Efficient elasticity for character skinning with contact and collisions

Aleka McAdams[1,3] Yongning Zhu[2] Andrew Selle[1] Mark Empey[1]
Rasmus Tamstorf[1] Joseph Teran[3,1] Eftychios Sifakis[4,1]

[1] Walt Disney Animation Studios [2] PDI/DreamWorks
[3] University of California, Los Angeles [4] University of Wisconsin, Madison

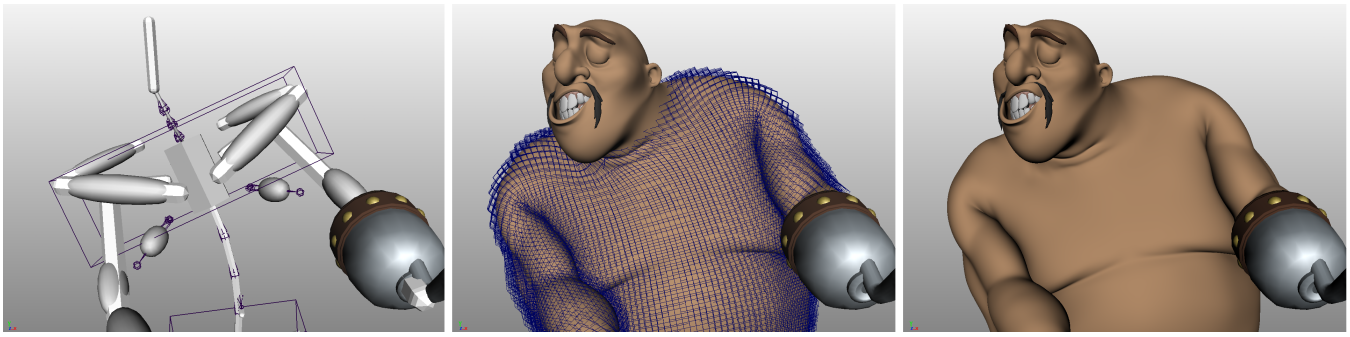


Figure 1: *Our method takes a geometric internal skeleton (left) and a source surface mesh (not pictured) as input. Based on a hexahedral lattice (center) it then simulates a deformed surface (right) obeying self-collision and volumetric elasticity. The example shown here has 106,567 cells and simulates at 5.5 seconds per frame. ©Disney Enterprises, Inc.*

Abstract

We present a new algorithm for near-interactive simulation of skeleton driven, high resolution elasticity models. Our methodology is used for soft tissue deformation in character animation. The algorithm is based on a novel discretization of corotational elasticity over a hexahedral lattice. Within this framework we enforce positive definiteness of the stiffness matrix to allow efficient quasistatics and dynamics. In addition, we present a multigrid method that converges with very high efficiency. Our design targets performance through parallelism using a fully vectorized and branch-free SVD algorithm as well as a stable one-point quadrature scheme. Since body collisions, self collisions and soft-constraints are necessary for real-world examples, we present a simple framework for enforcing them. The whole approach is demonstrated in an end-to-end production-level character skinning system.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: skinning, corotated elasticity, physics-based modeling, elastic deformations

Links: DL PDF WEB

1 Introduction

Creating appealing characters is essential for feature animation. One challenging aspect is the production of life-like deformations for soft tissues comprising both humans and animals. In order to provide the necessary control and performance for an animator, such deformations are typically computed using a skinning technique and/or an example based interpolation method. Meanwhile, physical simulation of flesh-like material is usually avoided or relegated to an offline process due to its high computational cost. However, simulations create a range of very desirable effects, like squash-and-stretch and contact deformations. The latter is especially important as it can guarantee pinch-free geometry, which is important for subsequent simulations like cloth and hair.

Although the benefits of solving the equations of the underlying physical laws for character deformation are clear, computational methods are traditionally far too slow to accommodate the rapid interaction demanded by animators. Many simplified approaches to physical simulation can satisfy interactivity demands, but any such approach must provide all of the following functionality to be useful in production: (1) robustness to large deformation, (2) support for high-resolution geometric detail, (3) fast and accurate collision response (both self and external objects). Ideally, for rigging, it should also provide path independent deformations determined completely by a kinematic skeleton. However, this is not possible since contact deformations in general depend on the path taken to the colliding state.

Whereas previous works have addressed many of these concerns individually, e.g., robustness to large deformation in [Irving et al. 2004], high resolution detail [Zhu et al. 2010], and quasistatic simulation [Teran et al. 2005], we present a novel algorithmic framework for the simulation of hyperelastic soft tissues that targets all aspects discussed above. Our approach is robust to large deformation (even inverted configurations) and extremely stable by virtue of careful treatment of linearization. We present a new multigrid approach to efficiently support hundreds of thousands of degrees of freedom (rather than the few thousands typical of existing techniques) in a production environment. Furthermore, these performance and robustness improvements are guaranteed in the presence of both colli-

sion and quasistatic/implicit time stepping techniques. We demonstrate the impact of these advances in a complete production system for physics-based skinning of skeletally driven characters.

2 Related work

Skeleton driven skin deformation was first introduced by [Magnenat-Thalmann et al. 1988]. Since then such techniques have been used extensively, especially the "linear blend skinning" technique (aka. "skeleton subspace deformation" (SSD) or "enveloping"). However, the limitations of such techniques are well-known and have been the topic of numerous papers [Wang and Phillips 2002; Merry et al. 2006; Kavan et al. 2008]. Despite improvements, skinning remains, for the most part purely kinematic. It has proven very difficult to get more accurate, physically based deformations (e.g., from self-collisions and contact with other objects). Instead, such phenomena are typically created through a variety of example based approaches [Lewis et al. 2000; Sloan et al. 2001]. Although example based methods are computationally cheap, they often require extreme amounts of user input, especially for contact and collision. Recently, authors have also considered automatic means of fitting skeletons and binding their movement to deformation as in [Baran and Popović 2007].

Simulation recently enabled significant advances to character realism in [Irving et al. 2008] and [Clutterbuck and Jacobs 2010], albeit with the luxury of extreme computation time. Nevertheless these approaches demonstrated the promise of simulation. Many techniques reduce the accuracy of the elasticity model to help improve performance and interactivity. [Terzopoulos and Waters 1990; Chadwick et al. 1989] first demonstrated the effectiveness of comparatively simple mass/spring based approaches. [Sueda et al. 2008] add interesting anatomic detail using the tendons and bones in the hand, but use simple surface-based skin. [Shi et al. 2008] use simplified surface-based spring forces to provide dynamics given an example skeleton and deformed poses. [Kry et al. 2002] use principle component analysis of off-line elasticity simulation to provide interactive physically based SSD. [Capell et al. 2005; Capell et al. 2002; Galopo et al. 2007] used a skeleton based local rotational model of simple linear elasticity. [Müller et al. 2005] introduced shape matching, a technique that uses quadratic modal elements defined per lattice cell, allowing realtime albeit less accurate deformations. [Rivers and James 2007] extended the accuracy of this method while maintaining high performance with a fast SVD. Warped stiffness approaches [Müller et al. 2002; Müller and Gross 2004; Müller et al. 2004] are a more general example of the techniques developed by Cappel et al. and use an inexact force differential to yield easily solvable symmetric positive definite (SPD) linearizations. However, [Chao et al. 2010] recently demonstrated the importance of a more accurate approximation to rotational force differentials lacking in warped stiffness approaches. We illustrate this important robustness limitation of the warped stiffness approximation in figure 2 (see also section 5). The instability of the method hinders its use in skinning applications. Unfortunately, the more accurate linearizations yield indefinite systems and thus require more expensive linear algebra techniques (e.g., GMRES). In the present work, we demonstrate a solution procedure that rivals the efficiency of warped stiffness, but without the robustness difficulties.

Typically elastic simulation requires the solution of large sparse systems. Conjugate gradients is a popular method for solving such systems by virtue of simplicity and low-memory overhead; however, it is plagued by slow convergence (especially with high resolution models). Multigrid techniques potentially avoid these convergence issues, but can be costly to derive for problems over irregular domains. [Zhu et al. 2010] developed a multigrid approach that achieves nearly optimal convergence properties for incompressible materials on irregular domains. However, their technique for corotational elasticity uses a pseudo-Newton iteration that does not guarantee convergence on the large deformations typical in skeleton driven animation. [Dick et al. 2011; Georgii and Westermann 2006; Wu and Tendick 2004] also examine multigrid methods for rapidly solving the equations of corotational elasticity. However these techniques are based on the warped stiffness approximation to corotational force differentials and demonstrate similar convergence issues as Zhu et al. Multigrid has also been shown to provide excellent parallel performance (e.g., on the GPU in [Dick et al. 2011] and on the CPU in [Zhu et al. 2010]). [Otaduy et al. 2007] consider FAS multigrid methods for solving implicit dynamics on unstructured grids. Our multigrid approach is the first to robustly provide near-interactive performance for non-linear elasticity models with hundreds of thousands of degrees of freedom.

3 Elasticity and discretization

The demand for high-resolution simulation with optimal performance and robustness motivated the development of our novel corotational elasticity discretization. Following [Chao et al. 2010], we use an accurate treatment of force derivatives to yield a more robust solver than the simplified warped-stiffness techniques. We show that these careful linearizations can be done both cheaply and simply and are essential for our desired robustness and efficiency. We perform the discretization over a uniform hexahedral lattice (rather than an unstructured tetrahedral one) to facilitate performance on modern hardware. Although standard methods for hexahedra require 8 point Gauss quadrature per cell for stability, we develop a much more efficient one-point quadrature discretization (section 4). In section 3.2 we begin with a quasistatic method, but we extend our technique to dynamics in section 6.

