the users to add (or remove) control handles very quickly, allowing them to realize their creative intent in a single interactive session.*

Abstract

We propose a method to design linear deformation subspaces, unifying linear blend skinning and generalized barycentric coordinates. Deformation subspaces cut down the time complexity of variational shape deformation methods and physics-based animation (reduced-order physics). Our subspaces feature many desirable properties: interpolation, smoothness, shape-awareness, locality, and both constant and linear precision. We achieve these by minimizing a quadratic deformation energy, built via a discrete Laplacian inducing linear precision on the domain boundary. Our main advantage is speed: subspace bases are solutions to a sparse linear system, computed interactively even for generously tessellated domains. Users may seamlessly switch between applying transformations at handles and *editing the subspace* by adding, removing or relocating control handles. The combination of fast computation and good properties means that designing the *right* subspace is now just as creative as manipulating handles. This paradigm shift in handle-based deformation opens new opportunities to explore the space of shape deformations.

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

Keywords: Deformation modeling, skinning, subspace methods, reduced-order physics.

1 Introduction

Shape deformation brings life to animated characters and transforms existing geometric designs into novel variations. As shape complexities grow, so does the need for high-quality real-time shape deformation methods. Successful real-time deformation relies on building a subspace of admissible deformations, controlled by a small number of intuitive parameters. Previous attempts to design such subspaces achieve one or more desirable qualities, but no method achieves all of them.

Subspace bases derived from Euclidean-distances often boast simple, closed-form expressions, but do not respect the semantic information defined by the shape's boundary. A basis should adapt to the given input shape: it should be *shape aware* [Joshi et al. 2007]. Recent research demonstrates the effectiveness of a variety of control structures or *handles* used to parameterized real-time deformations. Isolated point constraints, rigid bones, selected regions and exterior cages each have tasks for which they excel and those for which they fail. A good subspace should unify all handle types or combination thereof [Jacobson et al. 2011]. Finally, a deformation subspace should reproduce simple *affine* transformations (i.e. translations, rotations and scales) if applied uniformly to all handles [Hildebrandt et al. 2011]. This ensures intuitive control via simpler position constraints rather than requiring explicit full linear transformations at each handle.

Superficially, our subspace design is similar to existing variational methods: we optimize a smoothness energy within the shape with appropriate boundary conditions to support a variety of handle types. However, we make an important observation that has drastic effects on the utility of our bases. This observation begins by acknowledging the property most responsible for the success of cage-based "generalized barycentric coordinates:" linear precision or, equivalently, rest-pose reproduction. Existing variational methods inherit the *lack* of linear precision of their quadratic smoothness energies, constructed with discrete Laplacians lacking linear precision along the shape boundary. Instead, our carefully constructed volumetric Laplacian retains linear precision on the boundary of the

domain, thus implying a linearly precise deformation subspaces, i.e., subspaces which contain the rest pose.

In this way, we *bake* linear precision into our energy, rather than enforce it via additional constraints or force the user to provide full linear transformations at all handles. This means subspace computation and re-computation is faster and subspace design is more flexible. The computational efficiency invites new user experiences and in general turns the subspace definition process into a creative task. New subspaces are computed rapidly enough for users to explore different handle placements interactively. The freedom to specify simple position constraints at points may be combined with the power of traditional transformation-based handle types such as rigid bones and selected regions. This combination enables new workflows for direct shape manipulation and more effective reduced domains for variational modeling (Figure 1) or physical simulations with dynamics (Figure 23).

2 Related work

The need for low-dimensional control of deformation fields was identified early in computer graphics. Among the first approaches was Free-Form Deformation [Sederberg and Parry 1986], which relied on regular lattices to specify spatial deformations. Cage-based deformation methods (see [Nieto and Susin 2013] for a recent survey) were an important step forward, because control polytopes offer much better adaptability to the input shape. The underlying theme of many cage-based methods is to generalize barycentric coordinates from simplices to general polytopes. Mean value coordinates for closed polyhedrons [Ju et al. 2005] offer many desirable properties and can be calculated using closed-form expressions, but are not fully shape-aware. This shortcoming has been addressed by harmonic coordinates [Joshi et al. 2007]. While many new intriguing coordinates and their underlying mathematical properties have been studied in recent years [Hormann and Sukumar 2008; Weber et al. 2011; Li and Hu 2013], a problem common to all cage-based method remains: the design of control cages requires experience with polygonal modeling. Any change, such as adding or deleting vertices, requires the user to update the cage connectivity.

While cage-based techniques are an important class of linear deformation methods, several other linear approaches have been explored both for surfaces [Sorkine et al. 2004; Botsch and Kobbelt 2004; Sorkine and Alexa 2007] as well as for volumetric (spatial) deformations [Bookstein 1989; Botsch and Kobbelt 2005]. Another popular linear deformation model is linear blend skinning [Magnenat-Thalmann et al. 1988], where the deformation degrees of freedom contain not only translations, but also local rotations. Jacobson [2013] showed that linear blend skinning is equivalent to higher-order barycentric coordinates [Langer and Seidel 2008]. With region controls only, our subspace becomes equivalent to Animation Space [Merry et al. 2006], but without the reliance on input training animations to derive the skinning weights.

Analogously to generalized barycentric coordinates, automatic methods for calculating linear blend skinning weights exist. The heat-diffusion method of [Baran and Popović 2007] pioneered the field of automatic skinning weights. The more recent bounded biharmonic weights (BBW) [Jacobson et al. 2011] feature superior smoothness and generalize to different control structures, including scattered control points. However, note that the underlying deformation model is linear blend skinning, which requires affine transformations even at control points. Without full transformations, point handles will produce unappealing results (see Figure 2). Also, the combination of smoothness, non-negativity, and interpolation of BBW can result in compromised *fairness* of the resulting deformations. Our weights avoid the fairness issues (Section 4) and

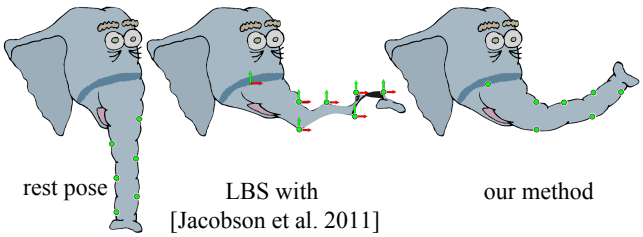

Figure 2: *Linear blend skinning requires full transformations specified even at isolated point handles. Deformations look awkward when only specifying translations.*

support both true control points (translation-only handles) as well as affine transformations. This allows us to combine the advantages of generalized barycentric coordinates and linear blend skinning in a single linear subspace.

However, linear deformation methods have certain inherent limitations [Botsch and Sorkine 2008] which lead to the development of non-linear techniques [Botsch et al. 2006; Sorkine and Alexa 2007; Botsch et al. 2007; Chao et al. 2010]. Non-linear methods guarantee high quality even for large deformations, but they are slow due to the underlying non-convex optimization problems. The complexity of non-linear variational methods can be dramatically reduced by restricting the available degrees of freedom to a carefully designed low-dimensional subspace. An early example of this important idea are variational harmonic maps [Ben-Chen et al. 2009] which employ a cage as a tool to generate suitable linear deformation subspaces. In linear blend skinning, the *necessity* of specifying local affine transformations for each control handle can be burdensome. This motivated Jacobson et al. [2012a] to develop a method to automatically compute a subset of the degrees of freedom of linear blend skinning by minimizing as-rigid-as-possible energies in a linear blend skinning subspace, assuming the linear blend skinning weights are given. In this paper we address precisely this assumption: we contribute a method to *design* a suitable linear subspace. To provide a complete deformation system, we also implement subspace as-rigid-as-possible deformations, but only as an application of our subspace.

