Analytic Eigensystems for Isotropic Distortion Energies

BREANNAN SMITH, FERNANDO DE GOES, and THEODORE KIM, Pixar Animation Studios

Fig. 1. We present closed-form expressions for the eigensystems of isotropic distortion energies suited to geometry processing and physical simulation. We use the analytic eigensystem of the Symmetric Dirichlet energy to optimize a 2D parameterization (left). Our analytic eigensystem for the ARAP energy was used to simulate the over-inflated tires in this scene from *Cars 3* (right). ©Disney/Pixar

Many strategies exist for optimizing non-linear distortion energies in geometry and physics applications, but devising an approach that achieves the convergence promised by Newton-type methods remains challenging. In order to guarantee the positive semi-definiteness required by these methods, a numerical eigendecomposition or approximate regularization is usually needed. In this article, we present analytic expressions for the eigensystems at each quadrature point of a wide range of isotropic distortion energies. These systems can then be used to project energy Hessians to positive semi-definiteness analytically. Unlike previous attempts, our formulation provides compact expressions that are valid both in 2D and 3D, and does not introduce spurious degeneracies. At its core, our approach utilizes the invariants of the stretch tensor that arises from the polar decomposition of the deformation gradient. We provide closed-form expressions for the eigensystems for all these invariants, and use them to systematically derive the eigensystems of any isotropic energy. Our results are suitable for geometry optimization over flat surfaces or volumes, and agnostic to both the choice of discretization and basis function. To demonstrate the efficiency of our approach, we include comparisons against existing methods on common graphics tasks such as surface parameterization and volume deformation.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**;

Additional Key Words and Phrases: Second-order methods, geometry optimization, distortion energy

ACM Reference format:
Breannan Smith, Fernando De Goes, and Theodore Kim. 2019. Analytic Eigensystems for Isotropic Distortion Energies. *ACM Trans. Graph.* 38, 1, Article 3 (February 2019), 15 pages.
https://doi.org/10.1145/3241041

Authors' address: B. Smith, F. De Goes, and T. Kim, Pixar Animation Studios, 1200 Park Ave. Emeryville, CA 94608; emails: {breannan, fernando, tkim}@pixar.com.

ACM 0730-0301/2019/02-ART3 $15.00
https://doi.org/10.1145/3241041

1 INTRODUCTION

The minimization of distortion energies is at the core of many graphics applications, such as surface parameterization and physically-based deformation. Unfortunately, most energies are non-convex, which make them challenging to optimize numerically. In particular, the classic second-order Newton algorithm (see, e.g., Nocedal and Wright (2006)) tends to stall and even diverge in the presence of an indefinite energy Hessian. To overcome this issue, many strategies have been developed for modifying the Hessian matrix using a variety of convex proxies. For instance, some techniques replace the Hessian with a Laplacian-like preconditioning matrix at the cost of degrading convergence to first-order (Claici et al. 2017; Kovalsky et al. 2016). Other methods alter the Hessian by removing negative eigenvalues numerically (Stomakhin et al. 2012; Teran et al. 2005) or via 2D composite majorization (Shtengel et al. 2017).

In this article, we resolve Hessian indefiniteness by investigating eigenstructures analytically. A similar approach was considered previously by Chen and Weber (2017) for the restricted case of 2D isotropic energies, but their formulation is based on complex analysis and thus does not generalize to 3D. Conversely, some attempts have sacrificed generality to derive fast Hessian projections for specific energies. In particular, McAdams et al. (2011) obtained analytic eigenvalues for Co-rotational elasticity, while Smith et al. (2018) addressed a custom Neo-Hookean energy. Our method automatically produces these results as special cases, but goes further and establishes a general method for revealing compact eigensystems for a variety of isotropic distortion energies in both 2D and 3D.

Our key contribution is a list of analytic expressions that can be used to efficiently project the Hessian of isotropic distortion energies to positive semi-definiteness. These expressions can be easily incorporated into a projected Newton solver, thus providing second-order convergence with existing distortion optimization techniques. To achieve this, we depart from previous methods by addressing isotropic energies using the invariants of the stretch tensor that arises from the polar decomposition of the

deformation gradient. We present closed-form expressions for the eigenvalues and eigenvectors of all these tensor invariants, and use these results to arrive at a systematic scheme for constructing the eigenstructure of the Hessians for any isotropic energy. Our approach leads to analytic expressions for ⅔ of the eigenpairs of any isotropic distortion energy in 3D, and for ½ of the eigenpairs in 2D. We also show that these closed-form eigenvectors are invariant to the distortion model both in 2D and 3D. The remaining eigenvalues and eigenvectors can often be found analytically or, in the worst case, by solving a small $d \times d$ eigenproblem ($d = 2$ or 3). We demonstrate the effectiveness of our method by providing fully analytic expressions for the eigensystems of many popular distortion energies, including As-Rigid-As-Possible (ARAP), Symmetric ARAP, Co-rotational, Symmetric Dirichlet, and Most Isometric Parameterization (MIPS).

2 RELATED WORK

Many graphics applications are based on the optimization of isotropic distortion energies, so, for brevity, we will focus on previous methods that have addressed the numerics of these optimizations.

2.1 First-Order Methods

The local-global algorithm is arguably the most popular optimization technique in geometry processing. It consists of an expectation-maximization scheme that alternates local and global steps. Alexa et al. (2000), for instance, employed this method to minimize the ARAP energy in 2D. The local step projects each deformation gradient to a rotation, while the global step uses a Laplacian to fuse the rotated elements using a linear solve. This approach has been applied to many applications, such as surface modeling (Sorkine and Alexa 2007), mesh parameterization (Liu et al. 2008), and volume deformation (Jacobson et al. 2012).

Bouaziz et al. (2012) extended the local-global method to a broad class of distortion energies using proximal operators. The Laplacian in the global step was replaced by the Jacobian matrix of the proximal operators, which resembles the Hessian approximation of a Gauss-Newton method, while element-wise local steps were retained. This technique was later employed in physical simulation (Bouaziz et al. 2014) and improved using the Alternating Direction Method of Multipliers (ADMM) (Narain et al. 2016), Chebyshev extrapolation (Wang 2015), and Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) updates (Liu et al. 2017).

Other distortion energies were addressed using block coordinate descent methods (Fu et al. 2015; Horman and Greiner 1999) and Gauss-Newton solvers (Eigensatz and Pauly 2009). More recently, Kovalsky et al. (2016) showed that the Laplacian matrix is a reliable quadratic proxy for many distortion energies, while Claici et al. (2017) advocated the discrete Killing operator as an isometry-aware proxy. Alternatively, Rabinovich et al. (2017) proposed an iterative reweighting scheme of the Laplacian based on gradient residuals. Pighin and Lewis (2007) instead used an iterative reweighted least-squares technique, while Bommes et al. (2009) adopted a greedy progressive stiffening of the Laplacian.

All of these techniques are solely contingent on the gradient of the distortion energy, and are, therefore, inherently first-order. In contrast, our work investigates the use of second-order methods.

2.2 Second-Order Methods

Second-order methods are preferred in numerical optimization due to their superior convergence. However, special care must be taken in order to handle indefinite Hessians; otherwise, the computations can stall or diverge. For example, Chao et al. (2010) used a trust-region solver for the optimization of ARAP and Co-rotational energies, which is essentially an adaptive regularization of the Hessian. Shtengel et al. (2017) advocated a convex-concave decomposition of the distortion energies combined with a convex majorizer solver. However, finding an effective convex-concave decomposition is an opaque process, and the approach of Shtengel et al. (2017) is limited to 2D. We instead present a systematic, second-order approach for distortion energies in both 2D and 3D.

An alternative is to use a projected Newton technique, which projects the Hessian to a positive semi-definite matrix by clamping its negative eigenvalues at every solver iteration. Since extracting the eigenvalues and eigenvectors of the global Hessian is computationally prohibitive, several authors have projected the eigensystem of a local Hessian evaluated at quadrature points (see, e.g., Fu and Liu (2016)). Consequently, the eigendecomposition of each element-wise matrix becomes the computational bottleneck. Our results accelerate this stage by presenting closed-form expressions for the eigenvalues and eigenvectors.

By viewing the distortion energies in terms of the invariants of the 3D Cauchy-Green strain tensor, the work of Teran et al. (2005) was able to probe the indefiniteness of the Hessian using one 3×3 and three 2×2 eigenproblems. While this approach reduces the problem size, it does not yield the closed-form expressions for the underlying eigenvalues and eigenvectors, so the construction of the projected Hessian is performed numerically. The use of tensor invariants with respect to the Cauchy-Green strain tensor is also limiting because it is insufficient to express stretch-based energies such as ARAP or the Co-rotational model. This limitation was addressed in Stomakhin et al. (2012) by replacing the Cauchy-Green invariants with the singular values of the deformation gradient, but the inefficient numerical eigensolves for Hessian blocks remain. Xu et al. (2015) improved these computations further by restricting the analysis to energies that satisfy the Valanis-Landel hypothesis. Unfortunately, many geometric energies, such as MIPS, Symmetric Dirchlet, and Symmetric ARAP do not fall into this category. Our results, instead, lead to faster, more compact, and more general code.

The work of McAdams et al. (2011) presented an analytic solution for the indefiniteness of the Co-rotational energy. To do so, they decomposed the 4th -order tensor defined by the energy Hessian into symmetric and skew-symmetric parts. This yields the eigenstructure of the Co-rotational model, albeit embedded inside 4th -order tensors. We propose a more general derivation that produces their eigenvalue expressions as a special case and, additionally, reveals the structure of the underlying eigenvectors.

Chen and Weber (2017) considered analytic eigensystems for 2D isotropic energies. However, their derivation relies heavily on complex derivatives and, thus, does not extend to 3D. Their generic expressions also contain degeneracies that invalidate the eigenanalysis for a variety of energies and deformation configurations. For example, at uniform scaling configurations such as the rest

state, their formulas return a null vector as an eigenvector (see ğ1.3 in the supplemental material). In sharp contrast, our formulation is well-defined for any deformation gradient configuration and for any isotropic energy.

More recently, Smith et al. (2018) took an energy-specific approach and resolved the indefiniteness of a custom Neo-Hookean model. In doing so, they derived the analytic eigensystems for the Frobenius squared norm and the determinant of the deformation gradient in 3D. However, their approach lacks generality as they did not derive the eigensystem of a third invariant (I_1 in Section 4.1) that is not present in the Neo-Hookean model. This additional term appears in almost all geometric distortion energies, and precludes the use of their results in this context. In contrast, our formulation handles this term in closed-form. By addressing this invariant, we also provide the eigenstructure for the *rotation gradient*, a term that has been known to introduce numerical difficulties in the past (Twigg and Kačić-Alesić 2010), and is often computed by performing a set of matrix inverses (Barbič and Zhao 2011). Our analysis reveals that there is, in fact, a simple, direct expression for this term that bypasses these difficulties. With the complete set of eigenstructures for all the tensor invariants in hand, we are then able to construct closed-form eigensystems for isotropic energies in 2D and 3D.

3 DEFINITIONS

We begin by establishing several basic definitions and concepts. Scalars will be denoted with unbolded lowercase (a), vectors as bolded lowercase ($\mathbf{a}$), matrices (a.k.a., 2nd-order tensors) as bolded uppercase ($\mathbf{A}$), and 4th-order tensors using blackboard bold ($\mathbb{A}$).

Tensor Notation: We will be analyzing many 2nd- and 4th-order tensors, so we adopt a notation similar to Golub and Van Loan (2012) that flattens 2nd- and 4th-order tensors to vectors and matrices respectively. Concretely, given a 2×2 matrix $\mathbf{A}$, we define the vectorization operator "vec" as the column-wise flattening:

$$\mathbf{A} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \Leftrightarrow \operatorname{vec}(\mathbf{A}) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \mathbf{a}. \tag{1}$$

We can arrange a 4th-order tensor as a matrix of matrices:

$$\mathbb{A} = \begin{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} & \begin{bmatrix} i & k \\ j & l \end{bmatrix} \\ \begin{bmatrix} e & g \\ f & h \end{bmatrix} & \begin{bmatrix} m & o \\ n & p \end{bmatrix} \end{bmatrix} = \begin{bmatrix} [\mathbf{A}_{11}] & [\mathbf{A}_{12}] \\ [\mathbf{A}_{21}] & [\mathbf{A}_{22}] \end{bmatrix}, \tag{2}$$

and define its vectorization operator "vec" as a double unfolding that yields the matrix:

$$\begin{aligned} \operatorname{vec}(\mathbb{A}) &= [\,\operatorname{vec}(\mathbf{A}_{11}) \big| \operatorname{vec}(\mathbf{A}_{21}) \big| \operatorname{vec}(\mathbf{A}_{12}) \big| \operatorname{vec}(\mathbf{A}_{22})\,] \\ &= \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix} = \mathbf{A}. \end{aligned} \tag{3}$$

This flattening convention has several properties that we leverage in our derivations. The double contraction of two 2nd-order tensors becomes a dot product, i.e., $\mathbf{A} : \mathbf{B} = \operatorname{tr}(\mathbf{A}^\top \mathbf{B}) = \mathbf{a}^\top \mathbf{b}$, and the eigenvector of a flattened matrix $\mathbf{B}\,\mathbf{a} = \lambda\,\mathbf{a}$ is equivalent to the "eigenmatrix" of the corresponding higher-order tensor, i.e., $\mathbb{B} : \mathbf{A} = \lambda\,\mathbf{A}$.