3.1 Fundamentals

We represent the deformation of a 3D elastic body by a function $\boldsymbol{\phi} : \Omega \rightarrow \mathbf{R}^3$, which maps a material point $\boldsymbol{X}$ to a deformed world-space point $\boldsymbol{x}$ so $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X})$. Subsequently, we use $\boldsymbol{x}$ and $\boldsymbol{\phi}$

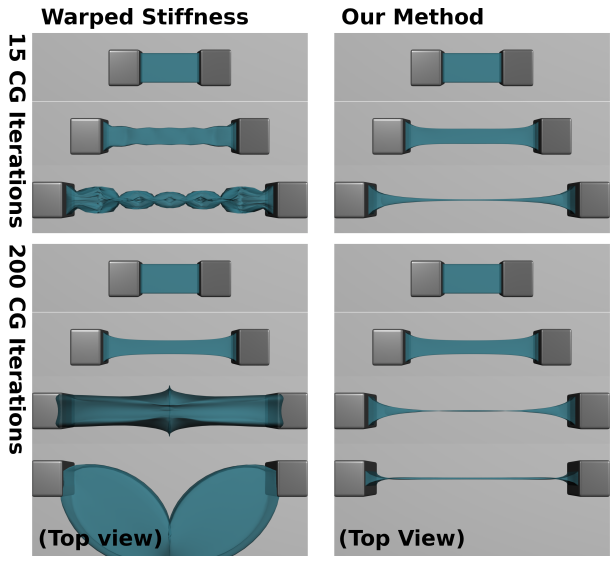


Figure 2: *Inexact methods of computing force differentials lead to instabilities, even with moderate Poisson's ratio (0.3), unlike our method. In this example, instabilities occur at* $2.4\times$ *stretch, which is common near joints in skinning applications.*

interchangeably, i.e. we identify $\boldsymbol{x}(\boldsymbol{X}) \equiv \boldsymbol{\phi}(\boldsymbol{X})$. For hyperelastic materials in general, the response can be computed based on the deformation energy:

$$E = \int_\Omega \Psi(\boldsymbol{X}, \mathbf{F}(\boldsymbol{X}))d\boldsymbol{X} \tag{1}$$

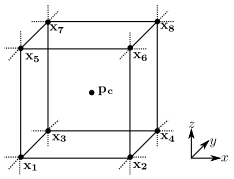
For simplicity we will here consider the energy density Ψ as a function of the deformation gradient $F_{ij} = \partial\phi_i/\partial\boldsymbol{X}_j$. Specifically for corotational elasticity we have

$$\Psi = \mu\|\mathbf{F} - \mathbf{R}\|_F^2 + \frac{\lambda}{2}\text{tr}^2(\mathbf{R}^T\mathbf{F} - \mathbf{I}) \tag{2}$$

where μ, λ are the Lamé coefficients, and $\mathbf{R}$ is the rotation from the polar decomposition $\mathbf{F} = \mathbf{RS}$.

We discretize our model domain Ω into cubic elements Ω_e of step size h so $\Omega = \cup_e\Omega_e$. The degrees of freedom are world space samples of $\boldsymbol{x}_i = \boldsymbol{\phi}(\boldsymbol{X}_i)$. The discrete version of equation (1) then becomes a sum of energies from each element E_e. Using just a single quadrature point for the voxel center $\boldsymbol{p}^c$ gives $E_e \approx h^3\Psi(\mathbf{F}^e)$ where $\mathbf{F}^e \approx \mathbf{F}(\boldsymbol{p}^\mathbf{c})$ is computed with central differences about the cell center from averaged faces. This approximation can be written

$$F_{ij}^e = \sum_k G_{jk}^e x_k^{(i)} \tag{3}$$



where $x_k^{(i)}$ is the i-th component of the three-dimensional vector $\boldsymbol{x}_k$ (see right) and we have a *discrete gradient*

$$\mathbf{G}^e = \frac{1}{4h}\begin{pmatrix} -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Thus, we have a means to compute the total energy in terms only of the nodal world positions of our hexahedral lattice.

3.2 Force and equilibrium

A discrete force per node can in general be written as

$$\boldsymbol{f}_i = -\frac{\partial E}{\partial \boldsymbol{x}_i} = \sum_e\left(-\frac{\partial E_e}{\partial \boldsymbol{x}_i}\right) = \sum_e \boldsymbol{f}_i^e. \tag{4}$$

Using equation (3) and the fact that Ψ is a function of the deformation gradient alone, a concise expression for each of the components of $\boldsymbol{f}_i^e = (f_i^{(1)}, f_i^{(2)}, f_i^{(3)})$ (the force contribution to node i from element e) is:

$$\begin{aligned}\boldsymbol{f}_i^{(j)} &= -\frac{\partial E_e}{\partial \boldsymbol{x}_i^{(j)}} = -V_e\frac{\partial\Psi(\mathbf{F}^e)}{\partial \boldsymbol{x}_i^{(j)}} = -V_e\sum_{k,l}\left.\frac{\partial\Psi}{\partial F_{kl}}\right|_{\mathbf{F}^e}\frac{\partial F_{kl}^e}{\partial \boldsymbol{x}_i^{(j)}} \\ &= -V_e\sum_{k,l}[\mathbf{P}(\mathbf{F}^e)]_{kl}G_{li}^e\delta_{jk} = -V_e\sum_l[\mathbf{P}(\mathbf{F}^e)]_{jl}G_{li}^e \\ &= -V_e[\mathbf{P}(\mathbf{F}^e)\mathbf{G}^e]_{ji}. \end{aligned} \tag{5}$$

where $\mathbf{P}(\mathbf{F}) := \partial\Psi/\partial\mathbf{F}$ is the 1st Piola-Kirchhoff stress tensor. For corotational elasticity, equation (2), we specifically get:

$$\mathbf{P} = \mathbf{R}\left[2\mu(\mathbf{S} - \mathbf{I}) + \lambda tr(\mathbf{S} - \mathbf{I})\right] \tag{6}$$

We note that this expression does not depend on $\partial\mathbf{R}/\partial\mathbf{F}$ due to cancellation, as described in the attached technical document. Combining this with equation 5 we have a matrix that maps the nodal positions of a cell to its force contribution

$$\mathbf{J}^e = \begin{pmatrix} \boldsymbol{f}_1^e & \boldsymbol{f}_2^e & \cdots & \boldsymbol{f}_8^e \end{pmatrix} = -V_e\mathbf{P}(\mathbf{F}^e)\mathbf{G}^e. \tag{7}$$

At each nodal position $\boldsymbol{x} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ we compute the forces $\boldsymbol{f}(\boldsymbol{x}) := (\boldsymbol{f}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{f}_N(\boldsymbol{x}_N))$ and we might additionally have external forces $\mathbf{g}$. For quasistatics we solve the resulting force balance equation $\mathbf{f} + \mathbf{g} = 0$ using Newton-Raphson where the kth iterate requires the solution of the linearized system:

$$\mathbf{K}(\boldsymbol{x}^{(k)})\delta\boldsymbol{x}^{(k)} = \boldsymbol{g} + \boldsymbol{f}(\boldsymbol{x}^{(k)}). \tag{8}$$

Here $\mathbf{K}(\boldsymbol{x}^{(k)}) := -\left.\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}^{(k)}}$ and $\delta\boldsymbol{x}^{(k)} := \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}$.

3.3 Differentials of force and stress

Equation (8) requires solving $\mathbf{K}$ at every iteration. However, forming the matrix explicitly would incur significant performance losses from the 243 non-zero entries needed per node. Instead, we define a procedure that directly determines the product $\mathbf{K}\delta\boldsymbol{x}$ (where $\delta\boldsymbol{x}$ is a displacement), allowing the use of a Krylov solver. The product $\mathbf{K}\delta\boldsymbol{x} = -\delta\boldsymbol{f}$ is the force differential induced by the displacements. Applying differentials on equations (4) and (7) we can write the differential of each nodal force as $\delta\boldsymbol{f}_i = \sum_e \delta\boldsymbol{f}_i^e$, where:

$$\begin{pmatrix} \delta\boldsymbol{f}_1^e & \delta\boldsymbol{f}_2^e & \cdots & \delta\boldsymbol{f}_8^e \end{pmatrix} = -h^3\delta\left[\mathbf{P}(\mathbf{F}^e)\right]\mathbf{G}^e. \tag{9}$$

Taking differentials of $\mathbf{R}^T\mathbf{R}=\mathbf{I}$ we get $(\mathbf{R}^T\delta\mathbf{R})^T + \mathbf{R}^T\delta\mathbf{R} = 0$, thus the matrix $\delta\mathbf{R}^T\mathbf{R}$ is skew symmetric. Consequently $\text{tr}(\delta\mathbf{R}^T\mathbf{F}) = \text{tr}(\delta\mathbf{R}^T\mathbf{RS}) = (\delta\mathbf{R}^T\mathbf{R}) : \mathbf{S} = 0$ as a contraction of a symmetric and a skew symmetric matrix is zero. Using this result, we take differentials on equation (6) to obtain

$$\begin{aligned}\delta\mathbf{P} = \;& 2\mu(\delta\mathbf{F} - \delta\mathbf{R}) + \lambda\left\{\text{tr}(\delta\mathbf{R}^T\mathbf{F}) + \text{tr}(\mathbf{R}^T\delta\mathbf{F})\right\}\mathbf{R} \\ & + \lambda\text{tr}(\mathbf{R}^T\mathbf{F} - \mathbf{I})\delta\mathbf{R} \\ = \;& 2\mu\delta\mathbf{F} + \lambda\text{tr}(\mathbf{R}^T\delta\mathbf{F})\mathbf{R} + \{\lambda\text{tr}(\mathbf{S} - \mathbf{I}) - 2\mu\}\,\delta\mathbf{R}. \end{aligned} \tag{10}$$

The differential $\delta\mathbf{F}$ of the (discrete, cell-centered) deformation gradient is computed from equation (3) according to the formula:

$$\delta F_{ij}^e = \sum_k G_{jk}^e\delta x_k^{(i)}. \tag{11}$$

The differential of rotation $\mathbf{R}$ is given by

$$\delta\mathbf{R} = \mathbf{R}\left[\mathcal{E} : \left((\text{tr}(\mathbf{S})\mathbf{I} - \mathbf{S})^{-1}\left(\mathcal{E}^T : (\mathbf{R}^T\delta\mathbf{F})\right)\right)\right] \tag{12}$$

where $\mathcal{E}$ is the alternating third order tensor which maps a vector to a cross product matrix. Equation (12) is a compact expression of the result presented in [Twigg and Kačić-Alesić 2010], see also the attached technical document for a detailed proof. In summary, for every cell Ω_e, we compute the cell-centered deformation gradient $\mathbf{F}^e$ using equation (3) and compute its polar decomposition. Using equations (10), (11) and (12) we compute $\delta\mathbf{P}$ corresponding to the displacements $\delta\mathbf{x}$. Finally, using equation (9) we compute the contribution of Ω_e to the force differential, and accumulate the computed values onto $\delta\mathbf{f}$.