Subspace design has been extensively studied in the context of physics-based simulation [Gilles et al. 2011; Faure et al. 2011]. Perhaps the simplest approach is to design a subspace for small deformations [James and Pai 2002; Hauser et al. 2003], leading to bases analogous to harmonic modes [Vallet and Lévy 2008]. Large deformations can be accommodated by applying Principal Component Analysis on pre-existing simulation data [Krysl et al. 2001], or using modal derivatives [Barbič and James 2005]. Modal derivatives have also been employed for shape design [Hildebrandt et al. 2011]. All of these methods create a global basis of deformations, whereas our subspace leads to local deformations, allowing us to control individual parts of the shape. Locality can be achieved via sparsity inducing norms [Neumann et al. 2013; Ozolinš et al. 2013]. The most recent method due to Zhang et al. [2014] achieves locality along with non-negativity and interpolation, however, their weights are very slow to compute and compromise smoothness. Our weights require only a solution of a relatively small linear system, which is many orders of magnitude faster. A special type of localized bases was presented by [Harmon and Zorin 2013], by using the Boussinesq static deformation response to indentation of a few vertices. In our work, we aim to design general-purpose bases which are linearly precise and can be used for large deformations of the input shape.

Our subspaces are derived by minimizing a quadratic fairness functional, analogously to thin-plate splines [Bookstein 1989] or more

general radial basis functions [Botsch and Kobbelt 2005]. In both cases, the deformation energies are minimized over the entire Euclidean space $\mathbb{R}^d$, which implies the resulting deformations are oblivious to the shape. The same limitation is inherent also to Voronoi-based methods [Bobach et al. 2006]. Instead, our method utilizes a quadratic deformation energy which respects the domain including its deep concavities. In one dimension, our method is equivalent to natural cubic splines (we elaborate on this in Section 3.2). Our method can be seen as generalization of the same idea to general shapes.

3 Method

We will define our basis for a possibly non-convex input subregion $\Omega \in \mathbb{R}^d$, discretized as a triangle mesh ($d = 2$) or tetrahedral mesh ($d = 3$). We store the vertices of the rest pose as rows of matrix $\overline{\mathbf{V}} \in \mathbb{R}^{n \times d}$, where n is the number of vertices. The deformed vertices will be denoted as $\mathbf{V} \in \mathbb{R}^{n \times d}$.

Control points and regions Our basis will interpolate isolated *control points* and affine transformations applied to selected *control regions*. Generic region handles are familiar to variational modeling [Botsch and Kobbelt 2004]. Regions also encompass rigidly deforming *bones* common in classical skinning. The combination of positional constraints at isolated points (i.e. pure translations) and full affine transformations at handles *generalizes* previous subspace formulations. Previous cage-based methods (e.g. [Huang et al. 2006; Ben-Chen et al. 2009]) could implicitly achieve this combination, but cages are difficult to create and the resulting subspaces are merely cage-aware, rather than truly shape-aware. With only region handles, our subspace resembles traditional skinning. With only point handles, our subspace corresponds to generalized barycentric coordinates (without the requirement that points form a cage). Our method allows users to combine the benefits of region and point handles in a single linear subspace.

Assuming the user selected m_p control points and r regions, we gather the rest positions of all mesh vertices incident on handles in the rows of $\overline{\mathbf{C}} \in \mathbb{R}^{m \times d}$, where $m = m_p + m_r$ includes m_p vertices at control points and m_r vertices found within the r regions. Typically, m is much smaller than n. Each of the m_p isolated control point contributes d dimensions to our subspace. A single affine transformation controls all the vertices of each region and contributes $(d + 1)d$ dimensions. The total dimension of our subspace will be $m_p d + r(d + 1)d$. We gather the target positions of the control points in the rows of $\mathbf{C}_1 \in \mathbb{R}^{m_p \times d}$ and we stack the transposed affine transformations of the regions in a matrix $\mathbf{H}_2 \in \mathbb{R}^{r(d+1) \times d}$. We can then write the target positions of vertices in the regions as $\mathbf{C}_2 = \mathbf{J}_2\mathbf{H}_2 \in \mathbb{R}^{m_r \times d}$, where $\mathbf{J}_2 \in \mathbb{R}^{m_r \times r(d+1)}$ is a rigid skinning matrix [Kavan et al. 2010]: a block matrix stacking homogeneous coordinates of rest positions for vertices in each region.

We can combine the influence of control point target positions and region transformations into one matrix equation:

$$\underbrace{\begin{pmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}}_{\mathbf{C}} = \underbrace{\begin{pmatrix} \mathbf{I}_{m_p} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2 \end{pmatrix}}_{\mathbf{J}} \underbrace{\begin{pmatrix} \mathbf{C}_1 \\ \mathbf{H}_2 \end{pmatrix}}_{\mathbf{H}} \tag{1}$$

where $\mathbf{I}_{m_p} \in \mathbb{R}^{m_p \times m_p}$ is an identity matrix. The resulting matrix $\mathbf{C} \in \mathbb{R}^{m \times d}$ contains *target positions* of all vertices at points or in regions. The matrix $\mathbf{H} \in \mathbb{R}^{h \times d}$ gathers all parameters spanning our subspace, where $h = m_p + r(d + 1)$.

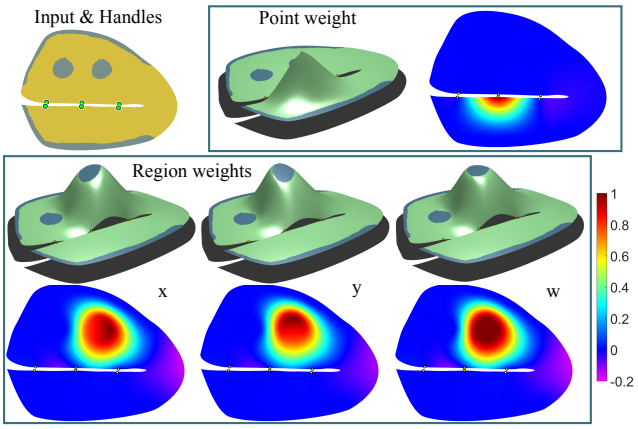

Figure 3: *Our region handles (blue) have vector-valued weights, in 2D corresponding to x,y, and w homogeneous coordinates (clockwise). Points have scalar-valued weights.*

Variational form Equation (1) provides a succinct way to express the subspace degrees of freedom. We now review how to derive a linear deformation subspace from the *variational form* of a quadratic energy minimization problem subject to constraints ensuring interpolation of point and region handles.

For now, let us consider a generic quadratic mesh deformation energy subject to satisfying constraints at vertices lying at control handles:

$$\mathbf{V} = \underset{\mathbf{X} \in \mathbb{R}^{n \times d}}{\operatorname{argmin}} \ \tfrac{1}{2}\operatorname{trace}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X}) \text{ subject to } \mathbf{S}\mathbf{X} = \mathbf{J}\mathbf{H}, \tag{2}$$

where $\mathbf{A}$ is *for now* a generic positive semi-definite quadratic form measuring the fairness of $\mathbf{X}$ (we propose a specific energy in 3.1), and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a binary selector matrix, i.e., matrix such that $\mathbf{S}\mathbf{X}$ selects m rows of $\mathbf{X}$ in the order implied by Equation (1). By definition, $\mathbf{A}$ has constant precision iff $\mathbf{A}\mathbf{1}_n = \mathbf{0}$, where $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ is a vector of ones, and linear precision iff $\mathbf{A}\overline{\mathbf{V}} = \mathbf{0}$.