Discretization: We consider the distortion optimization of a d-dimensional mesh $\mathcal{M}$, where $d=2$ or 3. We use $\bar{\mathbf{x}}$ to denote the vertex positions of $\mathcal{M}$ in the rest pose, and $\mathbf{x}$ to indicate the new positions computed by the optimization. These positions can be interpolated using any choice of basis functions associated with quadrature points $\{q\}$ distributed over $\mathcal{M}$. For instance, the typical case of simplicial meshes sets the quadrature point to the center of simplices (i.e., triangles in 2D and tetrahedra in 3D) with piecewise linear bases. The number of degrees of freedom in a mesh is denoted n and it is equal to the dimension d of the mesh embedding multiplied by the the number of vertices in $\mathcal{M}$.

Deformation Gradient: The distortion of $\mathcal{M}$ from a rest configuration $\bar{\mathbf{x}}$ to a posed configuration $\mathbf{x}$ can be quantified by computing the deformation gradient $\mathbf{F}$ at each quadrature point q. The deformation gradient at q indicates the linear term of the affine transformation that maps $\mathbf{x}_q = \mathbf{F}_q\bar{\mathbf{x}}_q + \mathbf{y}$, where $\mathbf{y}$ is a translation vector. Note that $\mathbf{F}_q$ depends on the basis functions used to interpolate $\bar{\mathbf{x}}$ and $\mathbf{x}$ at the quadrature point q. We denote the singular value decomposition (SVD) of the deformation gradient as $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^\top$, where $\mathbf{U}, \mathbf{V} \in \mathrm{SO}(d)$ and Σ is a diagonal matrix containing the principal stretches $\{\sigma_1, \ldots, \sigma_d\}$. The principal stretches may have negative values in order to accomodate reflections (see, e.g., Irving et al. (2004) and Twigg and Kačić-Alesić (2010)). This SVD can be used to define the polar decomposition $\mathbf{F} = \mathbf{RS}$, where $\mathbf{R} = \mathbf{U}\mathbf{V}^\top$ is a rotation and $\mathbf{S} = \mathbf{V}\Sigma\mathbf{V}^\top$ is a (symmetric) stretch tensor.

Distortion Energy: The distortion quantified by the deformation gradient at each quadrature point can be aggregated into a scalar distortion energy of the form:

$$\Psi(\mathbf{x}) = \sum\nolimits_q \Psi_q(\mathbf{F}_q)|q|, \tag{4}$$

where $\Psi(\mathbf{x})$ is the energy over the entire mesh, Ψ_q is the energy at a quadrature point, and $|q|$ is a volume weight associated with quadrature point q at rest. We can further apply the chain rule to express the first and second derivatives of Ψ in terms of $\mathbf{f} = \operatorname{vec}(\mathbf{F})$:

$$\begin{cases} \dfrac{\partial \Psi}{\partial \mathbf{x}} = \displaystyle\sum_q |q| \frac{\partial \mathbf{f}_q}{\partial \mathbf{x}}^\top \frac{\partial \Psi_q}{\partial \mathbf{f}_q}, & \text{(5a)} \\ \dfrac{\partial^2 \Psi}{\partial \mathbf{x}^2} = \displaystyle\sum_q |q| \frac{\partial \mathbf{f}_q}{\partial \mathbf{x}}^\top \left(\frac{\partial^2 \Psi_q}{\partial \mathbf{f}_q^2} \right) \frac{\partial \mathbf{f}_q}{\partial \mathbf{x}}. & \text{(5b)} \end{cases}$$

Here, $\partial \mathbf{f}_q / \partial \mathbf{x}$ is a $d^2 \times n$ matrix, $\partial \Psi_q / \partial \mathbf{f}_q$ is a vector of size d^2, and $\partial^2 \Psi_q / \partial \mathbf{f}_q^2$ is a $d^2 \times d^2$ matrix, where n is the number of degrees of freedom in $\mathcal{M}$ and d is the embedding dimension. While this change of variables is not a similarity transform and can modify the spectrum, the positive-semi definiteness of $\partial^2 \Psi_q / \partial \mathbf{f}_q^2$ guarantees the positive-semi definiteness of the resulting product in Equation 5(b). Therefore, we can construct a numerical solver that is agnostic to the discretization by analyzing the derivatives with respect to $\mathbf{F}$ (see, e.g., Sifakis and Barbic (2012)) . We will,

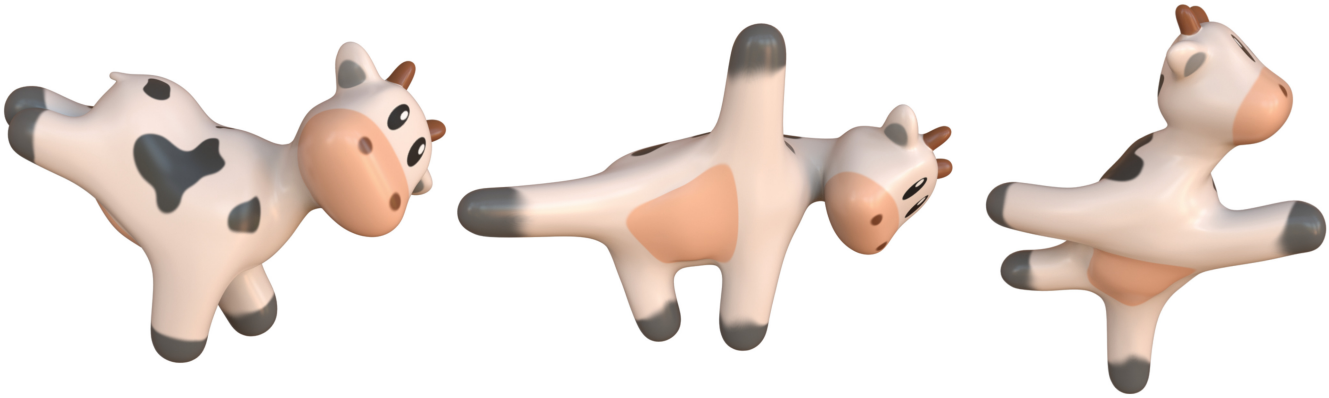

Fig. 2. A volumetric mesh with 23,237 tetrahedra is deformed into the peacock pose, the halfmoon pose, and the lord of the dance pose in an interactive modeling session. All poses are computed using our projected Newton solver with the Symmetric Dirichlet energy.

henceforth, refer to direct modifications of $\partial^2\Psi/\partial\mathbf{x}^2$ as $\mathbf{x}$-based, and direct modifications of $\partial^2\Psi_q/\partial\mathbf{f}_q^2$ as $\mathbf{F}$-based.

Our tensor notation is preferable to conventional mode-k flattenings (Kolda and Bader 2009) because it reorders the 4th-order Hessian $\partial^2\Psi_q/\partial\mathbf{F}_q^2$ into a symmetric matrix $\partial^2\Psi_q/\partial\mathbf{f}_q^2$ that admits an eigendecomposition. Our convention also differs from that of Teran et al. (2005) and Stomakhin et al. (2012), which instead cluster the diagonals of the block matrices.

4 OUR APPROACH

We now present our core contribution. First, we demonstrate that the use of scalar invariants based on $\mathbf{S}$, the stretch tensor of $\mathbf{F}$, is the correct perspective for the analysis. Then, we show that the eigensystem of each invariant can be stated in closed-form. Finally, we use the eigensystems of the invariants to arrive at a procedure for deriving the analytic eigensystems of isotropic distortion energies in both 2D and 3D. Since our analysis is performed per quadrature point, we drop the subscript q for conciseness.

4.1 S-centric Invariants

The majority of distortion energies Ψ used in geometry optimization are isotropic and can be expressed in terms of rotation-invariant scalars derived from $\mathbf{F}$. These invariants can represent the principal stretches $\{\sigma_i\}$ directly or combinations thereof. In this work, we seek invariants that facilitate the eigenanalysis of energy Hessians.

We advocate the use of invariants derived from the stretch tensor $\mathbf{S}$ that arises from the polar decomposition $\mathbf{F}=\mathbf{RS}$:

$$\boxed{\begin{aligned} I_1 &= \operatorname{tr}(\mathbf{S}) = \textstyle\sum_i \sigma_i, && \text{(6a)}\\ I_2 &= \|\mathbf{S}\|^2 = \textstyle\sum_i \sigma_i^2, && \text{(6b)}\\ I_3 &= \det(\mathbf{S}) = \textstyle\prod_i \sigma_i. && \text{(6c)} \end{aligned}}$$

The first invariant I_1 sums one-dimensional stretches, and allows the linear terms from distortion energies such as ARAP and Co-rotational elasticity to be represented. The second invariant I_2 is the Frobenius squared norm of the deformation gradient (i.e. $\|\mathbf{S}\|^2=\|\mathbf{F}\|^2$) and is used to measure the least-squares distortion. The third invariant I_3 encodes the volume change associated with $\mathbf{F}$.

Other choices of invariants have previously been investigated. Physics-driven methods (Irving et al. 2004; Teran et al. 2005), and mechanics approaches, in general (Bonet and Wood 2008; Marsden and Hughes 1994), traditionally use the invariants of the Cauchy-Green strain tensor $\mathbf{C}=\mathbf{F}^\mathsf{T}\mathbf{F}$ to obtain the alternate invariants $I_\mathbf{C}=\operatorname{tr}(\mathbf{C})$, $II_\mathbf{C}=\|\mathbf{C}\|^2$, and $III_\mathbf{C}=\det(\mathbf{C})$. However, Stomakhin et al. (2012) and Xu et al. (2015) observed that these invariants are insufficient to express the linear distortion terms that appear in many popular models, such as the Co-rotational energy. Instead, these methods expanded distortion energies in terms of principal stretches, at the cost of verbose expressions and slower computations. Alternatively, Smith et al. (2018) used the 3D versions of the I_2 and I_3 we describe here, but their analysis is incomplete because it misses the existence of I_1 entirely. Shtengel et al. (2017) and Chen and Weber (2017) proposed to use the similarity and anti-similarity parts of $\mathbf{F}$, but this decomposition is valid only in 2D.

By using the $\mathbf{S}$ invariants, we are able to perform a complete eigenanalysis that has none of the shortcomings of these previous approaches. This substitution can be employed without loss of generality, because we show in Appendix A that the 3D Cauchy-Green invariants can be written in terms of our invariants. The characteristic polynomial of $\mathbf{F}$ can also be expressed with our invariants via:

$$\text{[2D]}\quad \sigma_i^2 - I_1\sigma_i + I_3 = 0, \tag{7a}$$

$$\text{[3D]}\quad \sigma_i^3 - I_1\sigma_i^2 + \frac{1}{2}\left(I_1^2 - I_2\right)\sigma_i - I_3 = 0. \tag{7b}$$

Note that one invariant is redundant in 2D since $I_2=I_1^2-2I_3$, but we still use all three to maintain consistency between 2D and 3D.

To better understand the relationship between invariants and Hessians, we expand the quadrature-point-wise derivatives in Equation (5) in terms of $\{I_1, I_2, I_3\}$. The gradient of Ψ with respect to the flattened deformation gradient $\mathbf{f}=\operatorname{vec}(\mathbf{F})$ yields

$$\frac{\partial\Psi}{\partial\mathbf{f}} = \sum_i \frac{\partial\Psi}{\partial I_i}\frac{\partial I_i}{\partial\mathbf{f}}. \tag{8}$$

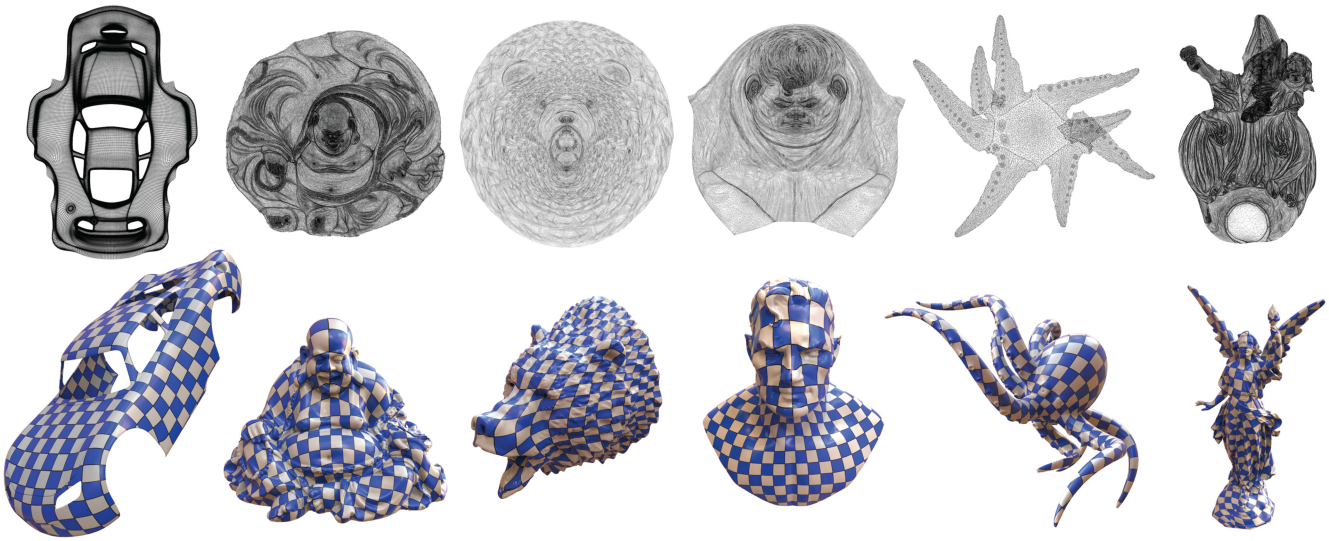

Fig. 3. Surface parameterizations computed with our algorithm applied to the Symmetric Dirichlet energy. The parameterizations are visualized in the plane (top row), and with a texture map on the 3D meshes (bottom row). From left to right, the meshes have 80k, 47k, 296k, 190k, 27k, and 1M triangles. Leftmost image ©Disney/Pixar.