4 Stabilization

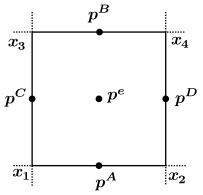
In section 3 we outlined a discretization using a single quadrature point per cell. This choice promises better performance since it requires only one SVD/polar decomposition per cell (rather than 8 with Gauss quadrature). Unfortunately, this leads to catastrophic defects if used without modification. Our method stabilizes the one point quadrature approach and thus improves performance significantly by requiring just one SVD/polar decomposition per cell.

Consider that a cell has 8 points (24 DOFs), but the one-point quadrature based elemental energy only depends on the cell-centered deformation gradient (9 DOFs), leading to a large subspace of deformation modes that have no effect on the discrete energy. If we were fortunate, this "nullspace" might only appear element-by-element and in the union of all elements, these modes would be penalized. Unfortunately, in our case, there exist non-physical global modes that have no effect on discrete energy. For example, consider a red-black ordering of the grid nodes, and assign one constant displacement to red nodes and another to all black ones. This is not seen by the discrete energy and is visible as parasitic "hourglassing" artifacts in force equilibrium (as in figure 3). These oscillatory nullspace modes are more than visual artifacts, they compromise the ellipticity of the discrete equations in multigrid methods. This is why standard discretizations use higher-order quadrature, gaining stability at higher cost.

We remedy this instability by proposing a new integration rule that is stable yet computationally cheap (requires only one polar decomposition per cell). As an initial observation, we have experimentally verified that the term $\mu\|\mathbf{F}-\mathbf{R}\|_F^2$ in equation (2) is the one which primarily determines stability; we observed that if a stable technique (e.g. 8-point Gauss quadrature) is used to integrate this term, the entire scheme will remain stable, even if the naive 1-point quadrature is used for the term $\frac{\lambda}{2}\text{tr}^2(\mathbf{R}^T\mathbf{F}-\mathbf{I})$. Thus, we will initially present our approach in the context of the simpler energy density $\Psi=\mu\|\mathbf{F}-\mathbf{R}\|_F^2$, and address the 2D case first (i.e., on a square lattice).

The use of staggered grids to avoid instability from non-physical modes when using central differencing can be seen in many Eulerian fluid dynamics methods. [Harlow and Welch 1965] introduced the staggered MAC grid for velocities and pressure, and [Gerritsma 1996; Goktekin et al. 2004] extended this to viscoelastic fluids by staggering the second order stress or strain tensors. Similarly, we introduce four additional quadrature points $\boldsymbol{p}^q$, $q\in\{A,B,C,D\}$, located on edge centers of the quadrilateral lattice (see right). We write Ψ as

$$\Psi=\mu\sum_{i,j}(F_{ij}-R_{ij})^2. \tag{13}$$



Our approach will essentially follow a different quadrature rule for every term $(F_{ij}-R_{ij})^2$ in this expression. In particular, instead of using the single quadrature point $\boldsymbol{p}^e$ at the cell center, we will use those locations within the cell (possibly more than one) where F_{ij} is "naturally" defined, as a central difference of just two degrees of freedom. In this way, we avoid the averaging and risk of cancellation associated with expressing all derivatives exclusively at the cell center.

We observe that the x-derivatives F_{11} and F_{21} are naturally defined at the centers $\boldsymbol{p}^A,\boldsymbol{p}^B$ of x-oriented edges, while the y-derivatives F_{12} and F_{22} are naturally located at points $\boldsymbol{p}^C$ and $\boldsymbol{p}^D$. We also evaluate the cell-centered deformation gradient $\mathbf{F}^e$ once more, following exactly equation (3) as before. We compute matrix $\mathbf{R}^e$ from the polar decomposition of $\mathbf{F}^e$, and use the information from this matrix wherever R_{ij} is needed in equation (13). Finally, our proposed quadrature method takes the form:

$$E_e=\frac{\mu h^2}{2}\sum_{i=1}^{2}\left[\sum_{q\in\{A,B\}}(F_{i1}^q-R_{i1}^e)^2+\sum_{q\in\{C,D\}}(F_{i2}^q-R_{i2}^e)^2\right]. \tag{14}$$

We have that $F_{i1}^e=\frac{1}{2}\sum_{q\in\{A,B\}}F_{i1}^q$ and $F_{i2}^e=\frac{1}{2}\sum_{q\in\{C,D\}}F_{i2}^q$ since the entries of $\mathbf{F}^e$ were defined as averaged central differences. Using these identities, equation (14) becomes $E_e=E_1+E_2$, with

$$E_1=\frac{\mu h^2}{2}\sum_{i=1}^{2}\left((F_{i1}^A)^2+(F_{i1}^B)^2+(F_{i2}^C)^2+(F_{i2}^D)^2\right), \text{ and} \tag{15}$$

$$E_2=\mu h^2\left[-2\text{tr}\left(\mathbf{R}^{eT}\mathbf{F}^e\right)+\|\mathbf{I}\|_F^2\right] \tag{16}$$

The energy discretization suggested by equations (15) and (16) is stable, as seen by our results and the convergent multigrid schemes we have constructed on its basis. In order to better explain the mechanics of this approach, we manipulate the μ-component of the energy as follows:

$$\Psi=\mu\|\mathbf{F}-\mathbf{R}\|_F^2=\mu\|\mathbf{F}\|_F^2-2\mu\text{tr}\left(\mathbf{R}^T\mathbf{F}\right)+\mu\|\mathbf{I}\|_F^2.$$

Equation (15) suggests a quadrature rule for the term $\mu\|\mathbf{F}\|_F^2$. The integral $\frac{1}{2}\int\|\mathbf{F}\|_F^2$ is the weak form of the component-wise Laplace operator; thus equation (15) generates an energy discretization for the Laplace operator $2\mu\Delta$. Equation (16) is nothing but a one-point quadrature, but on the term $-2\mu\text{tr}\left(\mathbf{R}^T\mathbf{F}\right)+\mu\|\mathbf{I}\|_F^2$. In fact, at this point we can re-introduce the omitted λ-term of the energy, and write Ψ as:

$$\Psi=\underbrace{\mu\|\mathbf{F}\|_F^2}_{\Psi_\Delta}\underbrace{-2\mu\text{tr}\left(\mathbf{R}^T\mathbf{F}\right)+\mu\|\mathbf{I}\|_F^2+\frac{\lambda}{2}\text{tr}^2(\mathbf{R}^T\mathbf{F}-\mathbf{I})}_{\Psi_{\text{aux}}} \tag{17}$$

We implement this discretization, by separating energy, forces, and force differentials into two components: (a) a term stemming from the Laplace energy density Ψ_Δ, and (b) an auxiliary term originating from Ψ_{aux}, which is integrated with the simple one-point quadrature as in section 3. Note that the forces arising from the Laplace term are purely linear, and the stiffness matrix resulting from the same term is constant (and equal to a Laplace matrix), leading to a minimal implementation overhead, over the standard cost of one-point quadrature for the auxiliary term.

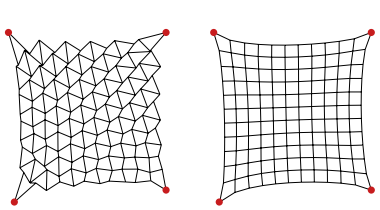


Figure 3: *A 2D elastic patch is stretched by pinning the four corners to target locations. Left: The unmodified one-point quadrature method is riddled by hourglassing instabilities. Right: Our method.*

5 Indefiniteness correction

Symmetry of the stiffness matrix, $\mathbf{K}$, allows the use of certain Krylov methods, but positive definiteness is required for conjugate gradients. While $\mathbf{K}$ will be positive definite close to equilibrium (it is the energy Hessian), in practice Newton-Raphson may generate intermediate indefinite states. Like [Teran et al. 2005], we will modify $\mathbf{K}$ to guarantee definiteness while retaining the same nonlinear solution (though maybe via different iterates).

Similar to [Teran et al. 2005] we will conservatively enforce the definiteness of $\mathbf{K}$ by projecting each elemental stiffness matrix to its positive semi-definite counterpart, i.e. a matrix with the same eigenvectors, but with negative eigenvalues clamped to zero. Naturally, we want to avoid an explicit eigenanalysis of the 24×24

elemental stiffness $\mathbf{K}^e$, and even avoid forming it at altogether. We describe a procedure to perform this semi-definite projection in an inexpensive, matrix-free fashion. We will initially describe the definiteness projection for the simple one-point quadrature rule, defined in section 3. This will be a stepping stone in designing a definiteness fix for our stabilized quadrature rule, described in section 4. The elemental stiffness matrix is positive semi-definite if and only if $0 \leq \delta\boldsymbol{x}^T\mathbf{K}^e\delta\boldsymbol{x} = -\delta\boldsymbol{x}^T\delta\boldsymbol{f}$, where $\delta\boldsymbol{x}, \delta\boldsymbol{f}$ are the stacked nodal position and force differentials for Ω_e. Taking differentials on both sides of equation (5) we get:

$$\delta f_i^{(j)} = -V_e \sum_k \delta P_{jk} G^e_{ki} = -V_e \sum_{k,l,m} T_{jklm}\delta F^e_{lm} G^e_{ki}$$

where $\mathcal{T} = [T_{ijkl}]$ is the fourth order tensor defined as the stress derivative $\mathcal{T} := \partial\mathbf{P}/\partial\mathbf{F}$, or $T_{ijkl} = \partial P_{ij}/\partial F_{kl}$. We then write:

$$-\delta\boldsymbol{x}^T\delta\boldsymbol{f} = -\sum_{i,j}\delta f_i^{(j)}\delta x_i^{(j)} = V_e \sum_{j,k,l,m} T_{jklm}\delta F^e_{lm}\sum_i G^e_{ki}\delta x_i^{(j)}$$
$$\overset{\text{(Eqn.3)}}{=} V_e \sum_{j,k,l,m} T_{jklm}\delta F^e_{lm}\delta F^e_{jk} = V_e\left(\delta\mathbf{F}^e : \mathcal{T} : \delta\mathbf{F}^e\right).$$