Equation (2) specifies an equality constrained convex quadratic optimization problem, solved as a linear system via several standard methods. We choose direct substitution. Let $\mathbf{T} \in \mathbb{R}^{(n-m) \times n}$ be the complementary selector matrix of $\mathbf{S}$, i.e., corresponding to free vertices. This allows us to sort the variables of $\mathbf{X}$ into knowns $\mathbf{S}\mathbf{X}$ and unknowns $\mathbf{T}\mathbf{X}$. After rearranging terms, and setting the derivatives of Equation (2) with respect to $\mathbf{T}\mathbf{X}$ to zero, we have:

$$\mathbf{V} = \underbrace{\left(\mathbf{S}^\mathsf{T}\mathbf{J} - \mathbf{T}^\mathsf{T}(\mathbf{T}\mathbf{A}\mathbf{T}^\mathsf{T})^{-1}\mathbf{T}\mathbf{A}\mathbf{S}^\mathsf{T}\mathbf{J}\right)}_{\mathbf{W}}\mathbf{H} \tag{3}$$

The columns of $\mathbf{W}$ form the basis of our deformation subspace and we call them "weights", because they are analogous to weights used in linear skinning methods (for control regions) and generalized barycentric coordinates (for control points), see Figure 3. Provided that there exists at least one region or $d+1$ affine independent points, the matrix $\mathbf{T}\mathbf{A}\mathbf{T}^\mathsf{T} \in \mathbb{R}^{(n-m) \times (n-m)}$ is symmetric positive definite and therefore invertible. The main bottleneck consists in computing $(\mathbf{T}\mathbf{A}\mathbf{T}^\mathsf{T})^{-1}\mathbf{T}\mathbf{A}\mathbf{S}^\mathsf{T}\mathbf{J}$, i.e., solving a linear system with h right hand sides (because $\mathbf{T}\mathbf{A}\mathbf{S}^\mathsf{T}\mathbf{J}$ has h columns). We precompute its Cholesky factorization once and apply it to compute each column of $\mathbf{W}$ in parallel, which is very fast.

With only region controls, the equation $\mathbf{V} = \mathbf{W}\mathbf{H}$ resembles the matrix form of classical linear skinning. However, we note that our $\mathbf{W}$ is more general than linear blend skinning and is analogous

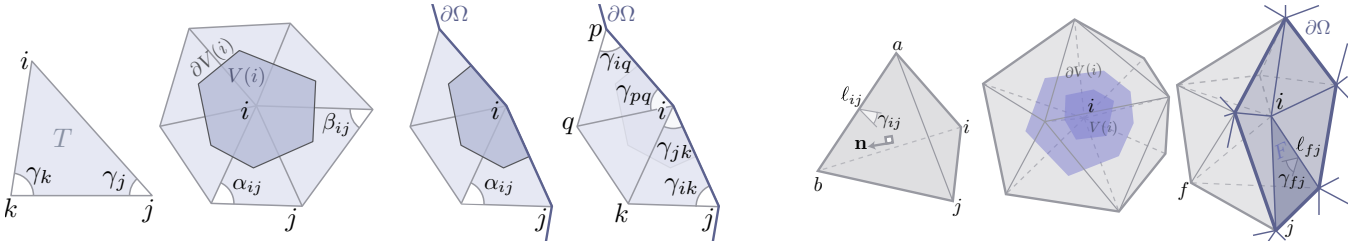

Figure 4: *Notation used in derivations.*

to the "Animation Space" method [Merry et al. 2006], related to "Multi-weight Enveloping" [Wang and Phillips 2002]. More details on different linear skinning methods can be found in [Jacobson et al. 2014].

Affine invariance The columns of $\mathbf{W}$ form a linear deformation subspace with several desirable properties. First, if $\mathbf{R} \in SO(d)$ is a rotation matrix, then $\mathbf{W}(\mathbf{HR}) = (\mathbf{WH})\mathbf{R}$, i.e., the resulting deformation is rotation invariant for arbitrarily large rotations of the handles. This property follows from our separable formulation in Equation (2), i.e., each column of $\mathbf{X}$ is independent. While this is a desirable property, we caution that the same equality holds for arbitrary $\mathbf{R} \in \mathbb{R}^{d\times d}$, i.e., the resulting deformation is also linearly invariant. This is an inherent limitation of linear deformations: Igarashi et al. [2005] proved that linear rotation invariant deformations must be also linearly invariant. In practice, it is often desirable to penalize non-rigid deformations; we achieve this by combining our subspace with minimization of advanced non-quadratic deformation energies (see Section 3.1).

Translation invariance imposes additional constraints on $\mathbf{W}$, or equivalently on $\mathbf{A}$ as we will see. Consider some arbitrary translation $\mathbf{t} \in \mathbb{R}^{1\times d}$ as a row vector. Fill the column vector $\mathbf{e} \in \mathbb{R}^{h\times 1}$ with ones for the first m_p entries followed by blocks of d zeros and one 1 for each of the r transformation handles. With this setup, $\mathbf{H} + \mathbf{t} \otimes \mathbf{e}$ corresponds to translating all of the handles by $\mathbf{t}$ ($\otimes$ is the Kronecker product). Let us examine the corresponding deformation within our subspace:

$$\mathbf{W}(\mathbf{H} + \mathbf{t} \otimes \mathbf{e}) = \mathbf{WH} + \mathbf{Wt} \otimes \mathbf{e} = \mathbf{WH} + \mathbf{t} \otimes \mathbf{We}$$

where the last equality follows from the properties of the Kronecker product. To complete the proof of translation invariance, it is sufficient to show that $\mathbf{We} = \mathbf{1}_n$. For an arbitrary quadratic energy $\mathbf{A}$, this equality will *not* hold, and translation invariance will be elusive. However, if we craft $\mathbf{A}$ to contain $\mathbf{1}_n$ in its null space the equality will hold and we will achieve translation invariance (see proof in Appendix A).

Exact reconstruction of the rest-pose from undeformed handles $\overline{\mathbf{H}}$ implies $\overline{\mathbf{V}} = \mathbf{W}\overline{\mathbf{H}}$. Again, this fundamental property is not true in general. We achieve it by crafting $\mathbf{A}$ such that the columns of $\overline{\mathbf{V}}$ are in the null space of $\mathbf{A}$. This important property called "linear precision" (or *rest-pose reproduction*) ensures that our deformation subspace contains the undeformed shape (Figure 5).

Figure 5: *Our method exactly reproduces the* Armadillo*'s rest pose, while the standard volumetric Laplacian (middle) and least-squares meshes [Sorkine and Cohen-Or 2004] (standard Laplace-Beltrami, right) do not.*

The key to obtaining rest-pose reproduction and translation invariance are our assumption on the null space of $\mathbf{A}$. Unfortunately, common quadratic energies used in previous work [Sorkine and Cohen-Or 2004; Jacobson et al. 2011] fail to satisfy our null space assumptions. We initially attempted to avoid the null space requirements by imposing additional constraints when computing our weights $\mathbf{W}$. Specifically, we explicitly imposed additional equality constraints $\overline{\mathbf{V}} = \mathbf{W}\overline{\mathbf{H}}$ [Zhang et al. 2014]. This approach produces desirable weights, but incurs significant computational penalty, because the constraint $\overline{\mathbf{V}} = \mathbf{W}\overline{\mathbf{H}}$ couples individual *columns* of $\mathbf{W}$. This means solving for all entries of $\mathbf{W}$ in a $nh \times nh$ linear system. Solving this linear system quickly becomes prohibitive even for moderate resolutions and small handle sets. Even on the meager 2D alligator example of 5945 triangles and 33 control points this global solve using MATLAB's backslah operator takes two minutes and 10 GB memory (compared to our 8 milliseconds and 2.7 MB). Fortunately, it is possible to design operators $\mathbf{A}$ which satisfy our null space requirements and therefore reduce the problem to solving an $(n-m)\times(n-m)$ sparse linear system with h right hand sides. This lead to dramatic improvements in computation time, allowing our method to scale to generous resolutions.