The energy Hessian in terms of $\mathbf{f}$ is then:

$$\begin{aligned}\frac{\partial^2\Psi}{\partial \mathbf{f}^2} = &\sum_i \frac{\partial \Psi}{\partial I_i}\overbrace{\frac{\partial^2 I_i}{\partial \mathbf{f}^2}} \\ &+ \sum_i \frac{\partial^2 \Psi}{\partial I_i^2}\underbrace{\left(\frac{\partial I_i}{\partial \mathbf{f}}\right)\left(\frac{\partial I_i}{\partial \mathbf{f}}\right)^\top} \\ &+ \sum_{i<j} \frac{\partial^2 \Psi}{\partial I_i \partial I_j}\underbrace{\left[\left(\frac{\partial I_i}{\partial \mathbf{f}}\right)\left(\frac{\partial I_j}{\partial \mathbf{f}}\right)^\top + \left(\frac{\partial I_j}{\partial \mathbf{f}}\right)\left(\frac{\partial I_i}{\partial \mathbf{f}}\right)^\top\right]}.\end{aligned} \tag{9}$$

Note that all the 2nd-order tensors (highlighted in red overbrace and blue underbrace) are composed of first and second derivatives of the invariants, so $\{I_1, I_2, I_3\}$ can be used as proxies for analyzing the eigenstructure of the Hessian. We perform this analysis next.

4.2 Eigensystems of Invariants

The eigenvalues and eigenvectors of our $\mathbf{S}$-based invariants can be stated *in closed-form*. In deriving these expressions, we will make heavy use of the tensor notation defined in Section 3. We note that two of the following eigensystems, the ones for I_2 and the 3D version of I_3, were also shown in Smith et al. (2018).

Eigensystem of I_1: The gradient of I_1 with respect to $\mathbf{F}$ is the rotation $\mathbf{R}$ from the polar decomposition of $\mathbf{F}$:

$$\frac{\partial I_1}{\partial \mathbf{f}} = \mathrm{vec}\left(\frac{\partial I_1}{\partial \mathbf{F}}\right) = \mathrm{vec}\,(\mathbf{R}) = \mathbf{r}. \tag{10}$$

The Hessian of I_1 is then the gradient of $\mathbf{R}$. So-called "rotation gradients" appear in many contexts, such as articulated dynamics (Twigg and Kačić-Alesić 2010) and computer vision (Papadopoulo and Lourakis 2000), and are usually computed by numerically differentiating the SVD. This operation tends to be fairly opaque, as it requires a matrix inverse (Barbič and Zhao 2011) that does not reveal any underlying structure. However, we have found that rotation gradients *have simple closed-form expressions in 2D and 3D.* We first consider the 2D case and introduce the $\mathbf{F}$-based *twist* matrix:

$$\mathbf{T} = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{V}^T. \tag{11}$$

The Hessian of I_1 in 2D can then be written in closed-form as:

$$\frac{\partial^2 I_1}{\partial \mathbf{f}^2} = \frac{\partial \mathbf{r}}{\partial \mathbf{f}} = \frac{2}{\sigma_1+\sigma_2}\mathbf{t}\mathbf{t}^\top, \tag{12}$$

where $\mathbf{t} = \mathrm{vec}\,(\mathbf{T})$ is a unit norm vector. Thus, the Hessian $\partial^2 I_1/\partial \mathbf{f}^2$ is a rank-one matrix with eigenvector $\mathbf{t}$ and eigenvalue inversely proportional to the average principal stretch. The 3D case involves three $\mathbf{F}$-based twists $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$, one for each axis. For example, the x-axis twist is:

$$\mathbf{T}_1 = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}\mathbf{V}^T. \tag{13}$$

The 3D Hessian of I_1 is then:

$$\frac{\partial^2 I_1}{\partial \mathbf{f}^2} = \frac{2}{\sigma_2+\sigma_3}\mathbf{t}_1\mathbf{t}_1^\top + \frac{2}{\sigma_3+\sigma_1}\mathbf{t}_2\mathbf{t}_2^\top + \frac{2}{\sigma_1+\sigma_2}\mathbf{t}_3\mathbf{t}_3^\top, \tag{14}$$

which is a rank-three matrix with eigenvectors $\mathbf{t}_i$ and eigenvalues inversely proportional to the average orthogonal principal stretches.

Eigensystem of I_2: We compute the derivatives of I_2 by noting that $I_2 = \|\mathbf{S}\|^2 = \|\mathbf{F}\|^2 = \|\mathbf{f}\|^2$, which yields

$$\frac{\partial I_2}{\partial \mathbf{f}} = 2\mathbf{f} \quad \text{and} \quad \frac{\partial^2 I_2}{\partial \mathbf{f}^2} = 2\mathbf{I}. \tag{15}$$

Here, $\mathbf{I}$ is a $d^2 \times d^2$ identity matrix ($d = 2$ or 3). The Hessian of I_2 is full-rank with a repeated eigenvalue equal to two, and the eigenvectors can be set to any orthonormal bases. This invariant thus serves as a regularizer for distortion energies expressed in terms of $\mathbf{F}$.

Eigensystem of I_3: The third invariant is more challenging since it is quadratic (cubic) in 2D (3D) with respect to $\mathbf{F}$. The 2D gradient is:

$$\frac{\partial I_3}{\partial \mathbf{f}} = \mathbf{g} = \text{vec}\,(\mathbf{G}) \quad \text{with} \quad \mathbf{G} = \mathbf{U}\begin{bmatrix}\sigma_2 & 0\\ 0 & \sigma_1\end{bmatrix}\mathbf{V}^\top. \tag{16}$$

For the Hessian, we again need the $\mathbf{F}$-based twist matrix $\mathbf{T}$, but additionally introduce the following *flip* ($\mathbf{L}$) and *pinch* ($\mathbf{P}$) matrices:

$$\mathbf{L} = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\mathbf{V}^\top \quad \text{and} \quad \mathbf{P} = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\mathbf{V}^\top. \tag{17}$$

By denoting $\mathbf{l} = \text{vec}\,(\mathbf{L})$ and $\mathbf{p} = \text{vec}\,(\mathbf{P})$, we can verify that $\{\hat{\mathbf{r}}, \mathbf{p}, \mathbf{l}, \mathbf{t}\}$ form orthonormal bases, where $\hat{\mathbf{r}}$ denotes the normalized rotation vector, i.e., $\hat{\mathbf{r}} = \mathbf{r}/\sqrt{2}$. The eigensystem of I_3 in 2D is then:

$$\frac{\partial^2 I_3}{\partial^2 \mathbf{f}} = \hat{\mathbf{r}}\hat{\mathbf{r}}^\top + \mathbf{t}\mathbf{t}^\top - \mathbf{p}\mathbf{p}^\top - \mathbf{l}\mathbf{l}^\top. \tag{18}$$

The eigenvalues are, respectively, 1 for the $\hat{\mathbf{r}}$ and $\mathbf{t}$ eigenvectors, and -1 for $\mathbf{p}$ and $\mathbf{l}$. The rank-two subspaces spanned by $\{\hat{\mathbf{r}}, \mathbf{t}\}$ and $\{\mathbf{p}, \mathbf{l}\}$ are, thus, arbitrary, but we have found this particular decomposition to be geometrically intuitive. Similar to 2D, the 3D gradient is:

$$\frac{\partial I_3}{\partial \mathbf{f}} = \mathbf{g} = \text{vec}\,(\mathbf{G}) \quad \text{with} \quad \mathbf{G} = \mathbf{U}\begin{bmatrix}\sigma_2\sigma_3 & 0 & 0\\ 0 & \sigma_3\sigma_1 & 0\\ 0 & 0 & \sigma_1\sigma_2\end{bmatrix}\mathbf{V}^\top. \tag{19}$$

Similar to Equation (13), the 3D versions of the flip matrices $\{\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3\}$ now appear along each axis. For example, the x-axis flip is:

$$\mathbf{L}_1 = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}\mathbf{V}^T. \tag{20}$$

The arbitrary subspaces from the 2D case disappear, so no 3D versions of $\mathbf{R}$ or $\mathbf{P}$ are needed. Each twist $\mathbf{t}_i$ is an eigenvector with corresponding eigenvalue σ_i, and $\mathbf{l}_i$ is an eigenvector paired with $-\sigma_i$. The remaining eigenpairs are the solutions of the depressed cubic derived by Smith et al. (2018), which we list in the supplement.

4.3 Eigensystems of Arbitrary Isotropic Energies

We now have all the components necessary to revisit Equation (9). To begin, we prove in Appendix B that the twist $\mathbf{t}$ and flip $\mathbf{l}$ vectors are orthogonal to the gradient of any invariant in 2D and in 3D:

$$\left(\frac{\partial I_i}{\partial \mathbf{f}}\right)^\top \mathbf{t} = \left(\frac{\partial I_i}{\partial \mathbf{f}}\right)^\top \mathbf{l} = 0. \tag{21}$$

Consequently, they have zero projection onto the blue (underbraced) matrices in Equation (9). The remaining red (overbraced) terms are all Hessians of invariants ($\partial^2 I_i/\partial \mathbf{f}^2$), and we can establish, via multiplication, that $\mathbf{t}$ and $\mathbf{l}$ are eigenvectors of any 2D isotropic energy. The eigenvalues λ then take the analytic form:

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{t} = \left[\frac{2}{\sigma_1+\sigma_2}\frac{\partial \Psi}{\partial I_1} + 2\frac{\partial \Psi}{\partial I_2} + \frac{\partial \Psi}{\partial I_3}\right]\mathbf{t} = \lambda_{\text{twist}}\mathbf{t}, \tag{22a}$$

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{l} = \left[2\frac{\partial \Psi}{\partial I_2} - \frac{\partial \Psi}{\partial I_3}\right]\mathbf{l} = \lambda_{\text{flip}}\mathbf{l}. \tag{22b}$$

Similarly, we can show that $\{\mathbf{t}_i\}$ and $\{\mathbf{l}_i\}$ are eigenvectors of any 3D isotropic energy and have the analytic eigenvalues:

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{t}_1 = \left[\frac{2}{\sigma_2+\sigma_3}\frac{\partial \Psi}{\partial I_1} + 2\frac{\partial \Psi}{\partial I_2} + \sigma_1\frac{\partial \Psi}{\partial I_3}\right]\mathbf{t}_1 = \lambda_{\text{x-twist}}\mathbf{t}_1, \tag{23a}$$

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{l}_1 = \left[2\frac{\partial \Psi}{\partial I_2} - \sigma_1\frac{\partial \Psi}{\partial I_3}\right]\mathbf{l}_1 = \lambda_{\text{x-flip}}\mathbf{l}_1. \tag{23b}$$

We have now unveiled the expressions for two (six) eigenpairs in 2D (3D). These expressions can be used to obtain the respective analytic eigenpairs for *any* isotropic energy. The remaining two (three) eigenpairs in 2D (3D) depend on the blue outer product terms in Equation (9). In the most general case, these terms result in a quadratic (cubic) system in 2D (3D) that is encoded in a 2×2 (3×3) matrix $\mathbf{A}$. We can construct this matrix $\mathbf{A}$ by probing $\partial^2\Psi/\partial\mathbf{f}^2$ with scaling modes; e.g., the x-scaling vector in 2D is

$$\mathbf{D}_1 = \mathbf{U}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\mathbf{V}^\top. \tag{24}$$