Thus, $\mathbf{K}^e$ will be positive semi-definite, if and only if the fourth order tensor $\partial\mathbf{P}/\partial\mathbf{F}$ is positive definite as well (in the sense that $\delta\mathbf{F} : \mathcal{T} : \delta\mathbf{F} \geq 0$, for all $\delta\mathbf{F}$). At this point, consider a different 4th order tensor $\hat{\mathcal{T}}$ defined by $\delta\mathbf{P} = \mathcal{T} : \delta\mathbf{F} = \mathbf{R}[\hat{\mathcal{T}} : (\mathbf{R}^T\delta\mathbf{F})]$. Intuitively, if we define the *unrotated* differentials $\delta\hat{\mathbf{P}} = \mathbf{R}^T\delta\mathbf{P}$, and $\delta\hat{\mathbf{F}} = \mathbf{R}^T\delta\mathbf{F}$, then $\hat{\mathcal{T}}$ is the tensor that maps $\delta\hat{\mathbf{P}} = \hat{\mathcal{T}} : \delta\hat{\mathbf{F}}$. Tensors $\mathcal{T}$ and $\hat{\mathcal{T}}$ are a similarity transform of one another; consequently they share the same eigenvalues, and performing the indefiniteness fix on one will guarantee the definiteness of the other. Using this definition and equations (10) and (12), $\delta\hat{\mathbf{P}}$ reduces to:

$$\delta\hat{\mathbf{P}} = \hat{\mathcal{T}} : \delta\hat{\mathbf{F}} = 2\mu\delta\hat{\mathbf{F}} + \lambda\text{tr}(\delta\hat{\mathbf{F}})\mathbf{I} + \{\lambda\text{tr}(\mathbf{S}-\mathbf{I}) - 2\mu\}\,\mathcal{S} : \delta\hat{\mathbf{F}} \quad (18)$$

where $\mathcal{S} = \mathcal{E} : \{\text{tr}(\mathbf{S})\mathbf{I} - \mathbf{S}\}^{-1} : \mathcal{E}^T$. Consider the decomposition of $\delta\hat{\mathbf{F}} = \delta\hat{\mathbf{F}}_{\text{sym}} + \delta\hat{\mathbf{F}}_{\text{skew}}$ into symmetric $\delta\hat{\mathbf{F}}_{\text{sym}} = (\delta\hat{\mathbf{F}} + \delta\hat{\mathbf{F}}^T)/2$ and skew symmetric $\delta\hat{\mathbf{F}}_{\text{skew}} = (\delta\hat{\mathbf{F}} - \delta\hat{\mathbf{F}}^T)/2$ parts; also consider a similar decomposition of $\delta\hat{\mathbf{P}} = \delta\hat{\mathbf{P}}_{\text{sym}} + \delta\hat{\mathbf{P}}_{\text{skew}}$. By collecting symmetric and skew symmetric terms from equation (18) we have:

$$\delta\hat{\mathbf{P}}_{\text{sym}} = 2\mu\delta\hat{\mathbf{F}}_{\text{sym}} + \lambda\text{tr}(\delta\hat{\mathbf{F}}_{\text{sym}})\mathbf{I} = \mathcal{T}_{\text{sym}} : \delta\hat{\mathbf{F}}_{\text{sym}} \quad (19)$$
$$\delta\hat{\mathbf{P}}_{\text{skew}} = 2\mu\delta\hat{\mathbf{F}}_{\text{skew}} + \{\lambda\text{tr}(\mathbf{S}\text{-}\mathbf{I})\text{-}2\mu\}\,\mathcal{S} : \delta\hat{\mathbf{F}}_{\text{skew}} = \mathcal{T}_{\text{skew}} : \delta\hat{\mathbf{F}}_{\text{skew}}$$

In essence, $\hat{\mathcal{T}} = \mathcal{T}_{\text{sym}} + \mathcal{T}_{\text{skew}}$ has a fully decoupled action on the two subspaces of symmetric, and skew symmetric matrices. Since the symmetric and skew subspaces are orthogonal, $\hat{\mathcal{T}}$ will be semi-definite, if and only if its skew and symmetric parts are semi-definite too. The tensor $\mathcal{T}_{\text{sym}} = 2\mu\mathcal{I}_{\text{sym}} + \lambda\mathbf{I}\otimes\mathbf{I}$ ($\mathcal{I}_{\text{sym}}$ is the operator that projects a matrix onto its symmetric part) is always positive semi-definite; thus no modification is necessary. If $\mathcal{I}_{\text{skew}}$ is the operator that projects a matrix onto its skew symmetric part, we can verify that $2\mathcal{I}_{\text{skew}} = \mathcal{E} : \mathbf{I} : \mathcal{E}^T$. Thus, $\mathcal{T}_{\text{skew}}$ is written as:

$$\mathcal{T}_{\text{skew}} = \mu\mathcal{E} : \mathbf{I} : \mathcal{E}^T + \{\lambda\text{tr}(\mathbf{S}-\mathbf{I}) - 2\mu\}\left[\mathcal{E} : \{\text{tr}(\mathbf{S})\mathbf{I} - \mathbf{S}\}^{-1} : \mathcal{E}^T\right]$$

$$= \mathcal{E} : \mathbf{L} : \mathcal{E}^T, \text{ where } \mathbf{L} = \mu\mathbf{I} + \{\lambda\text{tr}(\mathbf{S}-\mathbf{I}) - 2\mu\}\{\text{tr}(\mathbf{S})\mathbf{I} - \mathbf{S}\}^{-1}$$

$\mathcal{E}$ is also an orthogonal (although not orthonormal) tensor, thus the definiteness of $\mathcal{T}_{\text{skew}}$ is equivalent with the definiteness of the 3×3 symmetric matrix $\mathbf{L}$, which can be easily projected to its positive definite part. In fact, if the method we used to compute the polar decomposition were to first compute the entire SVD $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ of the deformation gradient, then we have $\mathbf{L} = \mathbf{V}\mathbf{L}_D\mathbf{V}^T$ where

$$\mathbf{L}_D = \mu\mathbf{I} + \{\lambda\text{tr}(\boldsymbol{\Sigma}-\mathbf{I}) - 2\mu\}\{\text{tr}(\boldsymbol{\Sigma})\mathbf{I} - \boldsymbol{\Sigma}\}^{-1}$$

is a diagonal matrix, whose diagonal elements simply need to be clamped to zero, to ensure definiteness for $\mathbf{L}$, for $\mathcal{T}_{\text{skew}}$ and ultimately for the entire element stiffness matrix. In practical implementation, the matrix $\mathbf{L}$, projected to its semi-definite component, is precomputed and stored at the same time when the Polar Decomposition of each element is performed. Then, the definitions of this section are followed to successively construct $\delta\hat{\mathbf{P}}_{\text{sym}}$ and $\delta\hat{\mathbf{P}}_{\text{skew}}$, using equations (19), and ultimately $\delta\mathbf{P} = \mathbf{R}\delta\hat{\mathbf{P}}$.

So far, we have discussed how to correct the indefiniteness of the stiffness matrix arising from the (unstable) one-point quadrature technique. In light of the energy decomposition reflected in equation (17) the difference in the discrete energy between the stable and unstable approaches, is the discrete quadrature that will be followed to integrate the part Ψ_Δ. Our stable technique employs equation (15) for this task, while the original unstable technique uses one-point quadrature. In two spatial dimensions, if we denote by E^S_Δ and E^U_Δ the discrete integral associated with the Laplace term in the stable, and unstable variants respectively, we then have:

$$E^U_\Delta = \mu h^2 \sum_{i=1}^{2}\left((F^e_{i1})^2 + (F^e_{i2})^2\right)$$
$$= \mu h^2 \sum_{i=1}^{2}\left(\left(\frac{F^A_{i1}+F^B_{i1}}{2}\right)^2 + \left(\frac{F^C_{i2}+F^D_{i2}}{2}\right)^2\right)$$
$$E^U_\Delta - E^S_\Delta \overset{(15)}{=} \mu h^2 \sum_{i=1}^{2}\left(\left(\frac{F^A_{i1}-F^B_{i1}}{2}\right)^2 + \left(\frac{F^C_{i2}-F^D_{i2}}{2}\right)^2\right) \geq 0$$

Thus, we can interpret our stable discretization as adding the unconditionally convex term $E^U_\Delta - E^S_\Delta$ to the unstable energy discretization of the single-point approach. The indefiniteness fix described in the context of the unstable method can also be interpreted as augmenting the real stiffness matrix with a supplemental term $\mathbf{K} \leftarrow \mathbf{K} + \mathbf{K}_{\text{supp}}$ that guarantees the definiteness of the resulting matrix. The last equation indicates that if the same "definiteness-boosting" matrix $\mathbf{K}_{\text{supp}}$ is added to the stable discretization, definiteness will be guaranteed. Algorithm 1 summarizes the entire procedure that implements the "auxiliary" stress differential corresponding to the Ψ_{aux} energy component. The differential of the additional force due to the Laplace term Ψ_Δ are computed as described in section 4.

Impact on convergence Although the formulations in this section provide the benefit of a symmetric and definite linear system in every Newton iteration, we should be conscious of the fact that the force differentials thus produced are not identical to the exact expressions from equations (9) and (10) (also described in [Chao et al. 2010]). Since our decision of proper force differentiation was motivated by the lesser accuracy of the warped stiffness procedure, or the approach of [Zhu et al. 2010], we would like to assess the magnitude of inaccuracy that our definiteness correction incurs. Our method, as well as the aforementioned approximations effectively amount to a Modified Newton procedure for the force equilibrium equation. Notably, all methods reach the same equilibrium solution if converged; they simply follow different search paths towards that solution. From the theory of modified Newton methods, the convergence properties of the modified procedure depend on the spectral radius $Q = \rho(\mathbf{I} - \hat{\mathbf{J}}^{-1}\mathbf{J})$, where $\mathbf{J}$ is the proper force Jacobian, and $\hat{\mathbf{J}}$ is the approximation used in the modified method. If $Q < 1$ the

Algorithm 1 Computation of the stress differential corresponding to the auxiliary energy term Ψ_{aux}. Fixed to guarantee definiteness.