3.1 Linearly precise smoothness energy

Previous works have highlighted the advantages of minimizing the the squared Laplacian energy to encourage smoothness:

$$\mathbf{x}^\mathsf{T}\mathbf{L}^\mathsf{T}{\mathbf{M}^{-1}}^\mathsf{T}\mathbf{M}\mathbf{M}^{-1}\mathbf{L}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{L}^\mathsf{T}\mathbf{M}^{-1}\mathbf{L}\mathbf{x}, \tag{4}$$

where $\mathbf{L} \in \mathbb{R}^{n\times n}$ is a discrete Laplacian (typically the "cotangent Laplacian") and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a mass matrix (often a diagonal *lumped* matrix of barycentric or Voronoi volumes per vertex). Clearly, $\mathbf{x}^\mathsf{T}\mathbf{L}^\mathsf{T}\mathbf{M}^{-1}\mathbf{L}\mathbf{x} = 0$ for constant and linear $\mathbf{x}$ if and only if the discrete Laplacian $\mathbf{L}$ measures zero on such functions.

All reasonable definitions of discrete Laplacians achieve constant precision: $\mathbf{L}\mathbf{1}_n = 0$. The *ubiquitous* choice of $\mathbf{L}$ (see [Sorkine et al. 2004; Botsch and Kobbelt 2004; Vallet and Lévy 2008; Xu et al. 2009; Lipman et al. 2010; Finch et al. 2011; Kavan et al. 2011; Jacobson et al. 2011; Jacobson et al. 2012b] etc.) is linearly precise for interior vertices but fails on the boundary. For triangle meshes in $\mathbb{R}^2$, these methods define the contents of $\mathbf{L}$ as:

$$\mathbf{L}_{ij} = \begin{cases} \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right) & \text{if } \{ij\} \text{ is an interior edge,} \\ \frac{1}{2}\cot\alpha_{ij} & \text{if } \{ij\} \text{ is a boundary edge,} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where α_{ij} is the angle opposite the edge $\{ij\}$ and β_{ij} is the other angle for interior edges (see Figure 4 left).

Colloquially, this definition simply ignores the "missing cotangent" for boundary edges. More rigorously, if we consider a different energy—the squared gradient of $\mathbf{x}$—then this $\mathbf{L}$ appears (unsquared) as the quadratic form when imposing Neumann boundary conditions $\nabla x \cdot \mathbf{n} = 0$ on $\partial\Omega$ (e.g. [Vallet and Lévy 2008]). These boundary conditions are "natural" or "implicit" as they minimize the squared gradient energy [Desbrun et al. 2002].

Using this $\mathbf{L}$ as our discrete Laplacian in our squared Laplacian energy does have advantages: it is convergent, well behaved for reasonable meshes, positive semi-definite, and symmetric [Wardetzky et al. 2007]. Unfortunately, it is *not linearly precise* at boundary vertices. Effectively, the row $\mathbf{L}_i$ corresponding to a boundary vertex i measures the integrated Laplacian in the small region around vertex i *minus* the integrated normal derivative along the small portion of boundary near vertex i. The Laplacian portion is linearly precise, but the normal derivative is not.

To fix this, we add the normal derivative to all rows corresponding to boundary vertices and construct a linearly precise cotangent Laplacian:

$$\mathbf{K} = \mathbf{L} + \mathbf{N}, \tag{6}$$

where $\mathbf{N}$ measures a discrete normal derivative at boundary vertices:

$$\mathbf{N}_i\mathbf{x} = \frac{\cot\gamma_{ik}}{2}(\mathbf{x}_i - \mathbf{x}_k) + \frac{\cot\gamma_{jk}}{2}(\mathbf{x}_j - \mathbf{x}_k) + \tag{7}$$

$$\frac{\cot\gamma_{iq}}{2}(\mathbf{x}_i - \mathbf{x}_q) + \frac{\cot\gamma_{pq}}{2}(\mathbf{x}_p - \mathbf{x}_q) \tag{8}$$

where j and k are neighboring boundary vertices, p and q form their respective triangles with i, and γ_{ab} is the angle opposite the edge $\{ab\}$

The derivation for tetrahedral meshes in $\mathbb{R}^3$ is completely analogous. In this case, the discrete Laplacian $\mathbf{L}$ is defined in terms of cotangents of dihedral angles and edge lengths:

$$\mathbf{L}_i\mathbf{x} = \sum_{T\in N(i)} \sum_{j\neq i\in T} \frac{\ell_{ij}}{6} \cot\gamma_{ij}(\mathbf{x}_j - \mathbf{x}_j), \tag{9}$$

where ℓ_{ij} is the length of the edge *opposite* edge $\{i, j\}$ in tetrahedron T and γ_{ij} is the dihedral angle along that edge (see Figure 4 right). The normal derivative matrix $\mathbf{N}$ is similarly defined:

$$\mathbf{N}_i\mathbf{x} = \sum_{F\in N(i)\cap\partial\Omega} \sum_{j\in F} \frac{\ell_{fj}}{6} \cot\gamma_{fj}(\mathbf{x}_j - \mathbf{x}_f), \tag{10}$$

where F is a triangular boundary facet incident on i and f is the vertex opposite F.

Having defined $\mathbf{K}$ as our linearly precise Laplacian, we can now define a squared Laplacian smoothness energy with constant *and linear* functions in its null space:

$$\mathbf{x}^\mathsf{T} \underbrace{\mathbf{K}^\mathsf{T}\mathbf{M}^{-1}\mathbf{K}}_{\mathbf{A}} \mathbf{x} \tag{11}$$

The normal derivative matrix $\mathbf{N}$ is not symmetric so neither is $\mathbf{K}$, but our energy is positive semi-definite as $\mathbf{A}$ is symmetric.

Minimizing the continuous analog of this energy $\int_\Omega (\Delta x)^2\, dV$ corresponds to solving a biharmonic equation $\Delta^2 x = 0$ subject to second- and third-order boundary conditions: $\Delta x = \nabla\Delta x \cdot \mathbf{n} = 0$ on $\partial\Omega$ (derived via calculus of variations after applying Green's first identity twice). Viewed as an application of finite volume method, our discretization also captures these boundary conditions. Previously, Jacobson et al. [2010] discretized the same energy using piecewise linear elements in a mixed formulation and considered a variety of boundary conditions, but unfortunately not the combination required here. Higher-order elements should, in theory, support direct discretization of these boundary conditions. Using discrete exterior calculus, Fisher et al. [2007] employed a squared divergence energy for tangent vector field design. After their boundary modification and substituting their unknown vector field for the gradient of an unknown scalar field we obtain the same discrete operator $\mathbf{A}$.

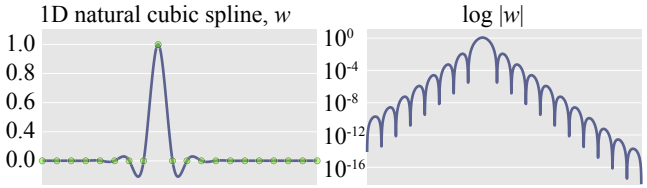

Figure 6: *We generalize natural cubic splines (left) to higher dimensions and non-Euclidean domains. In 1D, natural cubic splines have exponential decay in magnitude (right).*

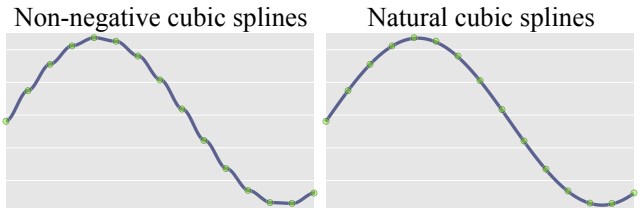

Figure 7: *Non-negative weights lead to ripples near interpolated values; negativity allows extrapolation for better fairness.*

3.2 Note about non-negativity

Our weights (columns of $\mathbf{W}$) are bounded, but they can be negative. Negative weights have been condemned in recent works [Lipman et al. 2007; Joshi et al. 2007; Hormann and Sukumar 2008; Jacobson et al. 2011; Jacobson et al. 2012b] because they can lead to counter-intuitive behavior, e.g., moving a control point to the right results in parts of the shape moving to the left. In this section, we highlight the fact that non-negative weights have certain important trade-offs, typically under-emphasized in previous work. Specifically, allowing negative weights is a necessary condition for achieving *fairness* and *extrapolation*.