The entries of $\mathbf{A}$ are obtained by computing the coefficients:

$$a_{ij} = \mathbf{d}_i^\top\left(\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\right)\mathbf{d}_j, \tag{25}$$

where $\mathbf{d}_i = \text{vec}(\mathbf{D}_i)$ is a unit vector orthogonal to the twist and flip vectors. The explicit expressions of a_{ij} are listed in Appendix D. We have found that, for the most popular distortion energies in graphics applications, the off-diagonal entries a_{ij} resolve to zero, and the scaling vectors are in fact the eigenvectors. In these cases, we can write simplified closed-form expressions for the last d eigenpairs. In 2D, the scaling vector $\mathbf{d}_1$ has the eigenvalue

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{d}_1 = \left[2\frac{\partial \Psi}{\partial I_2} + \frac{\partial^2 \Psi}{\partial I_1^2} + 4\sigma_1^2\frac{\partial^2 \Psi}{\partial I_2^2} + \sigma_2^2\frac{\partial^2 \Psi}{\partial I_3^2} + 4\sigma_1\frac{\partial^2 \Psi}{\partial I_1 \partial I_2} + 4I_3\frac{\partial^2 \Psi}{\partial I_2 \partial I_3} + 2\sigma_2\frac{\partial^2 \Psi}{\partial I_3 \partial I_1}\right]\mathbf{d}_1 = \lambda_{\text{x-scale}}\mathbf{d}_1, \tag{26}$$

and the eigenvalue for $\mathbf{d}_2$ reverses the positions of σ_1 and σ_2 in the above equation. In 3D, the eigenvalue for $\mathbf{d}_1$ becomes:

$$\frac{\partial^2 \Psi}{\partial \mathbf{f}^2}\mathbf{d}_1 = \left[2\frac{\partial \Psi}{\partial I_2} + \frac{\partial^2 \Psi}{\partial I_1^2} + 4\sigma_1^2\frac{\partial^2 \Psi}{\partial I_2^2} + \sigma_2^2\sigma_3^2\frac{\partial^2 \Psi}{\partial I_3^2} + 4\sigma_1\frac{\partial^2 \Psi}{\partial I_1 \partial I_2} + 4I_3\frac{\partial^2 \Psi}{\partial I_2 \partial I_3} + 2\sigma_2\sigma_3\frac{\partial^2 \Psi}{\partial I_3 \partial I_1}\right]\mathbf{d}_1 = \lambda_{\text{x-scale}}\mathbf{d}_1. \tag{27}$$

The eigenvalues for $\mathbf{d}_2$ and $\mathbf{d}_3$ respectively follow from σ_2 and σ_3, and are listed explicitly in Appendix E.

We now have a method for deriving analytic expressions for the complete eigensystem of any arbitrary isotropic energy. In all cases, simple expressions for d^2-d eigenpairs can be computed directly with Equations (22) and (23). As we will show in the next section, the off-diagonal entries in Equation (25) are often zero and yield compact expressions for the last d eigenpairs. Finally, even in the cases where a non-diagonal matrix $\mathbf{A}$ is present, we will show in Section 5.3 that relatively simple expressions appear.

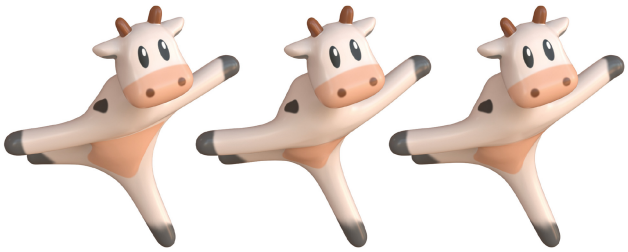

Fig. 4. A tetrahedral mesh of Spot the Cow is contorted with our method by minimizing the energies ARAP (left), Symmetric Dirichlet (center), and Symmetric ARAP (right), timing 2.46s, 21.60s, and 23.19s, respectively.

5 ENERGY EIGENSYSTEMS

To demonstrate the power of the expressions from the previous section, we use them to derive the analytic eigensystems for several popular distortion energies: ARAP, Symmetric Dirichlet, and MIPS. In Appendix F, we illustrate the generality of our approach by applying it to several other energies that are additive extensions of those in this section. We also provide Matlab scripts in the supplemental materials that symbolically validate these analytic eigensystems in both 2D and 3D. We denote eigenvalues with λ_i and the corresponding eigenvectors with $\mathbf{e}_i = \mathrm{vec}(\mathbf{E}_i)$.

5.1 ARAP Energy

The ARAP energy is commonly stated as $\Psi_{\mathrm{ARAP}} = \|\mathbf{F}-\mathbf{R}\|^2$ (see, e.g., Chao et al. (2010)), and can be written in terms of the $\mathbf{S}$ invariants as: $\Psi_{\mathrm{ARAP}} = \|\mathbf{F}\|^2 - 2\,\mathrm{tr}\,(\mathbf{S}) + \|\mathbf{R}\|^2 = I_2 - 2I_1 + d^2$. Applying Equation (22) to this expression yields the first two eigenpairs in 2D:

$$\begin{aligned} \lambda_1^{\mathrm{2D}} &= 2 - 4/(\sigma_1+\sigma_2) \qquad & \mathbf{e}_1 &= \mathbf{t}, \\ \lambda_2^{\mathrm{2D}} &= 2 & \mathbf{e}_2 &= \mathbf{l}. \end{aligned} \tag{28}$$

Next, applying Appendix D reveals that the scaling modes decouple because all the a_{ij} off-diagonals resolve to zero. Therefore, applying Equation (26) yields the other two eigenpairs:

$$\lambda_{3,4}^{\mathrm{2D}} = 2 \qquad \mathbf{e}_{3,4} = \mathbf{d}_1, \mathbf{d}_2. \tag{29}$$

An equivalent process yields the 3D eigenpairs:

$$\begin{aligned} \lambda_1^{\mathrm{3D}} &= 2 - 4/(\sigma_2+\sigma_3) \qquad & \mathbf{e}_1 &= \mathbf{t}_1, \\ \lambda_2^{\mathrm{3D}} &= 2 - 4/(\sigma_3+\sigma_1) & \mathbf{e}_2 &= \mathbf{t}_2, \\ \lambda_3^{\mathrm{3D}} &= 2 - 4/(\sigma_1+\sigma_2) & \mathbf{e}_3 &= \mathbf{t}_3, \\ \lambda_{4\ldots9}^{\mathrm{3D}} &= 2 & \mathbf{e}_{4\ldots9} &= \{\mathbf{l}_i\}, \{\mathbf{d}_i\}. \end{aligned} \tag{30}$$

Both 2D and 3D have repeated eigenvalues, respectively $\lambda_{2\ldots4}$ and $\lambda_{4\ldots9}$. While we listed explicit eigenvectors for concreteness, the bases for this subspace are, in fact, arbitrary. These expressions translate into straightforward code inside a projected Newton solver. The arbitrary subspace can be constructed using a diagonal matrix, and since the twist-based eigenvalues are the only ones that admit negative values, semi-positive-definiteness can be easily guaranteed. A 2D implementation is short enough to be listed in Appendix C.

The Co-rotational model (McAdams et al. 2011) builds on the ARAP model by adding a linearized volume penalty term. We show in Appendix F.1 that its eigensystem is correspondingly similar. As with ARAP, the arbitrary subspace does not need to be explicitly constructed.

5.2 Symmetric Dirichlet

We first point out that the Dirichlet energy $\Psi_{\mathrm{D}} = 1/2\|\mathbf{F}\|^2$ is in fact equivalent to half of our I_2, so its eigensystem is $\mathbf{I}$ (see Equation (15)). It is also well-known that this energy's full Hessian $\partial^2\Psi/\partial\mathbf{x}^2$ produces a Laplacian (Botsch et al. 2010). Therefore, we can conclude that an identity regularizer for the $\mathbf{F}$-based Hessian is equivalent to a Laplacian regularizer for the final $\mathbf{x}$-based Hessian.

We next examine the Symmetric Dirichlet energy (Smith and Schaefer 2015), which is defined as $\Psi_{\mathrm{SD}} = (\|\mathbf{F}\|^2 + \|\mathbf{F}^{-1}\|^2)/2$, and show that $\mathbf{F}$ inverses do not introduce any new difficulties. In 2D, this energy can be written using our $\mathbf{S}$ invariants as $\Psi_{\mathrm{SD}}^{\mathrm{2D}} = (I_2 + I_2/I_3^2)/2$. Similar to the ARAP case, the scaling modes decouple, so the 2D eigenpairs can be stated as:

$$\lambda_1^{\mathrm{2D}} = 1 + 3/\sigma_1^4 \qquad \mathbf{e}_1 = \mathbf{d}_1 \tag{31a}$$

$$\lambda_2^{\mathrm{2D}} = 1 + 3/\sigma_2^4 \qquad \mathbf{e}_2 = \mathbf{d}_2 \tag{31b}$$

$$\lambda_3^{\mathrm{2D}} = 1 + 1/I_3^2 + I_2/I_3^3 \qquad \mathbf{e}_3 = \mathbf{l} \tag{31c}$$

$$\lambda_4^{\mathrm{2D}} = 1 + 1/I_3^2 - I_2/I_3^3 \qquad \mathbf{e}_4 = \mathbf{t}. \tag{31d}$$

In 3D, the inverse term is more intricate and the energy can be written using the $\mathbf{S}$ invariants as:

$$\Psi_{\mathrm{SD}}^{\mathrm{3D}} = \frac{1}{2}I_2 + \frac{1}{8}\left(\frac{I_1^2 - I_2}{I_3}\right)^2 - \frac{I_1}{I_3}. \tag{32}$$

The three scaling eigenpairs are

$$\lambda_i^{\mathrm{3D}} = 1 + 3/\sigma_i^4 \qquad \mathbf{e}_i = \mathbf{d}_i, \tag{33}$$

while the three twist eigenpairs are

$$\lambda_{i+3}^{\mathrm{3D}} = 1 + \sigma_i^2/I_3^2 - (I_2 - \sigma_i^2)\sigma_i^3/I_3^3 \qquad \mathbf{e}_{i+3} = \mathbf{t}_i. \tag{34}$$

Finally, the flip eigenpairs are

$$\lambda_{i+6}^{\mathrm{3D}} = 1 + \sigma_i^2/I_3^2 + (I_2 - \sigma_i^2)\sigma_i^3/I_3^3 \qquad \mathbf{e}_{i+6} = \mathbf{l}_i. \tag{35}$$

Our eigenanalysis reveals a singularity in Symmetric Dirichlet under degenerate configurations and will generally reveal fundamental singularities in any distortion energy that it is applied to. Conversely, if a distortion energy is singularity-free, our analysis will produce singularity-free eigenvalue expressions. See Appendix F.3 for an example. Consistent with previous works, our Symmetric Dirichlet tests have found that this singularity requires no special treatment.

The Symmetric ARAP energy (Shtengel et al. 2017) adds several new terms to the Symmetric Dirichlet energy, particularly involving I_1. We show in Appendix F.2 that the resulting eigensystem is similar to Symmetric Dirichlet.

5.3 MIPS Energy

The MIPS energy (Horman and Greiner 1999) is defined in 2D as $\Psi_{\mathrm{MIPS}} = I_2/I_3$. We examine it here as a case where the scaling modes remain coupled. The twist and flip eigenpairs are, respectively:

$$\lambda_1^{\mathrm{2D}} = 2/I_3 - I_2/I_3^2 \qquad \mathbf{e}_1 = \mathbf{t}, \tag{36a}$$

$$\lambda_2^{\mathrm{2D}} = 2/I_3 + I_2/I_3^2 \qquad \mathbf{e}_2 = \mathbf{l}. \tag{36b}$$

Fig. 5. Our closed-form eigensystems can be combined with a projected Newton solver to robustly interpolate from a relaxed hand (left) to a clenching hand (right). The hand model is discretized with a tetrahedral mesh composed of 23,692 elements, as displayed by the cutaways. In this example, we minimized the 3D Symmetric ARAP energy, which took 0.21 seconds per frame. ©Disney/Pixar

The coupled scaling modes are:

$$\lambda_3^{2D} = I_2(I_2 - \alpha) - 2I_3^2 \qquad \mathbf{e}_3 = \frac{1}{\gamma}\left(\beta\mathbf{d}_1 + \mathbf{d}_2\right), \tag{37a}$$

$$\lambda_4^{2D} = I_2(I_2 + \alpha) - 2I_3^2 \qquad \mathbf{e}_4 = \frac{1}{\gamma}\left(\mathbf{d}_1 - \beta\mathbf{d}_2\right), \tag{37b}$$

where $\alpha = \sqrt{I_2^2 - 3I_3^2}$ and $\beta = I_2/(\sigma_2^2 - \sigma_1^2 + \alpha)$. The $\gamma = \sqrt{1+\beta^2}$ term normalizes the eigenvector.