function COMPUTE_L($\mathbf{\Sigma}, \mathbf{V}, \mu, \lambda, \mathbf{L}$)
  $\mathbf{L}_D \leftarrow \{\lambda \text{tr}(\mathbf{\Sigma}-\mathbf{I})-2\mu\}\{\text{tr}(\mathbf{\Sigma})\mathbf{I}-\mathbf{\Sigma}\}^{-1}$
  Clamp diagonal elements of $\mathbf{L}_D$ to a minimum value of $(-\mu)$ ▷ Term Ψ_Δ will boost this eigenvalue by μ
  $\mathbf{L} \leftarrow \mathbf{V}\mathbf{L}_D\mathbf{V}^T$
end function
function DPAUXDEFINITEFIX($\delta\mathbf{F}, \mathbf{R}, \mathbf{L}$) ▷ Returns $\delta\mathbf{P}_{\text{aux}}$
  $\delta\hat{\mathbf{F}}_{\text{sym}} \leftarrow \text{SYMMETRICPART}(\mathbf{R}^T\delta\mathbf{F})$
  $\delta\hat{\mathbf{F}}_{\text{skew}} \leftarrow \text{SKEWSYMMETRICPART}(\mathbf{R}^T\delta\mathbf{F})$
  $\delta\hat{\mathbf{P}}_{\text{sym}} \leftarrow \lambda\text{tr}(\delta\hat{\mathbf{F}}_{\text{sym}})\mathbf{I}$
  $\delta\hat{\mathbf{P}}_{\text{skew}} \leftarrow \mathcal{E} : \left\{\mathbf{L}(\mathcal{E}^T : \delta\hat{\mathbf{F}}_{\text{skew}})\right\}$
  $\delta\mathbf{P}_{\text{aux}} \leftarrow \mathbf{R}\left(\delta\hat{\mathbf{P}}_{\text{sym}} + \delta\hat{\mathbf{P}}_{\text{skew}}\right)$
  return $\delta\mathbf{P}_{\text{aux}}$
end function

modified procedure is convergent (in fact, when $Q \ll 1$, quadratic convergence is practically retained); however when $Q > 1$ there is no guarantee of convergence, and certain error modes exist that will be amplified by the modified iteration.

In our work, $\hat{\mathbf{J}}$ is the result of the indefiniteness correction previously described. For warped stiffness $\hat{\mathbf{J}}$ results from the approximation of the stress differential of equation (10) by the simpler expression $\delta\mathbf{P} = \mathbf{R}\left\{\mu\left[(\mathbf{R}^T\delta\mathbf{F}) + (\mathbf{R}^T\delta\mathbf{F})^T\right] + \lambda\text{tr}(\mathbf{R}^T\delta\mathbf{F})\right\}$. The formulation of [Zhu et al. 2010] implies a similar approximation, namely $\delta\mathbf{P} = \mathbf{R}\left\{2\mu(\mathbf{R}^T\delta\mathbf{F}) + \lambda\text{tr}(\mathbf{R}^T\delta\mathbf{F})\right\}$. The spectral radius Q corresponding to all three approaches remains relatively close to zero (< 0.05) for deformations that are small *and* smooth. However, our method remains safely convergent, significantly more so than the other two alternatives, even with larger, non-smooth deformations. For example, in the scenario of figure 2, at the moment when warped stiffness develops an instability ($2.4\times$ extension) we have $Q = 4.87$ for warped stiffness, $Q = 0.829$ for the approximation of [Zhu et al. 2010] and $Q = 0.0717$ for our approach.

6 Dynamics

Our method extends trivially to dynamic simulations that include inertial effects. However, it is important to note that the indefiniteness encountered in quasistatic time stepping also arises in implicit time stepping for dynamics. Fortunately, the definiteness fix outlined above can be used in this setting as well. In this case, we typically desire a fixed, large time step of $\Delta t \approx \frac{1}{30}$. Using backward Euler time stepping and Newton-Raphson linearization, the following linear update equation must be solved for the increment $\delta\boldsymbol{x}$ in the k^{th} iteration

$$\mathbf{K}^{\text{BE}}(\boldsymbol{x}_k^{n+1})\delta\boldsymbol{x} = \Delta t\mathbf{M}\left[\boldsymbol{v}^n + \Delta t(\boldsymbol{x}^n - \boldsymbol{x}_k^{n+1})\right] + \Delta t^2\boldsymbol{f}(\boldsymbol{x}_k^{n+1})$$

Here $\mathbf{K}^{\text{BE}} = \mathbf{M} + \Delta t^2\mathbf{K}(\boldsymbol{x}_k^{n+1})$, and $\mathbf{M}$ is the mass matrix. Indefiniteness of $\mathbf{K}(\boldsymbol{x}_k^{n+1})$ can thus be seen to potentially cause indefiniteness of $\mathbf{K}^{\text{BE}}(\boldsymbol{x}_k^{n+1})$. One could attempt to manipulate nodal masses or material properties to preserve definiteness, but this alters the behavior of the simulation in arbitrary ways. Although decreasing the timestep could also fix the indefiniteness, the time step cannot be decreased arbitrarily when interactivity is desired. Furthermore, it is important to note that the nodal mass is proportionate to the volume associated with each node. Therefore, as we increase the discrete spatial resolution of our domain, the nodal mass decreases thereby increasing the likelihood of encountering an indefinite backward Euler system matrix as it would behave more and more like the indefinite $\mathbf{K}(\boldsymbol{x}_k^{n+1})$. See Figure 4 for a numerical experiment that demonstrates this behavior. Therefore, when both high performance and high resolution are desired, indefiniteness in the backward Euler system matrix quite likely. Fortunately, the definiteness fix in section 5 for $\mathbf{K}(\boldsymbol{x}_k^{n+1})$ guarantees definiteness of the backward Euler system matrix.

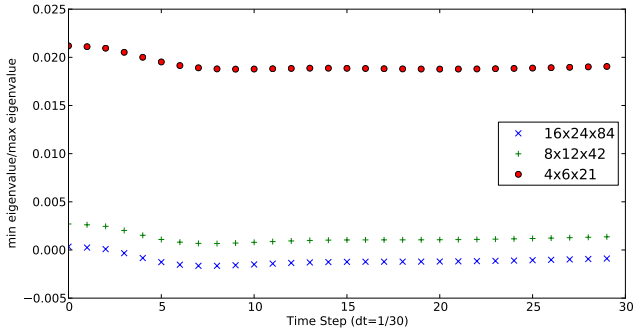


Figure 4: *Plot of ratios of minimum to maximum eigenvalues of the backward Euler matrix of a dynamic elastic bar simulation without our definiteness fix applied. Note that the minimum eigenvalues are negative in the 16x24x84 resolution example.*

7 Constraints and collisions

As previously discussed, the ability to handle elaborate collision is an essential benefit of simulation in production. We use point constraints to enforce both soft constraints, such as bone attachments, and to handle object and self collisions. Specifically, we embed proxy points ($\mathbf{x}_p$) in the simulation lattices and distribute their associated forces trilinearly to the vertices of the hexahedral cells that contain them. [Sifakis et al. 2007] show the effectiveness of this basic approach.

Collision detection The collision response is determined by a number of collision proxies approximately covering the embedded collision surface. We utilize a penalty based response dependent on the penetration depth and unit outward normal at each proxy point. For rigid objects, we simply query a level set representation of the object at each proxy point. However, for self-collision, the rapidly changing shape of the elastic objects precludes accurate reconstruction of a signed distance function at each time step.

Self-collision penetration depth evaluation For each proxy collision point, we first determine which deformed hexahedra contain it in the current configuration. This is done rapidly by querying an axis aligned bounding box hierarchy whose leaves surround each

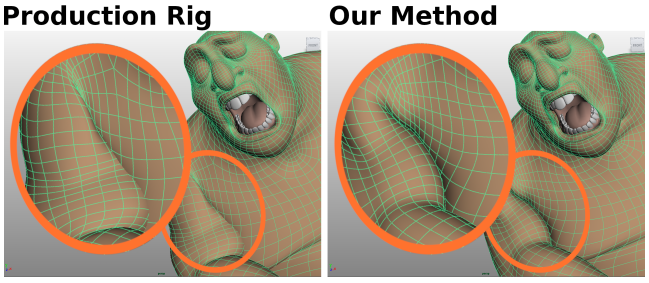


Figure 5: *Collisions and especially self-collisions drastically improve the quality of deformation when coupled with elasticity. On the left is a production rig that qualitatively exhibits the right look but does not resolve collisions. On the right is our method which resolves self-collisions producing a much more natural look.* ©*Disney Enterprises, Inc.*

deformed hexahedron in the current configuration. To prevent false positives, we do not look in the 27 hexahedra in the one ring of the proxy point in material coordinates. Each hexahedron deemed near a given proxy point is then tetrahedralized to barycentrically determine the proxy point's material location. For each material point, we query a level set stored in the undeformed configuration: ϕ_0. If there are multiple negative ϕ_0 values, we use the location with ϕ_0 closest to zero to compute the closest point on the undeformed surface. We then look up the deformed position of the closest surface point ($\mathbf{x}_s$) to estimate the penetration depth as $|\mathbf{x}_s - \mathbf{x}_p|$ and outward unit normal as $\mathbf{n} = (\mathbf{x}_s - \mathbf{x}_p) \,/\, |\mathbf{x}_s - \mathbf{x}_p|$.

Collision response For both self-collision and solid object collision scenarios, we instantiate a zero rest-length spring from the proxy point to the closest point on the surface. The Young's modulus of this spring is allowed to be anisotropic in the direction of the unit collision normal. Specifically, the spring force arises from the energy $\Psi(\mathbf{x}_p, \mathbf{x}_s) = k(\mathbf{x}_p - \mathbf{x}_s)^T \mathbf{M}(\mathbf{x}_p - \mathbf{x}_s)/2$ where $\mathbf{M} = (1-\alpha)\,\mathbf{n}\mathbf{n}^T + \alpha\mathbf{I}$, with $\alpha \in [0,1]$. $\alpha = 1$ corresponds with a traditional isotropic spring; $\alpha = 0$ results in a standard point repulsion. This anisotropic conception of the stiffness allows for sliding in the plane orthogonal to the penetration direction. In practice we found $\alpha \in [.1, .5]$ worked best for self collisions and $\alpha = 0$ was sufficient for object collisions.

8 Multigrid

To ensure that our method scales to high resolutions, we solve the equations of elasticity using a multigrid technique. In fact, we explored two possible approaches: the first option is to construct a multigrid cycle purely as a solver for the linear system (8) generated in every Newton-Raphson step. The other possibility is to implement a fully nonlinear multigrid cycle, based on the Full Approximation Scheme (FAS) which would replace and accelerate the entire sequence of Newton iterations. This section details several design subtleties, and algorithmic modifications that were necessary to efficiently implement these two multigrid schemes.