The fairness issues are easiest to explain in $d = 1$. In one dimension, our method is equivalent to natural cubic splines: let us assume Ω is an interval with regularly spaced control points $x_1, \ldots, x_N \in \mathbb{R}$. In this didactic setting, we can obtain an interpolation basis corresponding to both natural cubic splines and our method by finding functions $f : [x_1, x_N] \to \mathbb{R}$ which minimize $\int (f'')^2$ subject to $f(x_i) = \delta_{ij}$, where $j = 1, \ldots, N$ is the index of our basis functions and δ_{ij} is the Kronecker delta. One such function is shown in Figure 6 (left). Note that the function is bounded (and in fact quickly converges to zero), but it is certainly not non-negative. This negativity is essential to obtaining fair interpolation curves. If we impose non-negativity constraints, we will obtain the one-dimensional analogue of Bounded Biharmonic Weights [Jacobson et al. 2011]. Unfortunately, the combination of smoothness, interpolation, and non-negativity forces the derivatives of the reconstructed functions to vanish at the control points. This results in ripples, exemplified in Figure 7. We note that both functions are C^1 continuous; in fact, they minimize the same energy ($\int (f'')^2$) but with respect to different constraints: the result in Figure 7 (left) appends non-negativity constraints. Similar effects can

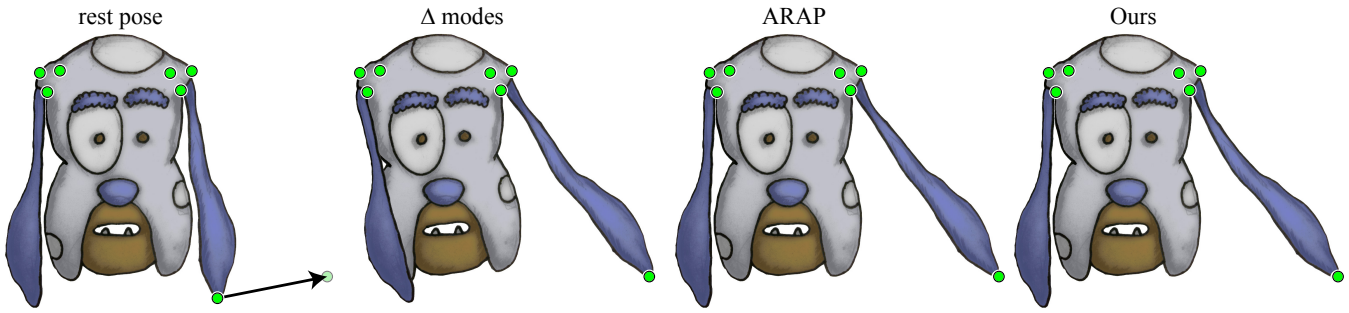

Figure 8: *Harmonic modes (top 100 eigenvectors of the standard Laplacian, augmented with the input's rest pose coordinates) induce a global deformation even if the guiding energy would be most satisfied with a local one. Our weights are local, so they do not hinder ARAP's locality.*

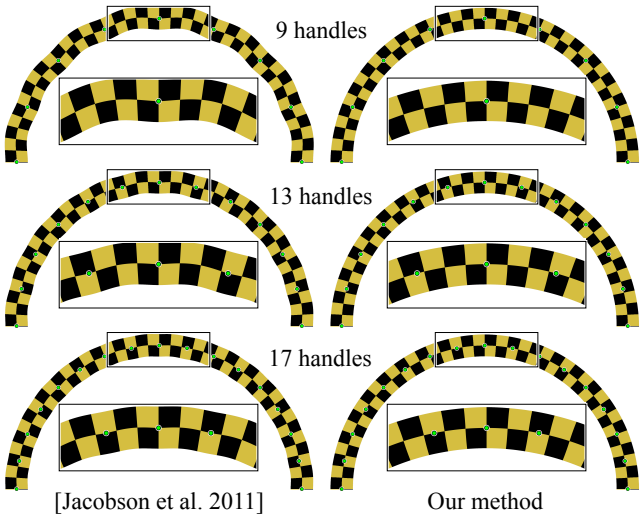

Figure 9: *Non-negative weights (such as [Jacobson et al. 2011], left) result in ripples when bending a 2D rectangle. These "fairness issues" shrink but remain with more handles. Our weights provide pleasingly fair deformations with any number of handles.*

be observed in higher dimensions, see Figure 9.

Putting quality of deformations aside, negative weights are also necessary to achieve extrapolation of control points. Extrapolation is an important asset because constraining the entire boundary or designing control cages may be impractical for more complex shapes. All reasonable definitions of generalized barycentric coordinates reduce to classical barycentric coordinates if the input shape is a simplex, i.e., convex set with $d+1$ control points. However, outside of the convex hull, barycentric coordinates are negative.

Finally, achieving non-negativity is often computationally expensive, e.g., Bounded Biharmonic Weights [Jacobson et al. 2011] require solutions of quadratic programs, which is orders of magnitude slower than our method and typically mandates off-line precomputation. Our weights can be computed much faster, allowing the user to explore different deformation subspaces *interactively*.

Smoothness Like all piecewise linear deformations, our smoothness is limited by the mesh resolution. In the limit of refinement our weights are C^1 near handles as is typical for biharmonic solutions [Jacobson et al. 2010], see Figure 3.

3.3 Domain reduction for non-linear optimization

Linear subspaces are ideal for reducing the computational complexity of non-linear optimizations [Barbič and James 2005; Hildebrandt et al. 2011; Jacobson et al. 2012a]. An ideal domain reduction untethers complexity from the discretization resolution without significantly reducing the quality of solutions.

In Section 4, we demonstrate the effectiveness of our particular subspace in two practical scenarios: 1) non-linear variational modeling for design tasks and 2) elastic physically based simulations with dynamics. We now briefly explain how we apply our linear subspace in each scenario.

As-rigid-as-possible deformation The so-called as-rigid-as-possible (ARAP) methods have earned popularity in geometric modeling due to their robust handling of large, localized rotations [Igarashi et al. 2005; Sorkine and Alexa 2007]. These methods minimize a local rigidity energy integrated over the domain. In the discrete setting, the energy is a sum over r local regions defined by *edge-sets* $\mathcal{E}_k$, $k = 1 \ldots r$:

$$\mathbf{V} = \underset{\mathbf{X}}{\operatorname{argmin}} \sum_{k=1}^{r} \min_{\mathbf{R}_k \in SO(d)} \sum_{\{i,j\} \in \mathcal{E}_k} \|(\mathbf{v}_i - \mathbf{v}_j) - \mathbf{R}_k(\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j)\|^2,$$

where $\mathbf{R}_k$ is thought of as the "best-fit" rotation matching the rest edge vectors $(\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j)$ of $\mathcal{E}_k$ to their deformations $(\mathbf{v}_i - \mathbf{v}_j)$. In $\mathbb{R}^2$, $\mathcal{E}_k$ is usually taken as the edges of the kth triangle [Igarashi et al. 2005; Liu et al. 2008]. In $\mathbb{R}^3$, $\mathcal{E}_k$ could be overlapping per-vertex patches defining a surface energy or could be the edges of a tetrahedron inside the shape defining a volumetric energy [Chao et al. 2010]. A popular method for optimizing this energy is the "local/global" or block coordinate descent approach: alternate between 1) fixing all rotations $\mathbf{R}_k$ and solving *globally* for positions $\mathbf{X}$ via a Poisson equation and 2) fixing all $\mathbf{X}$ and solve for each $\mathbf{R}_k$ via singular value decomposition.