The MIPS eigensystem indicates that, even when the 2D scaling modes are coupled (i.e., $a_{12} \neq 0$), it is still possible to solve the underlying quadratic analytically and obtain closed-form expressions for the remaining eigenpairs. More concretely, given the 2D matrix $\mathbf{A}$ with non-zero off-diagonals, the eigenvalues λ_3^{2D} and λ_4^{2D} correspond to the roots of the characteristic polynomial of $\mathbf{A}$ and can be computed using the quadratic formula, while the eigenvectors $\mathbf{e}_3$ and $\mathbf{e}_4$ follow Equation (37) with $\beta = (\lambda_3^{2D} - a_{22})/a_{12}$. The 3D case involves a cubic that can yield larger expressions, which can be seen, for instance, in the equations found by Smith et al. (2018). Although an analytical expression is available for the 3D eigenvalues (using, e.g., Cardano's formula), solving a 3×3 eigendecomposition can sometimes be the most expedient way of computing the last three eigenpairs. Finally, we point out that, given an eigenvector $\mathbf{z} = [z_1\ z_2\ z_3]^\top$ of the matrix $\mathbf{A}$, the corresponding eigenvector for the energy Hessian is $\mathbf{e} = \sum_i z_i \mathbf{d}_i$.

5.4 Discussion

We conclude this section by discussing the relationship of our results to previous work. Additional analyses of these relationships are available in the supplemental material.

Comparison to Accelerated Quadratic Proxy (AQP). Kovalsky et al. (2016) presented an AQP method that yielded convergence speedups by replacing the energy Hessian with a Laplacian matrix. Relatedly, we showed in Section 5.2 that the second derivative of I_2 introduces an identity matrix at each quadrature point that contributes a Laplacian to the global Hessian matrix. Therefore, the method of Kovalsky et al. (2016) can be viewed as a looser approximation of the energy Hessian that assigns the $\mathbf{F}$-based Hessian to identity at each quadrature point. In the case of the ARAP energy, for instance, this corresponds to the removal of part of the contribution of the twist eigenvectors from Section 5.1.

Comparison to Composite Majorization (CM). The CM method of Shtengel et al. (2017) considered Hessian projection to positive semi-definiteness computed per (simplicial) mesh element. However, the extension of their composite majorization to 3D appears to be difficult because it relies on the convex-concave decomposition of expressions for the singular values of the deformation gradient $\mathbf{F}$. Such expressions exist in 2D and can be obtained using the norm of the similarity and anti-similarity parts of $\mathbf{F}$, but this no longer holds in 3D because the problem becomes cubic. Moreover, the majorizer approximation in Shtengel et al. (2017) (Equation (9) in that paper) projects each term in the Hessian separately and then sums the result. Consequently, it can introduce approximations even when the Hessian was already positive-definite. By exploiting the eigenstructure of the $\mathbf{S}$ invariants, our method takes a more holistic view and projects the Hessian after all of its terms have been gathered. If a quadrature point is already positive semi-definite, our approach keeps its Hessian untouched. An extensive case study of our approach compared to CM for the specific case of the ARAP energy is presented in the supplement.

Comparison to Chen and Weber (2017). Closely related to our work, Chen and Weber (2017) considered analytic eigensystems for isotropic distortion energies. However, their results are limited to 2D because their analysis is dependent on complex derivatives. Moreover, their closed-form expressions exhibit a series of invalid states that need to be resolved on a case-by-case basis. In our supplement, we revisit the derivation of Chen and Weber (2017) and detail the causes of their degenerated configurations. In contrast, our method is suited to any isotropic energy both in 2D and 3D.

Comparison to Stomakhin et al. (2012); Teran et al. (2005). The work of Teran et al. (2005) used the **C**-centric invariants (see Appendix A) to show that the indefiniteness of 3D Hessians can be probed using three 2×2 eigenproblems and one 3×3 eigenproblem. Stomakhin et al. (2012) extended these results by replacing the **C** invariants with the 3D principal stretches. However, none of these analyses reveal the closed-form eigensystem expressions we found here. In particular, these previous techniques assembled the Hessian eigenvectors numerically via a series of tensor contractions. As we will show in Section 6, our analytic expressions offer simpler code, leading up to a 3.69× speedup in Hessian construction compared to Stomakhin et al. (2012).

Comparison to Smith et al. (2018). The work of Smith et al. (2018) established the eigensystems of the I_C invariant and $J = \det \mathbf{F}$, which correspond to our I_2 and I_3, respectively. However, they failed to find the eigensystem for the II_C invariant (see their ğ7), which prevented them from fully characterizing all Cauchy-Green-based energies. Most significantly, their formulation missed the existence of the I_1 invariant entirely, making it impossible to discover almost all of the closed-form eigensystems that we presented in Section 5. With the exception of the MIPS energy, all of the energies we address in this article contain the I_1 invariant, which puts these energies beyond the reach of the analysis in Smith et al. (2018). Conversely, our analysis yields the analytic eigensystems for both I_1 and II_C. As shown in Appendix A, the II_C invariant can be written in terms of our **S** invariants, thus, establishing its eigensystem as just a specific application of our derivation. We are thus able to explicitly list the closed-form eigensystem of II_C in ğ3 of the supplemental materials.

6 RESULTS

We exercise our method on a range of graphics applications that span geometry processing and physical simulation. On the geometry processing front, we apply our approach to surface parameterization (Section 6.1), volume deformation (Section 6.2), and interpolation (Section 6.3). Halfway between geometry processing and physical simulation, we show that our method can be used as a preprocessing procedure to obtain intersection-free input surfaces for a cloth simulator (Section 6.4). Finally, we show that our approach can be used in volume simulation with dynamics (Section 6.5). In these tasks, we employ triangle meshes with linear elements in 2D and tetrahedral meshes with linear elements in 3D. We use a standard projected Newton solver augmented with a line search for all tests. Algorithm 1 shows pseudo-code of our projected Newton, and includes our analytical Hessian computation. To prevent elements from inverting with Symmetric Dirichlet and Symmetric ARAP, we use the backtracking line search of Smith and Schaefer (2015) to limit the step-size to the largest inversion-avoiding value. For all remaining energies, we use a simple backtracking line search. We terminate the minimization when the norm of the gradient falls below a threshold: $\|\nabla\Psi\|_\infty \leq 10^{-4}$. In 2D, we solve linear systems using the Cholesky decomposition from the Intel Math Kernel Library (MKL) (Intel 2018). For AQP, the Laplacian decouples across each dimension, so we pre-factorize and solve two independent systems. In 3D, computing this decomposition is more costly due to the increased bandwidth of the stiffness matrix.

ALGORITHM 1: Projected Newton Pseudocode. Our approach allows `Eval_Energy_EigenSystem`(**U**, Σ, **V**) to be implemented in closed-form.

Function `Projected_Newton_Solver`($\mathbf{x}_0$)
  for $i \leftarrow 0$ ***to*** n **do**
    $\mathbf{b}_i \leftarrow \nabla\Psi(\mathbf{x}_i)$ // Equation (5a)
    if $\|\mathbf{b}_i\|_\infty \leq 10^{-4}$ **then**
      return $\mathbf{x}_i$
    end
    $\mathbf{H}_i \leftarrow$ `Project_Hessian`$(\mathbf{x}_i)$
    $\mathbf{d}_i \leftarrow -\mathbf{H}_i^{-1}\mathbf{b}_i$
    $\alpha_i \leftarrow$ `Line_Search`$(\mathbf{x}_i, \mathbf{d}_i)$
    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \alpha_i \mathbf{d}_i$
  end
  return $\mathbf{x}_{n+1}$
Function `Project_Hessian`(**x**)
  $\mathbf{H} \leftarrow \mathbf{0}$
  for *every quadrature point* q **do**
    $\mathbf{F} \leftarrow$ `Compute_Deformation_Gradient`(q, **x**)
    $\mathbf{f} \leftarrow \mathrm{vec}(\mathbf{F})$
    $\{\mathbf{U}, \Sigma, \mathbf{V}\} \leftarrow$ `Compute_SVD`(**F**)
    $\{\lambda_i, \mathbf{e}_i\} \leftarrow$ `Eval_Energy_EigenSystem`(**U**, Σ, **V**)
    $\mathbf{H}_q \leftarrow \sum_i \max(\lambda_i, 0)\, \mathbf{e}_i \mathbf{e}_i^\top$
    $\mathbf{H} \leftarrow \mathbf{H} + |q|\, (\partial\mathbf{f}/\partial\mathbf{x})^\top \mathbf{H}_q\, (\partial\mathbf{f}/\partial\mathbf{x})$ // Equation (5b)
  end
  return **H**

Instead, we found that Jacobi-preconditioned conjugate gradient provides adequate performance. All SVDs and numerical eigendecompositions were computed with Eigen (Guennebaud et al. 2010).

6.1 Surface Parameterization

Figure 3 shows the parameterization of six meshes computed by minimizing the Symmetric Dirichlet energy with our approach, where each was initialized with a Tutte embedding of the 3D surface. In Figure 6 (left), we compare the performance for these examples obtained with our analytic projected Newton solver, composite majorization (Shtengel et al. 2017) (using code provided by authors), SLIM (Rabinovich et al. 2017) (using the optimized parallel code provided by the authors), and a numerical projected Newton solver (see, e.g., Fu and Liu (2016)). The latter performs the numerical eigendecomposition of 6×6 matrices corresponding to the **x**-based Hessian per simplicial element. In contrast, our approach computes the analytic eigensystems for **F**-based Hessians of size 4×4 evaluated per quadrature point. We also employed our closed-form expressions to assemble the unclamped **x**-based Hessians in lieu of auto-differentiation, so the implementation of the numerical projected Newton solver was also improved by our results.

For the Symmetric Dirichlet energy, composite majorization and our method gave nearly identical performances in all but one test, and yielded consistently faster solves than the numerical projected Newton solver. While the latter can terminate in fewer iterations than the competing methods, the cost of the 6×6 per-element eigendecompositions outweigh these savings across all tests. Figure 9 plots the optimization progress for the bear model, both in terms of iterations and wall clock time (in seconds). In the

	Symmetric Dirichlet								ARAP							
	Ours		CM		Proj. Newton		SLIM		Ours		S-based CM		Proj. Newton		AQP, α/β CM	
Mesh	Iters.	Time	Iters.	Time	Iters.	Time	Iters.	Time	Iters.	Time	Iters.	Time	Iters.	Time	Iters.	Time
Bear	16	13.7	16	14.1	27	38.8	100	164.7	5	4.7	79	72.8	10	14.9	9	5.2
Buddha	17	25.6	17	25.6	15	36.5	144	461.5	3	4.7	24	38.3	6	15.0	8	8.3
Lucy	123	443.9	124	448.8	111	602.9	15256	163034	116	429.4	116	443.3	56	465.2	616	1177.9
Man	16	8.7	17	9.0	16	14.2	94	106.7	6	3.4	21	12.1	9	8.6	12	4.2
Octopus	37	2.2	36	2.2	63	6.9	1442	270.5	75	4.9	184	12.7	57	6.8	409	14.6
Car	44	1.2	44	1.3	39	1.9	651	38.9	175	5.5	198	6.7	77	3.8	338	4.7
Car 2	49	23.0	50	23.5	113	88.2	176	178.6	8	3.7	21	9.9	10	7.6	27	6.6

Fig. 6. Total Newton iterations and wall clock time (s) to compute surface parameterizations with Symmetric Dirichlet (left) and ARAP (right) using **F**-based per-quadrature point Hessian projection (Ours), composite majorization (CM), SLIM, and **x**-based per-element numerical projection (Proj. Newton). The fastest time for each mesh is highlighted in blue. Multiple methods are highlighted in case of a tie, where two timings are within 3% of each other. Two CM methods were considered for ARAP: one based on our invariants (**S**-based CM) and one based on the expansion from Shtengel et al. (2017) (α/β-based CM, which is equivalent to AQP (Kovalsky et al. 2016)).

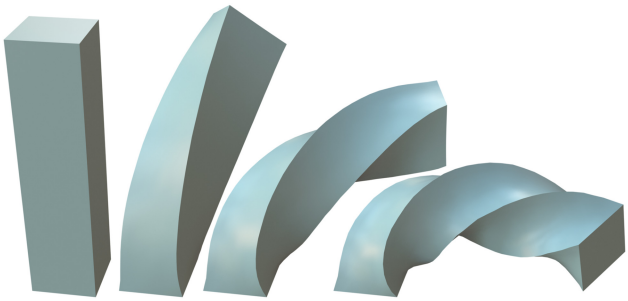

Fig. 7. We interpolate between an undeformed, tetrahedralized bar (left) and a deformed bar (right) by minimizing the 3D Symmetric ARAP energy.

supplemental material, we provide the plots for the other meshes, as well as the performance statistics of a larger dataset of 41 models.