Domain description Our discretization is based on a voxelized representation of the elastic body. At any given resolution, a cubic background lattice is defined and its cells are labeled either *internal* or *external* depending on any material overlap with the embedded deforming body. Internal cells can optionally be labeled as *constrained* (or Dirichlet) if the trajectories of their nodes will be specified as hard kinematic constraints. The Lamé coefficients μ and λ can be specified for each individual cell, allowing for inhomogeneous models. The coarser domains of a multigrid hierarchy are generated by a simple binary coarsening strategy. Similar to [Zhu et al. 2010] a label of constrained, internal or external is assigned in this order of priority, if fine children with more than one type are present. The Lamé parameters of coarse interior cells are computed by summing the μ or λ of any *interior* children, and dividing by eight; thus coarse cells overlapping with the boundary receive lower material parameters, to account for the partial material coverage of the cell.

Elasticity coarsening A multigrid method requires us to generate a hierarchy of discretizations. Specifically, if we use multigrid to solve the linear system (8), different versions of $\mathbf{K}$, denoted by $\mathbf{K}^h, \mathbf{K}^{2h}, \mathbf{K}^{4h}, \ldots$ need to be computed for every level of the multigrid hierarchy. We specifically avoid the Galerkin coarsening strategy since it requires forming the stiffness matrices explicitly. The alternative is a matrix-free approach which constructs $\mathbf{K}^{2h}$ from a re-discretization of our problem at the coarse grid. We can repeat the same process followed at the fine grid, and define coarse forces $\boldsymbol{f}^{2h}(\boldsymbol{x}^{2h}) = -\partial\Psi^{2h}/\partial\boldsymbol{x}^{2h}$ as well as a coarse stiffness $\mathbf{K}^{2h} = \partial\boldsymbol{f}^{2h}/\partial\boldsymbol{x}^{2h}$ and encode these in a matrix-free fashion as before. The challenge however is that the entries in $\mathbf{K}^{2h}$ depend on the current estimate of the solution and, more accurately, on a *coarse grid* version $\boldsymbol{x}^{2h}$ of this estimate. The general methodology is to define yet another restriction operator $\hat{\mathbf{R}}$ (possibly different than the one used to restrict residuals) to downsample the solution estimate as $\boldsymbol{x}^{2h} = \hat{\mathbf{R}}\boldsymbol{x}^h$. However, as a consequence of our geometric domain coarsening described in the previous paragraph, the discrete domain *grows* in size, as coarse cells with any interior children will now be considered fully interior, even if they include some exterior cells from the fine grid. Therefore, restricting the approximation $\boldsymbol{x}^h$ would require extrapolation of the deformation field. We found such extrapolations to be quite unstable, especially in the presence of collisions, and sometimes even ill-defined near concave, high curvature boundaries.

Our solution is based on the observation that the entries of $\mathbf{K}$ do not depend directly on the positions $\boldsymbol{x}$, but only through the deformation gradient $\mathbf{F}$. Note that this is also true for our stabilized discretization; the part $\mathbf{K}_\Delta$ of the stiffness matrix due to Ψ_Δ is a *constant* matrix, not dependent on positions at all. The auxiliary part $\mathbf{K}_{\text{aux}}$ due to Ψ_{aux} is fully determined by the discrete deformation gradient $\mathbf{F}^e$ at every cell. Thus, instead of restricting $\boldsymbol{x}^h \rightarrow \boldsymbol{x}^{2h}$ we instead downsample the deformation gradient as $\mathbf{F}^h \rightarrow \mathbf{F}^{2h}$, which is done with simple weighted averaging. Once the stiffness matrices have been constructed for all levels, the V-Cycle described in Algorithm 2 is used to solve the linearized Newton equation. The restriction and prolongation operators are constructed based on trilinear interpolation. Since we do not explicitly construct $\mathbf{K}$, we use a Jacobi smoother instead of a Gauss-Seidel one, since for the Jacobi procedure all force differentials may be computed in parallel. Note however that the elasticity matrix is not diagonally dominant, and the Jacobi procedure needs to be damped for stable convergence. We found that the damping coefficient could safely be as high as 0.8 in the interior of the object, while values of $0.3 - 0.4$ were more appropriate near the boundary, near soft constraints, and for higher values of Poisson's ratio.

Algorithm 2 Linear Multigrid V(1,1) Cycle for equation (8)

procedure LINEARVCYCLE
  $\boldsymbol{b}^h \leftarrow \boldsymbol{f}^h(\boldsymbol{x}^h) + \boldsymbol{g}^h$
  for $l = 0$ **to** $L-1$ **do** ▷ total of $L+1$ levels
    Smooth($\mathbf{K}^{2^l h}$,$\delta\boldsymbol{x}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
    $\boldsymbol{r}^{2^l h} \leftarrow \boldsymbol{b}^{2^l h} - \mathbf{K}^{2^l h}\delta\boldsymbol{x}^{2^l h}$
    $\boldsymbol{b}^{2^{l+1} h} \leftarrow$Restrict($\boldsymbol{r}^{2^l h}$), $\delta\boldsymbol{x}^{2^{l+1} h} \leftarrow 0$
  end for
  Solve $\delta\boldsymbol{x}^{2^L h} \leftarrow (\mathbf{K}^{2^L h})^{-1}\boldsymbol{b}^{2^L h}$
  for $l = L-1$ **down to** 0 **do**
    $\delta\boldsymbol{x}^{2^l h} \leftarrow \delta\boldsymbol{x}^{2^l h} +$Prolongate($\delta\boldsymbol{x}^{2^{l+1} h}$)
    Smooth($\mathbf{K}^{2^l h}$,$\delta\boldsymbol{x}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
  end for
end procedure

Point constraint coarsening Each soft constraint and active collision proxy is copied to the coarse grids based on its material location. Its associated stiffness modulus is scaled by a factor of .125 (or .25 in 2D) to accommodate its embedding in a larger element. Otherwise, the coarsened proxies are then treated in the same manner at every level of the hierarchy.

Nonlinear multigrid We also implemented a fully nonlinear multigrid solver, based on the Full Approximation Scheme (FAS)

approach. As before, the challenge is that the nonlinear force operator requires a coarse grid version of the solution estimate. Once again, the operator only depends on $\boldsymbol{x}$ through the deformation gradient; unfortunately the deformation gradient does not stay fixed through smoothing and v-cycles, requiring constant updates. We consider the restricted value of the deformation gradient as an "offset" (denoted by $\mathbf{F}_{\text{off}}$) and change our state variables for the coarser grids from *positions* ($\boldsymbol{x}$) to *corrections* (u) from this offset. We compute the updated deformation as $\mathbf{F} = \mathbf{F}_{\text{off}} + \mathbf{G}[\boldsymbol{u}]$, where $\mathbf{G}$ is the cell-centered gradient operator. The nonlinear forces computed based on this updated gradient are $\boldsymbol{f}^h(\mathbf{F}^h_{\text{off}}; \boldsymbol{u}^h)$. The FAS procedure is outlined in Algorithm 3. Damped Jacobi is used, albeit with re-linearization steps inserted between every 2-3 Jacobi iterations.

Algorithm 3 FAS V-Cycle for nonlinear equilibrium equation

procedure FASVCYCLE($\boldsymbol{f}^h(\mathbf{F}^h_{\text{off}}; \boldsymbol{u}^h) + \boldsymbol{g}^h = \mathbf{0}$)
  NONLINEARSMOOTH($\boldsymbol{f}^h(\mathbf{F}^h_{\text{off}}; \boldsymbol{u}^h) + \boldsymbol{g}^h = \mathbf{0}$)
  $\mathbf{F}^{2h}_{\text{off}} \leftarrow \hat{\mathbf{R}}(\mathbf{F}^h_{\text{off}} + \mathbf{G}^h[\boldsymbol{u}^h]), \boldsymbol{u}^{2h} \leftarrow \mathbf{0}$
  $\boldsymbol{g}^{2h} \leftarrow -\boldsymbol{f}^{2h}(\mathbf{F}^{2h}_{\text{off}}; \boldsymbol{u}^{2h}) + \mathbf{R}(\boldsymbol{f}^h(\mathbf{F}^h_{\text{off}}; \boldsymbol{u}^h) + \boldsymbol{g}^h)$
  SOLVE($\boldsymbol{f}^{2h}(\mathbf{F}^{2h}_{\text{off}}; \boldsymbol{u}^{2h}) + \boldsymbol{g}^{2h} = \mathbf{0}$) ▷ By recursive call
  $\boldsymbol{u}^h$ +=Prolongate($\boldsymbol{u}^{2h}$)
  NONLINEARSMOOTH($\boldsymbol{f}^h(\mathbf{F}^h_{\text{off}}; \boldsymbol{u}^h) + \boldsymbol{g}^h = \mathbf{0}$)
end procedure

Our experiments with the fully nonlinear FAS cycle exhibited significant acceleration in certain tasks, but was somewhat more questionable in terms of cost/performance for our typical use scenario. In particular, when solving for the equilibrium shape following a very sudden skeletal motion (such as fully curling an arm from a full extension, all in a single frame), the FAS scheme was able to converge in just a few cycles, while the linear multigrid approach would typically necessitate a large number of Newton iterations (even though each linear system was adequately solved). However, in our typical use, the skeletal motion was relatively incremental, and 1-2 Newton iterations would provide an excellent solution estimate, with 1-2 linear V-cycles as the underlying linear solver. Since the linear cycle is less expensive due to less frequent re-linearization, this approach was overall a better cost/performance choice for our specific application. Additionally, in the presence of collisions, extreme skeletal motions within a single frame were problematic, since the penalty-based collisions would easily become tangled in nonphysical configurations. Lastly, we observed that the ability of FAS to make large corrections towards the solution often times had the side effect of sometimes jumping between different local minima, especially in highly buckled configurations. This behavior can be controlled by augmenting FAS with a line-search step to prevent it from tunneling to different minima, but again this was not a priority for our target application.

9 Optimization

CPU. To improve performance on the CPU we utilized multithreading using a task queue. We designed our access patterns to be cache friendly by using blocking techniques. We also exploited SSE data level parallelism for the SVD computation. Additionally, we used templatization to optimize stride multiplication computations in array accesses. Constraint contributions to the matrix were baked into a structure that minimized indirection.

GPU. Since GPUs have become popular for parallel numerical algorithms, we did a naïve port of our CPU oriented code to the GPU. Our expertise at optimizing for the GPU's different bandwidth versus computation tradeoffs was limited, and we hope to utilize the grid optimization techniques of [Dick et al. 2011] in the future.