Recently, linear subspaces brought the optimization of these ARAP energies to interactive levels [Hildebrandt et al. 2011; Jacobson et al. 2012a]. The first change is to substitute the full-resolution optimization with the linear subspace $\mathbf{X} = \mathbf{WH}$ [Hildebrandt et al. 2011]. This reduces the complexity of the global step. To achieve truly real-time rates, the local step is reduced by clustering rotations into meaningful super-edge-sets or "rotation clusters" [Müller et al. 2005; Jacobson et al. 2012a].

In contrast to previous subspaces, we argue that our proposed subspace is much better suited for domain-reduced ARAP minimizations. Modal analysis produces very effective subspaces in terms of energy value [Barbič and James 2005; Hildebrandt et al. 2011], but

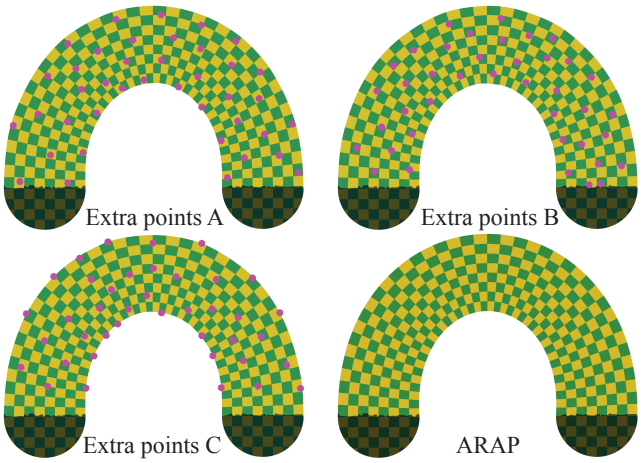

Figure 10: *Our results are not particularly sensitive to specific locations of auxiliary control points. The three different configurations (A,B,C) lead to similar results.*

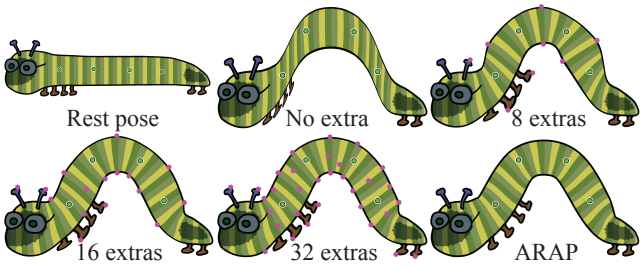

Figure 11: *Caterpillar deformed using two region and four point handles. Adding 8 extra points is not enough, 32 produces results similar to full-space ARAP.*

the global nature of the subspace weights is often undesirable. Even if the ARAP energy would be satisfied with local deformation, the global modes affect distant, unrelated parts of the shape (see Figure 8). Instead, our weights achieve locality due to their natural fall off when interpolating our control points (see Figure 16). The "additional weight functions" of [Jacobson et al. 2012a] are localized and fast to compute, but the underlying assumption of their method is that the user has brought as input a fixed set of linear blend skinning control handles. Presumably the skinning weights could be computed automatically, but state-of-the-art methods are too slow to recompute primary handle weights interactively [Jacobson et al. 2011]. The user is stuck with the input handle set. In contrast, our users may quickly add, remove, or edit handle constellations.

Elastic deformations with dynamics Many physics-based simulations rely on realistic looking deformations of elastic objects. Our linear subspace can be directly applied in any simulator which supports model reduction [Sin et al. 2013]. In our system we opted for a simpler yet effective approach: we augment the ARAP energy above with a momentum term [Chao et al. 2010; Martin et al. 2011; Jacobson 2013]. The implementation is straightforward because the extension amounts to adding a convex quadratic term to our energy.

Auxiliary control points To enrich the subspace we propose to place auxiliary control points throughout the domain. Unlike handles, these auxiliary control points are not intended to be directly controlled by the user. The user may view and adjust these points at any time, but we have found that it is most effective to hide the points once satisfied and interact only with a sparse set of handles.

Figure 12: *Auxiliary control points (purple) close to user-manipulated handles (green) are not a problem (middle figure). However, too close* handles *can lead to unpleasing deformations (right figure).*

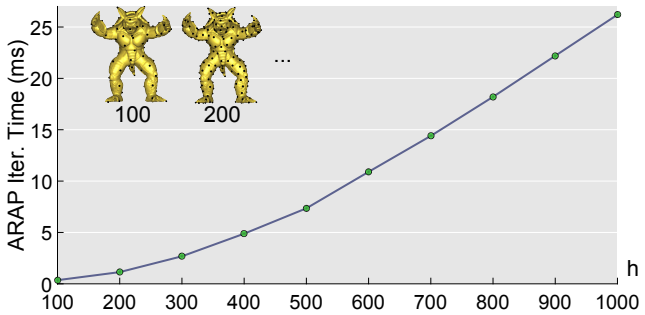

Figure 13: *Time per subspace ARAP iteration (in milliseconds) vs. the subspace dimension h.*

We use farthest point sampling, observing the results are not very sensitive to the exact locations of these points, see Figure 10. The number of auxiliary control points must be chosen judiciously by the user. Too few points do not produce appealing deformations; too many points lead to diminishing returns, see Figure 11. The proximity of auxiliary control points to user-controlled handles is not a problem because the ARAP energy will make the auxiliary control points move coherently with the handles. However, unpleasing deformations can occur if two handles are too close and are pulled apart by the user, see Figure 12.

4 Results

We conducted a number of experiments to verify the properties of our subspace and its effectiveness across a variety of models and applications. All of our 3D examples use tetrahedral meshes. We report computation times for the examples in this paper in Table 1, conducted on an Alienware laptop with an Intel i7 4800MQ CPU. Our basis computation involves a single linear system solve against h right hand sides, which we compute in parallel using CHOLMOD [Davis 2006]. In our experiments, CHOLMOD of SUITESPARSE is significantly better at parallelizing its factorization and backsubstitution than, for example, MATLAB's `backslash` or `chol` routines. The per-iteration time of subspace ARAP scales quadratically with the subspace dimension h, e.g., with the Armadillo model our system remains interactive until about $h = 1000$, see Figure 13. The local/global optimization converges typically within 5 - 30 iterations, but we already display intermediate results to the user.

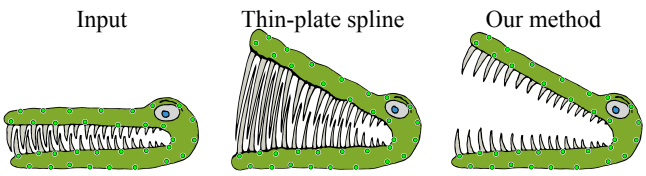

Figure 14: *Thin plate spline interpolation is not suitable for deforming non-convex shapes as it is not* shape-aware.