We also computed surface parameterizations using the ARAP energy. The composite majorization for ARAP was not addressed in Shtengel et al. (2017), so we considered two candidate derivations: one using the convex-concave decomposition obtained by their similarity and anti-similarity expansion of **F**, and another expressed in terms of our **S** invariants (detailed derivations are in the supplemental material). In contrast to our formulation, we observed that both composite majorization schemes led to looser Hessian approximations even in the regime where the energy Hessian is already positive semi-definite. As shown in the supplement, the behavior of CM is highly dependent on the choice of decomposition. A naïve choice produces a solve that is equivalent to AQP (Kovalsky et al. 2016), degrading convergence to first-order. In Figure 10, we show the number of iterations and the wall clock time (in seconds) for the bear model. We compared the performance of these methods to our solver over a dataset of 41 models, and complete results are in the supplemental material. Figure 6 (right) shows a selection of these results. In a few instances, the **x**-based formulation completed sooner, but for the vast majority of cases, our approach was clearly superior. In particular, for the low resolution car mesh (Car, Figure 6), the **x**-based formulation had the best performance. On a higher resolution version of the same car mesh obtained through Catmull-Clark subdivision (Car 2, Figure 6), however, we found that our method had the fastest performance.

6.2 Volume Deformation

We explored the effectiveness of our approach when computing a 3D deformation. Given a volumetric cow mesh containing 23k tetrahedra, we generated handles from the feet and head by hard-constraining and rigidly transforming groups of surface mesh vertices. The resulting position of the deformed mesh was computed by minimizing the distortion energy when a handle was transformed.

Figure 4 shows the results obtained after prescribing a transform to each handle. Our projected Newton solver successfully converged to a deformed shape, taking 2.46s with ARAP, 21.60s with Symmetric Dirichlet, and 23.19s with Symmetric ARAP. As composite majorization does not generalize to 3D, we instead compared to the per-element numerical projected Newton solver. Symmetric Dirichlet and Symmetric ARAP with per-quadrature point (9×9) numerical projection took a total of 40.91s and 53.97s, and Hessian projection was 5.49× and 7.89× slower than our method, respectively. Symmetric Dirichlet and Symmetric ARAP with per-element (12×12) numerical projection took a total of 44.12s and 56.64s, and Hessian projection was 6.54× and 8.67× slower than our method, respectively. Symmetric Dirichlet and Symmetric ARAP with Stomakhin et al. (2012) projection took a total of 32.67s and 37.50s, and Hessian projection was 3.47× and 3.69× slower than our method, respectively. For comparisons with the 9×9 and 12×12 eigendecompositions, we also tested the LAPACK `dsyevd` and `dsytrd/dorgtr/dsteqr` routines as implemented in Intel's MKL library, but found them to be roughly 30% slower than Eigen's self-adjoint eigendecomposition.

We further compare against the projection technique from Teran et al. (2005), which requires a model that can be phrased in terms of the **C** invariants. Computing the deformation in Figure 4 with the St. Venant-Kirchhoff (see Appendix F.3) material, which can be expressed in the **C** invariants, we found that our method took a total of 33.95s, while Teran et al. (2005) took 47.12s. Per-quadrature point (9×9) numerical projection took 53.23s, and per-element (12×12) numerical projection took 60.63s. Hessian projection with our method was 2.78× faster than Teran et al. (2005),

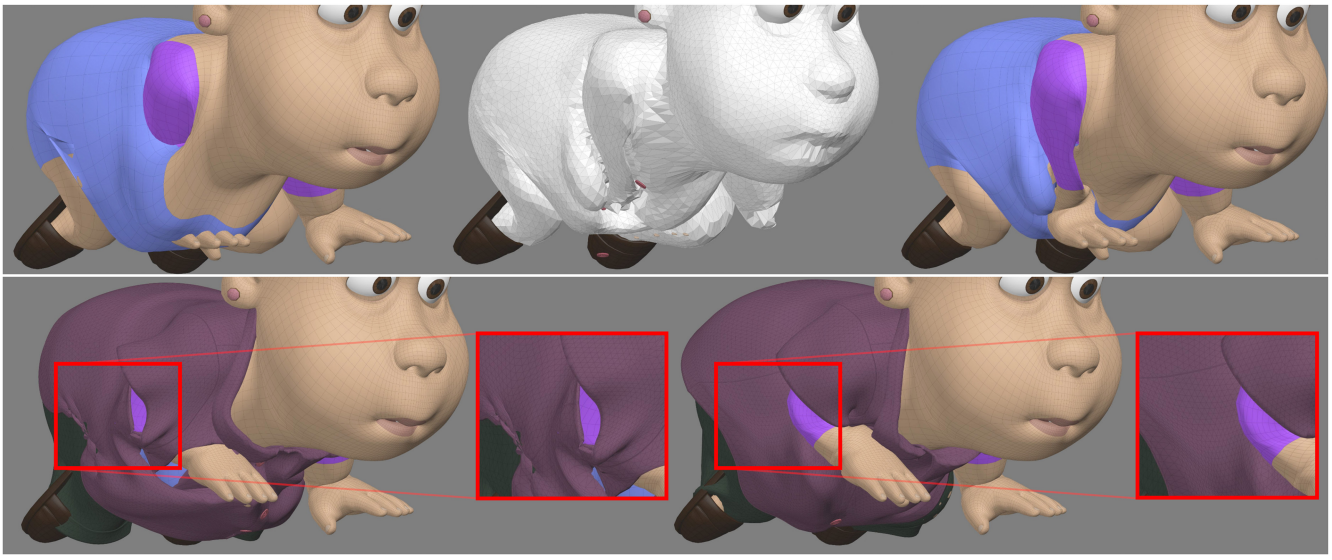

Fig. 8. For the short film *Bao*, intersection-free surfaces were obtained by running a collision-aware volume simulation with our ARAP expressions (Section 6.4). These were used as collision surfaces in a cloth simulator and produced untangled clothing. *Top row:* In the original animation, an arm that is not visible from the camera intersects the torso (left). Volume simulation finds the closest intersection-free state (middle). This state is then sent to the cloth simulator (right). *Bottom row:* Cloth simulation using the original character animation (left). The character's arm penetrates her shirt and artifacts appear along the hip. Simulation results obtained using the intersection-free volume solution are artifact-free (right). ©Disney/Pixar

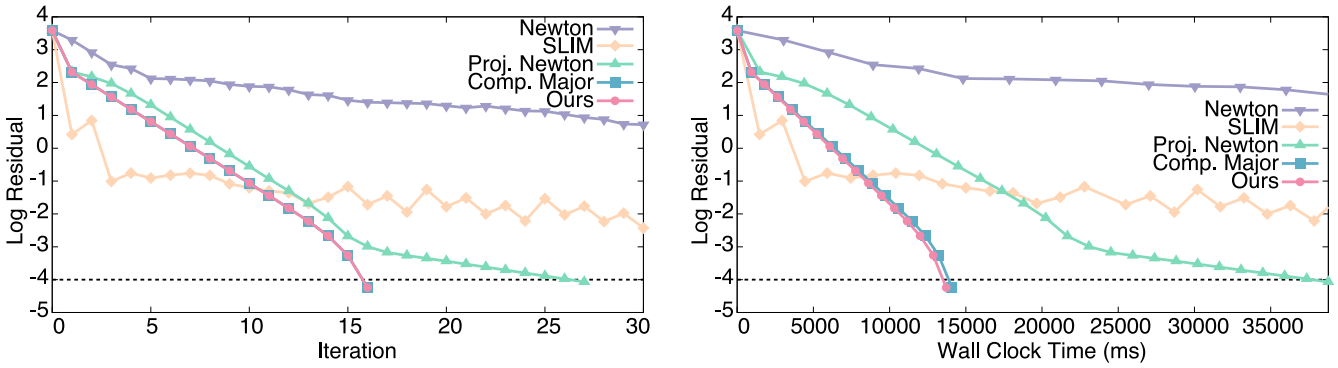

Fig. 9. Solver iterations (left) and wall clock time (right) to compute a parameterization with the Symmetric Dirichlet energy over the bear mesh using our method (pink circles), comp. major. (blue squares), per-element numerical projection (green upright triangles), SLIM (yellow diamonds), and Newton's method without Hessian projection (purple inverted triangle). The termination threshold is denoted by a dashed line.

3.70× faster than per-quadrature point numerical projection, and 4.89× faster than per-element numerical projection.

This example also demonstrates the generality of our approach, since we were able to extend Symmetric Dirichlet and Symmetric ARAP from 2D to 3D with no additional numerical machinery. Furthermore, we performed a live editing session with the Spot mesh using our solver (see our supplemental video). Figure 2 depicts different Yoga poses generated by interactively manipulating handles and then minimizing the Symmetric Dirichlet energy with our method.

6.3 Shape Interpolation in 3D

The ability to robustly optimize 3D deformation energies also leads to an effective method for 3D shape interpolation. Following the framework of Chao et al. (2010), we defined an interpolated energy as the convex combination of two deformation energies, i.e., $\Psi_t = (1-t)\,\Psi_1 + t\Psi_2$. Given two shapes, the deformation energy Ψ_1 was measured against the first shape, while the deformation Ψ_2 was measured against the second shape. The gradient and Hessian of this interpolated energy are simply, respectively, combinations of Ψ_1's and Ψ_2's gradient and Hessian. The sum of two positive-definite Hessians is still positive definite, so by projecting the individual Hessians, we removed any indefiniteness from the sum. To obtain the interpolated shape at some value $t \in [0, 1]$, we minimized Ψ_t for that value of t with our projected Newton solver. We tested this technique by minimizing the Symmetric ARAP energy with two examples. First, we interpolated between two poses of a bar (Figure 7), demonstrating that the solver is robust under bending and twisting deformations. In this example, the bar model had

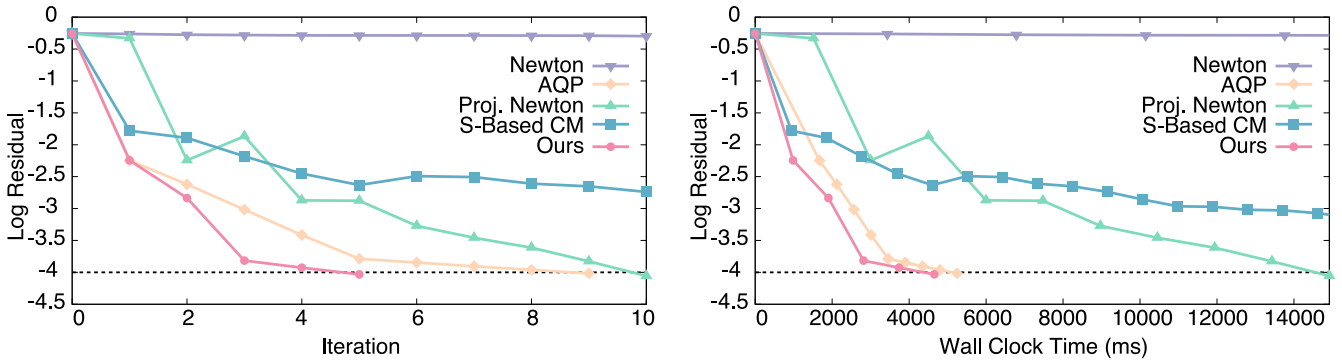

Fig. 10. Solver iterations (left) and wall clock time (right) to compute a parameterization with the ARAP energy over the bear mesh via our method (pink circles), comp. major. with an S-based decomposition (blue squares), per-element numerical projection (green upright triangles), AQP (yellow diamonds), and Newton's method without Hessian projection (purple inverted triangle). The dashed line indicates the termination threshold.

4,528 tetrahedra and performed at 0.029s per frame. Next, we interpolated between two poses of a hand (Figure 5) in order to validate our method with more complex input geometry. In this case, the hand model had 23,692 tetrahedra and took 0.21s per frame. For both sequences, we initialized the solver using the result of the previous frame.

6.4 Intersection-Free Cloth Simulation Inputs

The minimization of volume distortion can also be used as a preprocess to clean up animation artifacts prior to cloth simulation. For the short film *Bao*, the input animations for a character often contained self-intersections that would cause non-physical tangling when a cloth simulation was later performed over its body (Figure 8; bottom-left). These intersections were sufficiently severe that even the robust collision response (Bridson et al. 2002; Harmon et al. 2008) and untangling strategies (Baraff et al. 2003) in our production cloth simulator were unable to produce usable results. These challenging and non-physical configurations are quite common, as animators typically focus their efforts on geometry that is visible from the camera. Artifacts in non-visible regions can still destabilize a simulation, however, requiring time consuming and tedious manual cleanup. Previous attempts at Pixar to remove the self-intersections using surface-only techniques failed to give robust results, but our performant projective Newton solver enabled a volume-based workflow.

First, we generated a tetrahedral mesh for the T-pose of the body surface and warped its elements via a simple Poisson solve to match the surface of the input animation. These warped, intersecting volumetric meshes were then set as the rest configuration $\bar{\mathbf{x}}$ for our optimization at their respective frames. Starting from the tetrahedralized T-pose, we removed intersections by running a collision-aware ARAP volume solve for every frame (Figure 8; top-middle). The intersection-free surfaces produced by these optimized volumes were then used as an input body collider for our cloth simulator (Figure 8; top-right). The volumetric mesh contained 429,871 tetrahedra, and each frame of animation was processed in 4.59 seconds, on average. The workflow was sufficiently robust that it successfully produced artifact-free results for every clothed character in the short on the very first try. Our supplemental video displays an example of this application, and further details can be found in Wong et al. (2018).