We benchmarked our elasticity multigrid solver on a cube model with Dirichlet constraints on two opposite faces without collisions or point constraints. While we were able to attain a convergent method with as few as 2 Jacobi smoothing sweeps per grid transfer, we were able to achieve the best balance between speed and convergence rate with 5-10 Jacobi relaxation sweeps. In all cases, the first V-cycles significantly (by 1-2 orders of magnitude) lowered the residual before settling into a constant convergence rate between 0.5 and 0.75, depending on the number of relaxation sweeps. On a $32 \times 32 \times 32$ element cube, we averaged 0.031s on the GPU and 0.10s on the CPU per V-cycle with 10 relaxation sweeps per grid transfer on 4 levels. On a $64 \times 64 \times 64$ element cube, we averaged 0.086s on the GPU and 0.56s on the CPU per V-cycle with 10 relaxations sweeps on 5 levels. In practice, we found that 1-2 V-cycles were sufficient for the Newton-Raphson solver to converge.

9.1 Diagonal part of stiffness matrix

The Jacobi iteration used as the smoother in our multigrid scheme requires explicit knowledge of the diagonal part of the stiffness matrix $\mathbf{K}$. Since we never construct this matrix explicitly, a specialized process needs to be followed to compute the diagonal part directly and efficiently. We will provide the final result here, and refer the interested reader to the accompanying technical document for the detailed algebraic derivation. For element Ω_e we will define the contribution of each of the 24 degrees of freedom to the diagonal part of $\mathbf{K}$. In particular, we turn our attention to the degree of freedom $x_i^{(j)}$, that is the j-th component of the i-th element vertex, where $i \in \{1, \ldots, 8\}$. Also, define $\boldsymbol{g}$ to be the i-th *column* of the discrete gradient $\mathbf{G}$, and $\boldsymbol{r}^T$ the j-th *row* of $\mathbf{R}$. Also, define $\mathbf{M}_i = \mathcal{E} : \boldsymbol{g}$ (i.e. the right-side cross product matrix with $\boldsymbol{g}$). Also define $\mathbf{N}_i = \lambda \boldsymbol{g}\boldsymbol{g}^T + \mathbf{M}_i^T \mathbf{L} \mathbf{M}_i$. where $\mathbf{L}$ was defined in section 5. The diagonal contribution of Ω_e to the diagonal part corresponding to degree of freedom $x_i^{(j)}$ is $\frac{3}{2}\mu/h^2 + \boldsymbol{r}^T \mathbf{N}_i \boldsymbol{r}$. We can verify that the diagonal entries corresponding to antidiametric vertices (e.g. $\boldsymbol{x}_1$ and $\boldsymbol{x}_8$) are equal; thus the diagonal entry need only be computed for the components of 4 out of the 8 vertices of the element. Finally $\mathbf{M}_i$ can be precomputed, we only need to consider $i = 1, \ldots, 4$ as we explained, and these matrices are identical for all elements.

9.2 Fast Singular Value Decomposition

Our method makes use of the Singular Value Decomposition $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ to define the matrix $\mathbf{L}$, and in fact we use it to construct the rotational factor of the Polar Decomposition as well, as $\mathbf{R} = \mathbf{U}\mathbf{V}^T$. The cost of the 3×3 SVD is commonly acknowledged as a bottleneck for corotational or shape matching methods. We introduce a new, and highly efficient methodology which is virtually branch-free (other than the use of conditional assignments, which is an atomic instruction in SSE4.1 and other platforms), uses no expensive arithmetic other than addition, subtraction, multiplication and an *inexact* square root (i.e. the SSE `VRSQRTPS` instruction), and is trivially and extensively vectorizable. We ultimately obtain a cost per decomposition equal to 11ns on a 12-core X5650 workstation, using SSE and multithreading in conjunction.

The core cost of the SVD analysis is commonly reported to be the symmetric eigenanalysis. An iterative Jacobi diagonalization procedure is often used to bring the symmetric matrix $\mathbf{S} = \mathbf{F}^T\mathbf{F}$ into diagonal form. Instead of the exact Givens conjugation used in the Jacobi procedure, we define an *approximate* Givens angle, which can be obtained with minimal computation. The optimal Givens angle that annihilates element s_{21}, for example is known to satisfy $\tan(2\theta) = 2s_{12}/(s_{11} - s_{22})$. Instead of using an inverse trigonometric function (or the alternative approach of solving a quadratic equation), we consider the following asymptotic approximation,

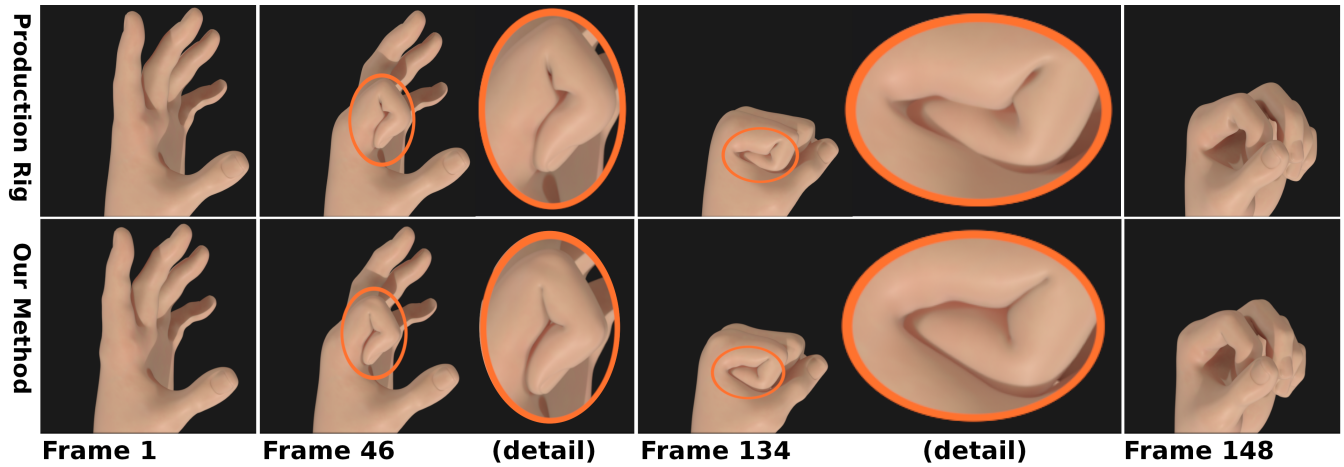


Figure 6: *Top: A hand rigged using linear blend skinning. Due to the many joints it is impractical to model all the shapes necessary fix the artifacts using an example based technique like PSD. To compensate, a significant amount of effort was spent on creating a nice skin bind for this hand. Bottom: The same hand rigged with our elasticity based deformer using 117,607 non-empty cells. Note how we get correct creasing and contact deformation where the finger bends.* ©*Disney Enterprises, Inc.*

valid when θ is small:

$$\frac{2s_{12}}{s_{11}-s_{22}} = \tan(2\theta) \approx 4\tan(\frac{\theta}{2}) \Rightarrow \frac{\sin(\theta/2)}{\cos(\theta/2)} \approx \frac{s_{12}}{2(s_{11}-s_{22})}$$

At this point, we observe that this approximate rotation can be stored as the un-normalized quaternion $(2(s_{11}-s_{22}), 0, 0, s_{12})$, eliminating the need for divisions or exact normalizations. In fact, we do perform an *inexact* normalization using the approximate VRSQRTPS SSE instruction, merely for the purpose of avoiding overflow, and without this inaccuracy affecting the semantics of the quaternion. However, the asymptotic approximation does not hold for arbitrary θ, and it would be possible that the off-diagonal element would not be reduced (let alone, annihilated) for certain cases. However, we can show that the best of either the above approximation *or* a choice of $\theta = \pi/4$ is guaranteed to reduce the magnitude of the off diagonal element s_{12} by at least a factor of 0.6 per application. Deciding which one to use is made with a simple algebraic check, as shown in Algorithm 4.

Algorithm 4 Computation of approximate Givens quaternion.

const $\gamma \leftarrow 3+2\sqrt{2}$, $c_* \leftarrow \cos(\pi/8)$, $s_* \leftarrow \sin(\pi/8)$
function APPROXGIVENSQUATERNION(a_{11}, a_{12}, a_{22})
  $c_h \leftarrow 2(a_{11}-a_{22})$ ▷ $c_h \approx \cos(\theta/2)$
  $s_h \leftarrow a_{12}$ ▷ $s_h \approx \sin(\theta/2)$
  $b \leftarrow [\gamma s_h^2 < c_h^2]$ ▷ b is boolean
  $\omega \leftarrow \mathsf{RSQRT}(c_h^2 + s_h^2)$ ▷ $\mathsf{RSQRT}(x) \approx 1/\sqrt{x}$
  $c_h \leftarrow b?\omega c_h : c_*$
  $s_h \leftarrow b?\omega s_h : s_*$
  return $(c_h, 0, 0, s_h)$ ▷ returns a quaternion
end function

Once the matrix $\mathbf{S}$ has been brought closer to a diagonal, our asymptotic approximation becomes extremely accurate, and essentially matches the efficiency of regular Jacobi iteration (but at a fraction of the implementation cost). We have invariably observed that 4 sweeps of our method offer the same efficacy in diagonalizing $\mathbf{S}$ as 3 sweeps of the regular Jacobi method, and this brings the magnitude of off-diagonal entries to 4-5 order of magnitude smaller than the singular values on the diagonal. Once the symmetric eigenanalysis has been computed, we robustly compute the rotational factor $\mathbf{U}$ by performing a Givens $\mathbf{QR}$ factorization on $\mathbf{AV} = \mathbf{U\Sigma}$, which

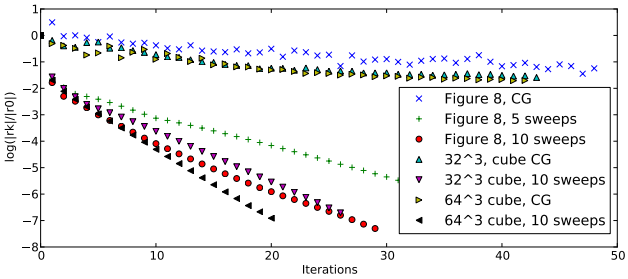


Figure 7: *Residual reduction $|r_k|_\infty/|r_0|_\infty$ per V-cycle or CG iteration for a number of examples. In practice 1-3 V-cycles were necessary for each Newton-Raphson update.*

generates an exactly orthogonal factor $\mathbf{Q} = \mathbf{U}$ and a triangular factor $\mathbf{R}$ which will in fact be *diagonal* (and equal to $\mathbf{\Sigma}$) as long as the columns of $\mathbf{U\Sigma}$ are sorted in descending order of magnitude by permuting them along with the columns of $\mathbf{V}$.