Model	d	n	#elem.	#points	#regions	h	$t_{\text{our}}^{SS}[s]$	$t_{\text{our}}^{MAT}[s]$	$t_{\text{bbw}}[s]$	$t_{\text{bbw}}/t_{\text{our}}^{SS}$	#clusters	iter [ms]
Alligator	2	3348	5945	33	0	33	0.000224	0.0014	0.151	672x	-	-
Octopus	2	4734	8778	100	10	130	0.000198	0.0012	0.187	944x	-	-
Bulldog-head	2	8602	16520	100	0	100	0.00056	0.004	0.54	965x	100	0.17
Woody-star	2	11290	22065	35	1	38	0.0013	0.006	0.731	542x	100	0.068
Elephant-2d	2	24928	49075	6	1	9	0.0094	0.0278	1.69	179x	-	-
Clam	2	27083	53335	6	4	18	0.005	0.0211	1.78	356x	-	-
Fatman	3	14266	52106	20	10	60	0.0015	0.0119	9.03	6018x	100	0.218
Cigar	3	17121	57125	2	2	10	0.0122	0.0458	32.21	2640x	20	0.024
Squirrel	3	41065	168117	100	2	108	0.0045	0.0591	176.1	39135x	100	0.374
Armadillo	3	47162	142685	120	2	128	0.0042	0.0472	9.94	2367x	100	0.44
Elephant	3	91637	396348	90	4	116	0.0102	0.1697	25.78	2527x	100	0.366

Table 1: *Performance statistics: dimension (d), number of vertices (n), number of elements (triangles/tets, #elem.), number of control points, control regions, dimension of our subspace (h), pre-compute time per weight with our method using SuiteSparse (t_{our}^{SS}) and MATLAB (t_{our}^{MAT}, both in seconds), pre-compute time per weight for BBW (t_{bbw}, in seconds), speedup of our method compared to BBW, number of ARAP rotation clusters, and per iteration time of the local/global solver (in milliseconds).*

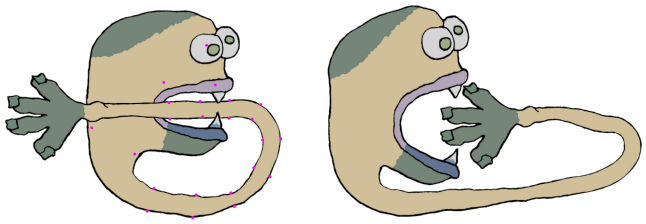

Figure 15: *Our intrinsic smoothness energy depends only on the metric of the domain, not its embedding. This rest mesh is self-overlapping (left). With our subspace we can* unfold *it smoothly.*

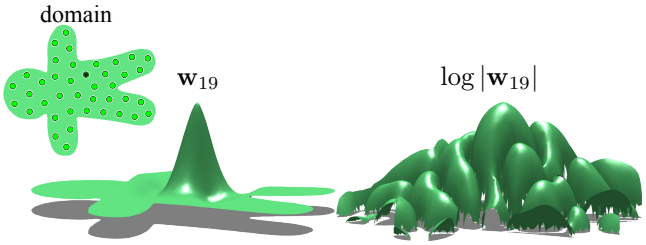

Figure 16: *A height field plot of the 19th weight shows its locality relative to the 39 other handles. The log-abs plot reveals that the weight magnitude decreases exponentially.*

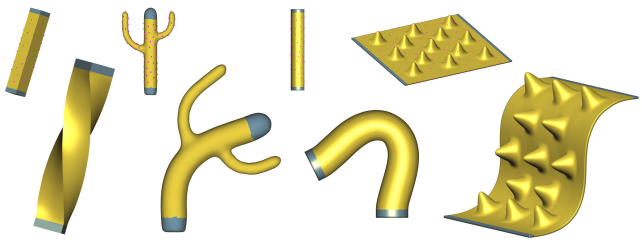

Figure 17: *Our system passes the standard benchmark of shape deformation methods (cf. [Botsch and Sorkine 2008]).*

Figure 18: *The user begins manipulating two points and two regions with the help of automatically positioned control points. Then the user adds handles interactively to stretch the ears and the tail.*

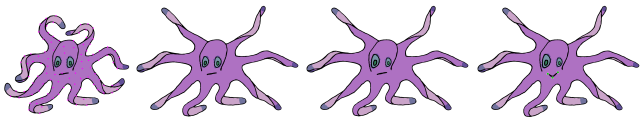

Figure 19: *After straightening the* Octopus*'s tentacles, scaling the eyes, don't forget to make him smile!*

As discussed in Section 3.2, our bases are an extension of natural cubic splines to irregular domains in $\mathbb{R}^2$ and $\mathbb{R}^3$. This *shape-awareness* is essential for maintaining intuitive control over non-convex shapes (Figure 14). The construction of our smoothness energy is purely *intrinsic* and will even retain shape-awareness for shapes with overlapping meshes [Sacht et al. 2013] (see Figure 15).

We do not explicitly ask for sparsity, yet we observe localized coordinates similar to 1D cubic spline basis functions (see Figure 6, right). Specifically, magnitudes decrease exponentially with each "ripple" between control points (see Figure 16). Even though not strictly sparse, for all practical purposes influence far away from a control point is effectively zero. This local behavior is analogous to Harmonic coordinates [Joshi et al. 2007] and heat diffusion weights [Baran and Popović 2007]. Unlike harmonic coordinates, we do not require an exterior cage, and our control points can be placed directly on or in the shape.

Combining our region handles with auxiliary control points guided automatically by an ARAP minimization, we reproduce the quality of nonlinear variational modeling techniques on the familiar benchmark shapes at real-time rates comparable to [Jacobson et al. 2012a] (see Figure 17 and cf. [Botsch and Sorkine 2008]). Jacobson et al. [2012a] assume skinning weights as input or presumably compute

Figure 20: *In a typical interactive sequence with our subspace, the user can freely add or adjust the control handle constellation without incurring long wait times while weights recompute. Designing the* right *subspace is now just as much a creative task as handle manipulation.*

bounded biharmonic weights [Jacobson et al. 2011], costing tens of seconds for handle. This deteriorates the interactive experience as the user must very carefully decide how many handles to place and where. In stark contrast, our computation finishes several orders of magnitude sooner. The user does not need to think twice about adding or editing the handle arrangement (see Figure 18).

Figure 21: *Placing a region over* Woody*'s star and delegating it to the ARAP minimization will maintain its shape without requiring the user to control its transformation manually.*

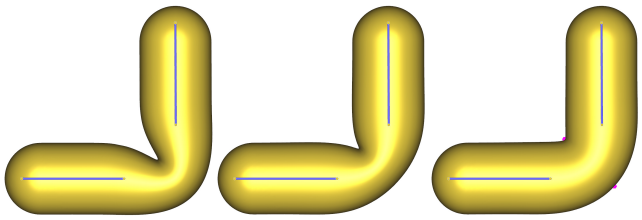

Figure 22: *Linear skinning methods suffer from joint collapse: linear blend skinning with BBW (left) and Animation Space (middle). Our subspace also suffers from linear artifacts, but these disappear after adding two auxiliary control points (right).*

Workflows are more natural when the subspace bases can be *designed* by the user interactively. The user adds point or region handles on an *as-needed* basis (see Figures 1 and 20). In both examples, automatically scattered auxiliary control points (shown in purple) improve the quality of deformations without excessively many degrees of freedom. This natural workflow also assists 2D cartoon editing (see Figure 19). Letting the ARAP minimization take control of region handles is useful to retain features of a shape without forcing the user to control them (see Figure 21).

Our region handles can be also used to define bones, achieving similar results as with traditional linear skinning methods. Without control points and ARAP energy minimization, our subspace is equivalent to Animation Space [Merry et al. 2006]. However, note that Merry et al. [2006] rely on input training data, whereas we calculate our weights solely using the rest pose. Linear skinning methods are prone to joint collapsing artifacts, see Figure 22. Using our auxiliary control points and subspace-ARAP, it is easy to deal with these artifacts; in our example, the problems disappear already after adding two strategically placed auxiliary control points, see Figure 22 (right).