6.5 Adding Dynamics

Distortion energies can be incorporated into volume simulations with dynamics. In this case, a minor modification is applied to the projected Newton solver so that mass and damping matrices are added to the global Hessian matrix in order to control inertia and timestepping. Moreover, the solver must often resolve inverted elements induced by input animations. Our approach is especially suited to this scenario, since our closed-form expressions are valid for any value of deformation gradient, even under inversion. Figure 1 shows a sequence from *Cars 3* that was computed by integrating our ARAP minimization into a production volume simulator. Our expressions were also used to simulate the character Jack-Jack in *Incredibles 2*. Our production volume simulator is a semi-implicit solver (Baraff and Witkin 1998) that uses a BDF2 time discretization (Ascher and Petzold 1998), combined with continuous collision detection (e.g., see Tang et al. (2010)), and with impulse-based and penalty-based collision resolution (Bridson et al. 2002).

7 CONCLUSIONS AND FUTURE WORK

We have presented a systematic method for obtaining the analytic, per-quadrature point eigensystem of any isotropic distortion energy. The most immediate application of these results is to recent Domain Specific Languages (DSLs) for physical simulation, such as Ebb (Bernstein et al. 2016) and Simit (Kjolstad et al. 2016). The hyperelastic strain energies used in physical simulations are functionally identical to geometric distortion energies. However, existing DSLs have only been able to support a limited number of these energies, as their gradients and Hessians are usually hand-derived and manually implemented on a case-by-case basis.

Although automatic differentiation methods are available, they result in slow and tortuous codes that are unable to discover structures like our closed-form rotation gradients (Equation (12)). Thus, no existing method has been able to automatically convert an arbitrary isotropic energy into efficient, parsimonious, projected Newton code. The expressions we have presented accomplish exactly this task. A user should now be able to type in any arbitrary isotropic strain energy, and moments later experiment with the behavior of that energy. This will enable qualitative exploration of the space of hyperelastic and geometric distortion energies in a way that was not previously possible, and enable experimentation with more sophisticated energies.

A better understanding of the underlying eigensystems also have potential implications in model reduction. Many such reduction methods use the intrinsic eigensystems of the underlying models to both construct reduced-order basis functions (An et al. 2008; von Tycowicz et al. 2013) and compute reduced-order gradients and Hessians (Barbič and James 2005). In particular, exact reduced-order derivatives have only ever been discovered for the St. Venant-Kirchhoff energy (Barbič and James 2005), which is a 4th -order polynomial. However, many of the distortion energies we examined are lower-order polynomials, so with our S-centric invariants and their eigensystems in hand, it may be possible to derive other closed-form, reduced-order expressions.

Finally, the eigensystems we derived have implications for multigrid solvers (McAdams et al. 2011). Our improved understanding of the spectral behavior at each quadrature point may be able to inform the design of better restriction and prolongation operators.

APPENDIXES

A GENERALITY OF S INVARIANTS

We show that tensor invariants can be written in terms of $\mathbf{S}$ instead of $\mathbf{C}$ without loss of generality. Denoting the $\mathbf{C}$ invariants as $I_\mathbf{C} = \mathrm{tr}(\mathbf{C})$, $II_\mathbf{C} = \|\mathbf{C}\|^2$, and $III_\mathbf{C} = \det(\mathbf{C})$, it is straightforward to verify that:

$$\begin{aligned} I_\mathbf{C} &= I_2 \\ II_{\mathbf{C},2\mathrm{D}} &= I_2^2 - 2I_3^2 \\ II_{\mathbf{C},3\mathrm{D}} &= \frac{1}{2}I_2^2 - \frac{1}{2}I_1^4 + I_1^2 I_2 + 4I_1 I_3 \\ III_\mathbf{C} &= I_3^2. \end{aligned}$$

Any isotropic energy written in terms of the $\mathbf{C}$ invariants can be re-written with the above substitutions. The converse does not hold, because the $\mathbf{C} = \mathbf{F}^\top\mathbf{F}$ squaring discards the sign information from $\mathbf{F}$, and precludes negative values of I_1 and I_3.

B ORTHOGONALITY OF TWIST AND FLIP MATRICES

Lemma B.1. *The twist* $\mathbf{t}$ *and flip* $\mathbf{l}$ *vectors are orthogonal to the gradient of any* $\mathbf{S}$ *invariant in 2D and 3D, i.e.,* $(\partial I_i/\partial \mathbf{f})^\top \mathbf{t} = (\partial I_i/\partial \mathbf{f})^\top \mathbf{l} = 0$.

Proof. We first note that the matrix versions of the invariant gradients in Equations (10), (15), and (16) correspond to matrices of the form $\mathbf{U}\mathbf{X}_i\mathbf{V}^\top$ with diagonal matrices $\mathbf{X}_i$. Since the middle term of the twist matrix in 2D has a zero diagonal (Equation (11)), we have:

$$\begin{aligned} \left(\frac{\partial I_1}{\partial \mathbf{f}}\right)^\top \mathbf{t} = \mathbf{r}^\top\mathbf{t} = \mathbf{R} : \mathbf{T} &= \frac{1}{\sqrt{2}}\,\mathrm{tr}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right) = 0, \\ \left(\frac{\partial I_2}{\partial \mathbf{f}}\right)^\top \mathbf{t} = 2\mathbf{f}^\top\mathbf{t} = 2\mathbf{F} : \mathbf{T} &= \frac{2}{\sqrt{2}}\,\mathrm{tr}\left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right) = 0, \\ \left(\frac{\partial I_3}{\partial \mathbf{f}}\right)^\top \mathbf{t} = \mathbf{g}^\top\mathbf{t} = \mathbf{G} : \mathbf{T} &= \frac{1}{\sqrt{2}}\,\mathrm{tr}\left(\begin{bmatrix} \sigma_2 & 0 \\ 0 & \sigma_1 \end{bmatrix}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right) = 0. \end{aligned}$$

Similarly, the flip matrix has a zero diagonal and, thus, it is orthogonal to the invariant gradients. The same proof holds in 3D trivially. □

C 2D ARAP HESSIAN IMPLEMENTATION

The following 2D Hessian code is implemented using Eigen (Guennebaud et al. 2010). We assume that an SVD implementation that moves all reflections into Σ and a vec function are both available.

```
using namespace Eigen;
Matrix4d ARAP::Hessian(const Matrix2d& F) {
  // Get the SVD of F
  Matrix2d U, Sigma, V;
  SVD(F, U, Sigma, V);

  // Build the eigenvector
  Matrix2d twist;
  twist >> 0, 1, -1, 0;
  twist *= 1.0 / sqrt(2.0);
  const Vector4d e = vec(U * twist * V.transpose());

  // Filter the eigenvalue to guarantee SPD-ness
  const double I_1 = Sigma.trace();
  const double filtered = (I_1 >= 2.0) ? 2.0 / I_1 :
  1.0;

  // Build the final Hessian
  Matrix4d H;
  H.setIdentity();
  H -= filtered * (e * e.transpose());
  H *= 2.0;
  return H;
}
```

D SCALING MODE MATRIX

The general matrix $\mathbf{A}$ that must be solved in 2D to obtain the final two eigenpairs is given by the entries:

$$\begin{aligned} a_{ij}^{2\mathrm{D}} = &\frac{\partial \Psi_q}{\partial I_3} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4I_3\frac{\partial^2 \Psi_q}{\partial I_2^2} + I_3\frac{\partial^2 \Psi_q}{\partial I_3^2} \\ &+ 2I_1\frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 2I_2\frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + I_1\frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1} \end{aligned}$$

$$\begin{aligned} a_{ii}^{2\mathrm{D}} = &2\frac{\partial \Psi_q}{\partial I_2} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_i^2\frac{\partial^2 \Psi_q}{\partial I_2^2} + \sigma_j^2\frac{\partial^2 \Psi_q}{\partial I_3^2} \\ &+ 4\sigma_i\frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 4I_3\frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 2\sigma_j\frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1}. \end{aligned}$$

In 3D, the matrix entries are:

$$\begin{aligned} a_{ij}^{3\mathrm{D}} = &\sigma_k\frac{\partial \Psi_q}{\partial I_3} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_i\sigma_j\frac{\partial^2 \Psi_q}{\partial I_2^2} + \sigma_k I_3\frac{\partial^2 \Psi_q}{\partial I_3^2} + 2\left(I_1 - \sigma_k\right)\frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} \\ &+ 2\sigma_k\left(I_2 - \sigma_k^2\right)\frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + \sigma_k\left(I_1 - \sigma_k\right)\frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1} \end{aligned}$$

$$\begin{aligned} a_{ii}^{3\mathrm{D}} = &2\frac{\partial \Psi_q}{\partial I_2} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_i^2\frac{\partial^2 \Psi_q}{\partial I_2^2} + \sigma_j^2\sigma_k^2\frac{\partial^2 \Psi_q}{\partial I_3^2} \\ &+ 4\sigma_i\frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 4I_3\frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 2\frac{I_3}{\sigma_i}\frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1}. \end{aligned}$$

The two equations above can be further simplified using $\sigma_i\sigma_j = I_3/\sigma_k$. We found that this substitution can sometimes help during implementation, as each entry then only depends on one singular value in addition to the three invariants.

E DECOUPLED 3D SCALING EIGENVALUES

In many cases, the 3D scaling modes become decoupled, in which case, their eigenvalues can be written down explicitly. The expression for $\lambda_{\text{x-scale}}$ is given in Equation (27), and the other two eigenvalues are:

$$\frac{\partial^2\Psi_q}{\partial \mathbf{f}^2}\mathbf{d}_2 = \left[2\frac{\partial\Psi_q}{\partial I_2} + \frac{\partial^2\Psi_q}{\partial I_1^2} + 4\sigma_2^2\frac{\partial^2\Psi_q}{\partial I_2^2} + \frac{I_3^2}{\sigma_2^2}\frac{\partial^2\Psi_q}{\partial I_3^2} + 4\sigma_2\frac{\partial^2\Psi_q}{\partial I_1\partial I_2} + 4I_3\frac{\partial^2\Psi_q}{\partial I_2\partial I_3} + 2\frac{I_3}{\sigma_2}\frac{\partial^2\Psi_q}{\partial I_3\partial I_1}\right]\mathbf{d}_2 = \lambda_{\text{y-scale}}\mathbf{d}_2$$

$$\frac{\partial^2\Psi_q}{\partial \mathbf{f}^2}\mathbf{d}_3 = \left[2\frac{\partial\Psi_q}{\partial I_2} + \frac{\partial^2\Psi_q}{\partial I_1^2} + 4\sigma_3^2\frac{\partial^2\Psi_q}{\partial I_2^2} + \frac{I_3^2}{\sigma_3^2}\frac{\partial^2\Psi_q}{\partial I_3^2} + 4\sigma_3\frac{\partial^2\Psi_q}{\partial I_1\partial I_2} + 4I_3\frac{\partial^2\Psi_q}{\partial I_2\partial I_3} + 2\frac{I_3}{\sigma_3}\frac{\partial^2\Psi_q}{\partial I_3\partial I_1}\right]\mathbf{d}_3 = \lambda_{\text{z-scale}}\mathbf{d}_3.$$

F ADDITIONAL ENERGY EIGENSYSTEMS

We provide the closed-form expressions for the eigensystems of extended versions of the distortion energies from Section 5.

F.1 Co-Rotational Energy

The Co-rotational energy (see, e.g., McAdams et al. (2011)) is defined as $\Psi_{\text{CR}} = \mu\|\mathbf{F}-\mathbf{R}\|^2 + \frac{\kappa}{2}\text{tr}^2(\mathbf{S}-\mathbf{I})$, where κ is a volume penalty. It can be written using the $\mathbf{S}$ invariants as $\Psi_{\text{CR}} = \mu(I_2 - 2I_1 + d^2) + \frac{\kappa}{2}(I_1^2 - 2dI_1 + d^2)$. In 2D, the eigensystem becomes:

$$\begin{aligned}
\lambda_1^{\text{2D}} &= 2\mu + 2\kappa(I_1 - 2 - 2\mu)/I_1 && \mathbf{e}_1 = \mathbf{t}\\
\lambda_2^{\text{2D}} &= 2\mu + 2\kappa && \mathbf{e}_2 = \mathbf{r}/\sqrt{2}\\
\lambda_{3,4}^{\text{2D}} &= 2\mu && \mathbf{e}_{3,4} = \{\mathbf{p},\mathbf{l}\}.
\end{aligned}$$

In 3D, these expressions become:

$$\begin{aligned}
\lambda_1^{\text{3D}} &= 2\mu + 2\kappa(I_1 - 3 - 2\mu)/(\sigma_2+\sigma_3) && \mathbf{e}_1 = \mathbf{t}_1\\
\lambda_2^{\text{3D}} &= 2\mu + 2\kappa(I_1 - 3 - 2\mu)/(\sigma_3+\sigma_1) && \mathbf{e}_2 = \mathbf{t}_2\\
\lambda_3^{\text{3D}} &= 2\mu + 2\kappa(I_1 - 3 - 2\mu)/(\sigma_1+\sigma_2) && \mathbf{e}_3 = \mathbf{t}_3\\
\lambda_4^{\text{3D}} &= 2\mu + 3\kappa && \mathbf{e}_4 = \mathbf{r}/\sqrt{3}\\
\lambda_{5\ldots9}^{\text{3D}} &= 2\mu && \mathbf{e}_{5\ldots9} = \text{5D subspace}.
\end{aligned}$$

Aside from the scaling by μ, the additions to the ARAP energy are most visible in the $\mathbf{t}_i$ eigenvalues, where a new $\kappa(I_1-3)$ term appears. Rotation also becomes a unique eigenvector. Similar to ARAP, there is no need to explicitly construct Co-rotational's 5D subspace in practice. See Appendix C.

F.2 Symmetric ARAP

The Symmetric ARAP energy (Shtengel et al. 2017) is defined as $\Psi_{\text{SARAP}} = \mu/2(\|\mathbf{F}-\mathbf{R}\|^2 + \|\mathbf{F}^{-1}-\mathbf{R}^{-1}\|^2)$. Using the $\mathbf{S}$ invariants, this energy can be written in 2D as:

$$\Psi_{\text{SARAP}}^{\text{2D}} = \frac{\mu}{2}I_2 - \mu I_1 + \frac{\mu}{2}\frac{I_2}{I_3^2} - \mu\frac{I_1}{I_3} + \mu d^2.$$

The $2I_1$ and $2I_1/I_3$ terms are new relative to the 2D Symmetric Dirichlet. The 2D eigensystem is:

$$\begin{aligned}
\lambda_i^{\text{2D}} &= \mu\left(1 - 2/\sigma_i^3 + 3/\sigma_i^4\right) && \mathbf{e}_i = \mathbf{d}_i\\
\lambda_3^{\text{2D}} &= \mu + \mu\left(I_2/I_3 - I_1 + 1\right)/I_3^2 && \mathbf{e}_3 = \mathbf{l}\\
\lambda_4^{\text{2D}} &= \mu + \mu\left((I_2+I_1)/I_3^2 - I_2I_1/I_3^3 - 2\right)/I_1 && \mathbf{e}_4 = \mathbf{t}.
\end{aligned}$$

The 3D version energy is expressed in terms of $\mathbf{S}$ invariants as:

$$\Psi_{\text{SARAP}}^{\text{3D}} = \frac{\mu}{2}I_2 - \mu I_1 + \frac{\mu}{8}\left(\frac{I_1^2 - I_2}{I_3}\right)^2 - \mu\frac{I_1}{I_3} - \frac{\mu}{2}\left(\frac{I_1^2 - I_2}{I_3}\right) + \mu d^2.$$

The eigensystem contains equivalent scaling modes:

$$\lambda_i^{\text{3D}} = \mu\left(1 - 2/\sigma_i^3 + 3/\sigma_i^4\right) \qquad \mathbf{e}_i = \mathbf{d}_i.$$

Its twist eigenpairs are:

$$\lambda_{i+3}^{\text{3D}} = \mu + \frac{\mu}{\sigma_j+\sigma_k}\left[\frac{1}{\sigma_j^2} + \frac{1}{\sigma_k^2} - \frac{1}{\sigma_j^3} - \frac{1}{\sigma_k^3} - 2\right] \qquad \mathbf{e}_{i+3} = \mathbf{t}_i,$$

which are defined over the (i,j,k) triplets $(1,2,3)$, $(2,3,1)$, and $(3,1,2)$. Finally, the flip eigenpairs are:

$$\lambda_{i+6}^{\text{3D}} = \mu\left[1 + \frac{1}{(\sigma_j\sigma_k)^2} + \frac{\sigma_j^2+\sigma_k^2}{(\sigma_j\sigma_k)^3} - \frac{1}{\sigma_j^2\sigma_k} - \frac{1}{\sigma_j\sigma_k^2}\right] \qquad \mathbf{e}_{i+6} = \mathbf{l}_i.$$

F.3 St. Venant-Kirchhoff

The St. Venant-Kirchhoff energy is defined as $\Psi_{\text{StVK}} = \mu\|\mathbf{E}\| + \frac{\kappa}{2}\text{tr}^2(\mathbf{E})$. Using the $\mathbf{S}$ invariants, this energy can be written in 3D as:

$$\Psi_{\text{StVK}}^{\text{3D}} = \frac{\kappa}{8}(I_2-3)^2 + \frac{\mu}{8}\left(8I_1I_3 + I_2^2 + 2I_1^2I_2 - 4I_2 - I_1^4 + 6\right).$$

The 3D twist and flip components of the eigensystem are:

$$\begin{aligned}
\lambda_i^{\text{3D}} &= -\mu + \frac{\kappa}{2}(I_2-3) + \mu\left(\sigma_j^2 + \sigma_k^2 - \sigma_j\sigma_k\right) && \mathbf{e}_i = \mathbf{t}_i,\\
\lambda_{i+3}^{\text{3D}} &= -\mu + \frac{\kappa}{2}(I_2-3) + \mu\left(\sigma_j^2 + \sigma_k^2 + \sigma_j\sigma_k\right) && \mathbf{e}_{i+3} = \mathbf{l}_i.
\end{aligned}$$

The remaining three components are given by the eigensystem of the 3×3 matrix $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$ with entries:

$$\begin{aligned}
a_{ii}^{\text{3D}} &= -\mu + \frac{\kappa}{2}(I_2-3) + (\kappa+3\mu)\sigma_i^2,\\
a_{ij}^{\text{3D}} &= \kappa\sigma_i\sigma_j.
\end{aligned}$$

Given the eigenvalues Λ_i and eigenvectors $\mathbf{Q}_i$ of $\mathbf{A}$, we complete St. Venant-Kirchhoff's eigensystem with:

$$\lambda_{i+6}^{\text{3D}} = \Lambda_i \qquad \mathbf{e}_{i+6} = \text{vec}\left(\mathbf{U}\,\text{diag}\left(\mathbf{Q}_i\right)\mathbf{V}^{\mathbf{T}}\right).$$

Observe that, like the energy itself, St. Venant-Kirchhoff's eigensystem is singularity free.

ACKNOWLEDGMENTS

We would like to thank David Eberle and Audrey Wong for the intersection-free simulation tests and Henry Garcia for running the inflating tire test.

REFERENCES

M. Alexa, D. Cohen-Or, and D. Levin. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of SIGGRAPH*. 157–164.

S. S. An, T. Kim, and D. L. James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27, 5 (2008).

Uri M. Ascher and Linda R. Petzold. 1998. *Computer Methods for Ordinary Differential Equations and Differential-algebraic Equations*. Vol. 61. SIAM.

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*. 43–54.

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling cloth. *ACM Trans. Graph.* 22, 3 (2003).

J. Barbič and Doug L. James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (2005), 982–990.

J. Barbič and Y. Zhao. 2011. Real-time large-deformation substructuring. *ACM Trans. Graph.* 30, 4 (2011).

G. L. Bernstein, C. Shah, C. Lemire, Z. Devito, M. Fisher, P. Levis, and P. Hanrahan. 2016. Ebb: A DSL for physical simulation on CPUs and GPUs. *ACM Trans. Graph.* 35, 2 (2016).

D. Bommes, H. Zimmer, and L. Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.* 28, 3 (2009).

J. Bonet and R. D. Wood. 2008. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. 2010. *Polygon Mesh Processing*. AK Peters.

S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly. 2012. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum* 31, 5 (2012), 1657–1667.

S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4 (2014).

R. Bridson, R. Fedkiw, and J. Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Trans. Graph.* 21, 3 (2002), 594–603.

I. Chao, U. Pinkall, P. Sanan, and P. Schröder. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph.* 29, 4 (2010).

R. Chen and O. Weber. 2017. GPU-accelerated locally injective shape deformation. *ACM Trans. Graph.* 36, 6 (2017).

S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. 2017. Isometry-aware preconditioning for mesh parameterization. *Comp. Graphics. Forum* 36, 5 (2017), 37–47.

M. Eigensatz and M. Pauly. 2009. Positional, metric, and curvature control for constraint-based surface deformation. *Comput. Graph. Forum* 28, 2 (2009), 551–558.

X.-M. Fu and Y. Liu. 2016. Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph.* 35, 6 (2016).

X.-M. Fu, Y. Liu, and B. Guo. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph.* 34, 4 (2015).

G. H. Golub and C. F. Van Loan. 2012. *Matrix Computations*. Vol. 3. JHU Press.

G. Guennebaud, B. Jacob, et al. 2010. Eigen v3. Retrieved from http://eigen.tuxfamily.org.

D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. 2008. Robust treatment of simultaneous collisions. *ACM Trans. Graph.* 27, 3 (2008), 23:1–23:4.

K. Horman and G. Greiner. 1999. MIPS: An efficient global parameterization method. In *Curve and Surface Design*. 153–162.

Intel. 2018. Math Kernel Library. Retrieved from https://software.intel.com/en-us/mkl.

G. Irving, J. Teran, and R. Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *SIGGRAPH/Eurog. Symp. on Comp. Anim.* 131–140.

A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine. 2012. Fast automatic skinning transformations. *ACM Trans. on Graphics* 31, 4 (2012).

F. Kjolstad, S. Kamil, J. Ragan-Kelley, D. I. W. Levin, S. Sueda, D. Chen, E. Vouga, D. M. Kaufman, G. Kanwar, W. Matusik, and S. Amarasinghe. 2016. Simit: A language for physical simulation. *ACM Trans. Graph.* 35, 2 (2016).

T. G. Kolda and B. W. Bader. 2009. Tensor decompositions and applications. *SIAM Rev.* 51, 3 (2009), 455–500.

S. Z. Kovalsky, M. Galun, and Y. Lipman. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.* 35, 4 (2016).

L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. 2008. A local/global approach to mesh parameterization. *Computer Graphics Forum* 27, 5 (2008), 1495–1504.

T. Liu, S. Bouaziz, and L. Kavan. 2017. Quasi-Newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph.* 36, 3 (2017).

J. E Marsden and T. JR Hughes. 1994. *Mathematical Foundations of Elasticity*. Dover Publications.

A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.* 30, 4 (2011).

R. Narain, M. Overby, and G. E. Brown. 2016. ADMM ⊇ projective dynamics: Fast simulation of general constitutive models. In *Proc. of the ACM SIGGRAPH/Eurog. Symp. on Comp. Anim.* 21–28.

J. Nocedal and S. J. Wright. 2006. *Numerical Optimization*. Springer.

T. Papadopoulo and M. I. A. Lourakis. 2000. *Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications*. Springer, 554–570.

F. Pighin and J. P. Lewis. 2007. Practical least-squares for computer graphics. In *ACM SIGGRAPH Courses*. 1–57.

M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Trans. Graph.* 36, 2 (2017).

A. Shtengel, R. Poranne, O. Sorkine-Hornung, S. Z. Kovalsky, and Y. Lipman. 2017. Geometric optimization via composite majorization. *ACM Trans. Graph.* 36, 4 (2017).

E. Sifakis and J. Barbic. 2012. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*.

B. Smith, F. de Goes, and T. Kim. 2018. Stable Neo-Hookean flesh simulation. *ACM Trans. Graph.* 37, 2 (2018).

J. Smith and S. Schaefer. 2015. Bijective parameterization with free boundaries. *ACM Trans. Graph.* 34, 4 (2015).

O. Sorkine and M. Alexa. 2007. As-rigid-as-possible surface modeling. In *Eurog. Symposium on Geometry Processing*, Vol. 4.

A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran. 2012. Energetically consistent invertible elasticity. In *ACM SIGGRAPH/Eurog. Symp. Comp. Anim.* 25–32.

M. Tang, D. Manocha, and R. Tong. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of I3D*. ACM, 7–13.

J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *ACM SIGGRAPH/Eurog. Symp. on Comp. Anim.* 181–190.

C. D. Twigg and Z. Kačić-Alesić. 2010. Point cloud glue: Constraining simulations using the procrustes transform. In *ACM SIGGRAPH/Eurog. Symp. on Comp. Anim.* 45–54.

C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt. 2013. An efficient construction of reduced deformable objects. *ACM Trans. Graph.* 32, 6 (2013).

H. Wang. 2015. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.* 34, 6 (2015).

Audrey Wong, David Eberle, and Theodore Kim. 2018. Clean cloth inputs: Removing character self-intersections with volume simulation. In *ACM SIGGRAPH Talks*. Article 42, 2 pages.

H. Xu, F. Sin, Y. Zhu, and J. Barbič. 2015. Nonlinear material design using principal stretches. *ACM Trans. Graph.* 34, 4 (2015).

Received May 2018; revised September 2018; accepted September 2018