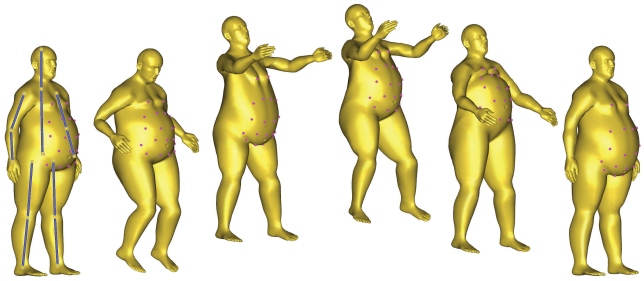

Figure 23: *The* Fat Man*'s belly jiggles due to inertia and gravity.*

Our subspaces allow us to seamlessly combine affine transformations with point displacements. We take advantage of this in our *Fat Man* example in Figure 23, where we combine bones (region handles) to drive the body and auxiliary control points to add flexibility to his belly. We drive the bone transformations using keyframe interpolation and we use our dynamic ARAP extension to produce inertia and gravity effects, resulting in realistic jiggling of the fat belly (see accompanying video).

5 Conclusions and Future Work

While our technique provides a very practical way of controlling deformations, there are several limitations and opportunities for future work. Our method requires discretization of the input domain. In 3D, tet-meshing has been notorious for its difficulties, however, recent tools enable us to create high-quality tet meshes even from imperfect polygon models [Jacobson et al. 2013; Xu and Barbič 2014]. In the future, we would like to explore generalizations of our method to simplicial meshes, combining features of different dimensions. In this paper, we focus on deformation fields, but we envision that similar methods could be used to interpolate arbitrary quantities defined at control points, not just deformations [Finch et al. 2011]. It will be interesting to apply advanced automatic control points sampling strategy to our method. For handle addition and removal, it is possible to combine our method with a fast biharmonic system updating technique, to further boost performance [Xu et al. 2009].

In summary, we presented a method to design quality deformation subspaces, suitable for variational shape deformation methods and dynamic simulation of elastic objects. The main advantage of our approach is speed: our subspaces are found as solutions of a sparse linear system with multiple right hand sides. Using modern linear solvers, the bases of our subspaces can be computed without any visible delay to the user, enabling new interactive experiences.

Acknowledgements

We would like to thank Keenan Crane, Mathieu Desbrun, Yotam Gingold, Eitan Grinspun, Maks Ovsjanikov, Eftychios Sifakis, Olga Sorkine-Hornung, Lifeng Zhu, and Denis Zorin for illuminating discussions. Harmony Li narrated the accompanying video. This research was sponsored by the National Science Foundation (IIS-1350330, CAREER-1055035, IIS-1422869) and the Sloan Foundation. The Columbia Computer Graphics Group is supported by the NSF, Intel, The Walt Disney Company, and Autodesk.

References

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. on Graph. 24*, 3 (Aug.), 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.

BOBACH, T., HERING-BERTRAM, M., AND UMLAUF, G. 2006. Comparison of voronoi based scattered data interpolation schemes. In *Proceedings of International Conference on Visualization, Imaging and Image Processing*, 342–349.

BOOKSTEIN, F. L. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence 11*, 6, 567–585.

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*.

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., 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 (July), 38:1–38:6.

DAVIS, T. A. 2006. CHOLMOD*: a sparse supernodal Cholesky factorization and modification package, version 3.0*. University of Florida.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. *Comput. Graph. Forum 21*, 3, 209–218.

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

FINCH, M., SNYDER, J., AND HOPPE, H. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. 30*, 6, 166:1–166:10.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph. 26*, 3.

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

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

HAUSER, K. K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformation using modal analysis with constraints. In *Proc. of Graphics Interface*, 247–256.

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.

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.

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

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. 32*, 4, 33:1–33:12.

JACOBSON, A., DENG, Z., KAVAN, L., AND LEWIS, J. 2014. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*.

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

JAMES, D. L., AND PAI, D. K. 2002. DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Trans. Graph. 21*, 3.

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.

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

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4, 93:1–93:9.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. for Numerical Methods in Engineering 51*.

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

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

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

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.

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

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based elastic materials. *ACM Trans. Graph. 30*, 4 (July), 72:1–72:8.

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

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. 24* (July), 471–478.

NEUMANN, T., VARANASI, K., WENGER, S., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Sparse localized deformation components. *ACM Trans. Graph. 32*, 6.

NIETO, J. R., AND SUSIN, A. 2013. Cage based deformations: a survey. In *Deformation Models*. Springer, 75–99.

OZOLINŠ, V., LAI, R., CAFLISCH, R., AND OSHER, S. 2013. Compressed plane waves-compactly supported multiresolution basis for the laplace operator. *Proc. of NAS*.

SACHT, L., JACOBSON, A., PANOZZO, D., SCHÜLLER, C., AND SORKINE-HORNUNG, O. 2013. Consistent volumetric discretizations inside self-intersecting surfaces. vol. 32, 147–156.

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

SIN, F. S., SCHROEDER, D., AND BARBIČ, J. 2013. Vega: Nonlinear fem deformable object simulator. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 36–48.

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

SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Proc. SMI*, 191–199.

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

VALLET, B., AND LÉVY, B. 2008. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 251–260.

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

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

XU, K., ZHANG, H., COHEN-OR, D., AND XIONG, Y. 2009. Dynamic harmonic fields for surface processing. *Computers & Graphics 33*, 3, 391–398.

ZHANG, J., DENG, B., LIU, Z., PATANÈ, G., BOUAZIZ, S., HORMANN, K., AND LIU, L. 2014. Local barycentric coordinates. *ACM Trans. Graph. 33*.

Appendix A: Properties of the solution operator

We prove that $\mathbf{We} = \mathbf{1}_n$, assuming that $\mathbf{W}$ is a solution operator corresponding to a positive semi-definite energy $\mathbf{A}$ which has $\mathbf{1}_n$ in its nullspace. An elegant way to see this is via the variational formulation, noting that $\mathbf{We}$ can be interpreted as:

$$\mathbf{We} = \underset{\mathbf{x}\in\mathbb{R}^{n\times 1}}{\operatorname{argmin}} \operatorname{trace}(\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}) \text{ subject to } \mathbf{S}\mathbf{x} = \mathbf{J}\mathbf{e} \qquad (12)$$

Note that due to the special structure of our extended skinning matrix $\mathbf{J}$, we have $\mathbf{Je} = \mathbf{1}_m$. However, the vector $\mathbf{1}_n$ leads to zero energy, because $\mathbf{A1}_n = \mathbf{0}$ and it clearly satisfies all of the constraints because $\mathbf{S1}_n = \mathbf{1}_m$. This means that $\mathbf{1}_n$ is a solution of the optimization problem from Equation (12). Assuming that we have at least one region or $d+1$ affine independent control points, the solution is unique and therefore, $\mathbf{We} = \mathbf{1}_n$.

A similar argument reveals why $\mathbf{W}\overline{\mathbf{H}} = \overline{\mathbf{V}}$, where $\overline{\mathbf{H}}$ corresponds to undeformed configuration. Specifically, the first m_p rows of $\overline{\mathbf{H}}$ contain the rest-pose positions of the vertices, followed by r blocks of $(d+1)\times d$ matrices of identity transformations (i.e. a $d\times d$ identity matrix followed by a row of zeros). This means the matrix $\mathbf{J}$ retrieves the rest-pose positions of the constrained vertices (corresponding to both point and region handles), formally: $\mathbf{J}\overline{\mathbf{H}} = \mathbf{S}\overline{\mathbf{V}}$. This means that $\mathbf{W}\overline{\mathbf{H}}$ is the following minimizer:

$$\mathbf{W}\overline{\mathbf{H}} = \underset{\mathbf{X}\in\mathbb{R}^{n\times d}}{\operatorname{argmin}} \operatorname{trace}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X}) \text{ subject to } \mathbf{S}\mathbf{X} = \mathbf{S}\overline{\mathbf{V}} \qquad (13)$$

Because we assume the columns of $\overline{\mathbf{V}}$ are in the nullspace of $\mathbf{A}$, we can see that $\overline{\mathbf{V}}$ is the solution of the optimization problem from Equation (13) and, as above, we can conclude that $\mathbf{W}\overline{\mathbf{H}} = \overline{\mathbf{V}}$.