We contrast this procedure to the simpler approach that computes the rotational factor as $\mathbf{R} = \mathbf{FS}^{-1}$, as demonstrated for example in [Rivers and James 2007]. Although quite efficient, such a treatment is not robust for our intended simulation task, where flattened or inverted elements are commonplace; in the case of an inverted element, where $\det(\mathbf{F}) < 0$ the formula $\mathbf{R} = \mathbf{FS}^{-1}$ would produce a matrix $\mathbf{R}$ that includes a reflection (i.e. $\det(\mathbf{R}) = -1$) which violates the proper semantics of the corotated formulation, and inhibits untangling of inverted elements in practice. Similarly, for flattened elements the factor $\mathbf{S}$ is near-singular, which jeopardizes the orthogonality of the computed matrix $\mathbf{R}$. Methods that produce the polar decomposition robustly in the case of singular or inverted $\mathbf{F}$ (using cross products to generate perfect rotations) will typically resort to case analysis, preventing effective vectorization. In contrast, our Givens QR factorization has a completely deterministic control flow and is trivially vectorized. See our supplemental technical document for full analysis, implementation, and comparison with existing iterative and closed form solvers.

10 Results

We tested our full system on a number of production-quality models, focusing on regions where artists typically struggle to achieve

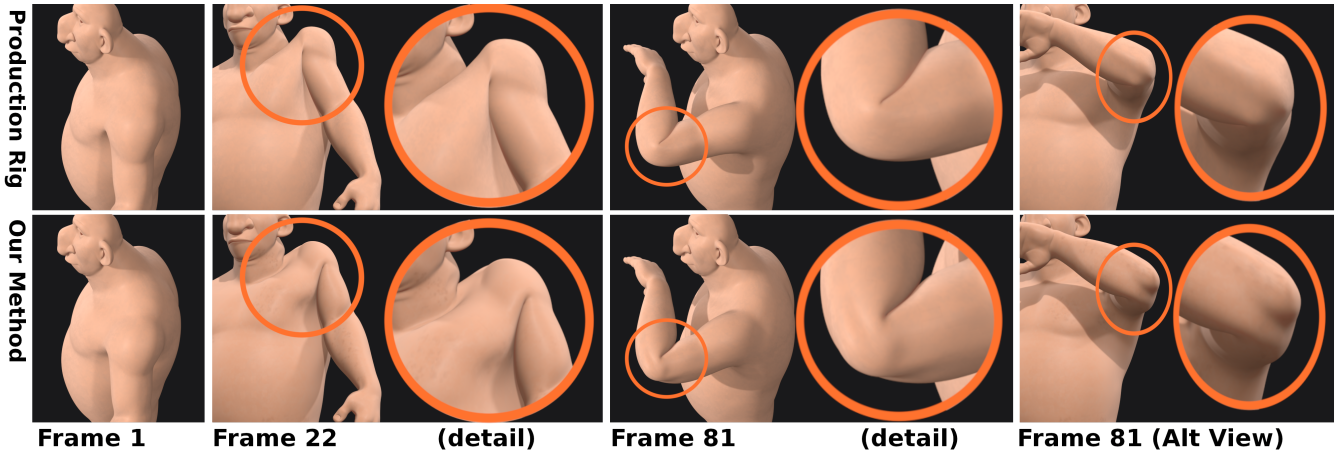


Figure 8: *A common problem with many existing techniques (and linear blend skinning in particular) is that it can be very difficult to maintain the illusion of an underlying bone-structure. As an example the top row here shows how the integrity of the region around the clavicle is lost as the character shrugs. In the bottom row, however, everything behaves as a connected entity. On the outside of the elbow we also get a nice protrusion of the ulna with our method as opposed to the more rubber-like behavior of the linear blend skin. (30,904 non-empty cells)* ©*Disney Enterprises, Inc.*

realistic, collision-free results using traditional rigging methods. Our simulation is driven by rigid bones attached to the character's existing skeleton. Simple geometric shapes such as cylinders or ellipsoids are sufficient to define the volumetric extent of these bones, and we found that using soft constraints with narrow bones gave the elasticity model freedom to produce appealing flesh-like shapes. More carefully modeled bones can be used as internal collision objects over which the material can slide, providing detail in regions such as the elbow or around the collarbone. We allowed additional artistic control by providing the ability to spatially vary μ and λ. The hexahedral lattice is constructed as an axis-aligned grid with uniformly sized cells in the undeformed configuration.

10.1 Examples

In all examples, we found that 1-2 V-cycles with 5 relaxation sweeps per grid transfer were sufficient for the linear solver. The number of Newton iterations required depended on our initial guess; when using the previous frame of an animation, we typically required between 1 and 10 iterations for full convergence. All reported CPU times were computed on an 8-core Intel Xeon X5550 workstation. GPU tests were performed on an NVIDIA Quadro 6000. In our first example, we applied our deformer to the arm, shoulder and neck region of a character (see Fig. 8). Each Newton iteration averaged 0.492s on the CPU and 0.345s on the GPU. Our average frame times were 3.22s and 2.38s on the CPU and GPU respectively. In Figure 6, we demonstrate our method on a character hand. On the CPU, we averaged 1.40s per Newton iteration for an average of 12.6s per frame. On the GPU, each Newton iteration averaged 0.612s for an average of 5.74s per frame. In Figure 1 we apply our deformer to the torso and arms of a large character with 106,567 elements. On the CPU, each Newton iteration averaged 0.876s for a total of 5.48s per frame. Our GPU implementation averaged 0.762s per Newton iteration and 5.14s per frame.

In Figure 7, we compare convergence rates per V-cycle or conjugate gradients iteration. We suspect the initial "bump" in residual reduction per V-cycle is due to the efficiency of our Jacobi smoother. A constant convergence rate emerges for all multigrid examples, in contrast with CG.

11 Discussion

Originally, we were motivated by the desire to make true physically based elasticity practical for production character rigging. We found that beyond our expectations, artists were impressed with the shapes a physical deformer could provide with little manual effort. What usually took days of weight painting or pose example sculpting was now easy to achieve, and what was impossible, collision and contact, was now possible.

A major decision in our project was to focus on optimizing for the CPU, because we were familiar with CPU optimization and because simulations often need to run on clusters without GPUs. At the same time we recognize GPUs are becoming more important because of their power and we sought to experiment with the GPU. Though we have little experience with the GPU, we believe our method will perform even better on the GPU with further optimizations. Further, we note that most GPU experiments tend to compare against relatively unoptimized CPU implementations, whereas our CPU implementation was heavily optimized.

Obviously, speed has been a major hurdle for the use of physical simulation in production, so the fact that our simulations can run at near-interactive rates changes the game for artists. Even so, there is plenty of future work to do. One area is making the parameters (e.g. Young's modulus) more intuitive for artists to control, and another is allowing art direction in other ways, such as optimized control (which also requires fast solvers). Robustness could be improved by using more accurate self-collision methods and a more accurate initial guess. Finally, our solver does not support near incompressible materials, and we would be interested in exploring additional features such as material anisotropy. But even without this future work, we believe that our contribution will help create the next generation of appealing characters.

Acknowledgments

Y.Z. was affiliated with Walt Disney Animation Studios and UCLA while working on this technique. We also thank David Kersey for his help with rendering. Part of this research was supported by NSF DMS-0502315, NSF DMS-0652427, NSF CCF-0830554, DOE 09-LR-04-116741-BERA, ONR N000140310071, and ONR N000141010730.

References

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. 26* (July), 7:21–7:28.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND Z. POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. In *Proc. SIGGRAPH 2002*, 586–593.

CAPELL, S., BURKHART, M., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2005. Physically based rigging for deformable characters. In *Proc. 2005 ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*

CHADWICK, J., HAUMANN, D., AND PARENT, E. 1989. Layered construction for deformable animated characters. In *Proc. SIGGRAPH 89*, 243–252.

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

CLUTTERBUCK, S., AND JACOBS, J. 2010. A physically based approach to virtual character deformation. In *ACM SIGGRAPH 2010 talks*, SIGGRAPH '10.

DICK, C., GEORGII, J., AND WESTERMANN, R. 2011. A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Sim. Mod. Prac. Th. 19*, 2, 801–816.

GALOPO, N., OTADUY, M., TEKIN, S., GROSS, M., AND LIN, M. 2007. Soft articulated characters with fast contact handling. *Comp. Graph. Forum 26*, 243–253.

GEORGII, J., AND WESTERMANN, R. 2006. A multigrid framework for real-time simulation of deformable bodies. *Comp. Grap. 30*, 3, 408 – 415.

GERRITSMA, M. 1996. *Time dependent numerical simulations of a viscoelastic fluid on a staggered grid*. PhD thesis, University of Groningen.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23* (August), 463–468.

HARLOW, F., AND WELCH, J. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *Phys. Fl. 8*, 2812–2189.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 131–140.

IRVING, G., KAUTZMAN, R., CAMERON, G., AND CHONG, J. 2008. Simulating the devolved: finite elements on WALL·E. In *ACM SIGGRAPH 2008 talks*, ACM, New York, NY, USA, SIGGRAPH '08, 54:1–54:1.

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

KRY, P., JAMES, D., AND PAI, D. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 153–159.

LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH '00*, 165–172.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proc. Graph. Inter. '88*, 26–33.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation Space: A Truly Linear Framework for Character Animation. *ACM Trans. Graph. 25* (August), 1400–1423.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proc. GI '04*, 239–246.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *Proc. of ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 49–54.

MÜLLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *Proc. Comp. Graph. Int*, 156–165.

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

OTADUY, M. A., GERMANN, D., REDON, S., AND GROSS, M. 2007. Adaptive deformations with fast tight bounds. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 181–190.

RIVERS, A., AND JAMES, D. 2007. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26* (July), 82:1–82:6.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2008. Example-based dynamic skinning in real time. *ACM Trans. Graph. 27*, 29:1–29:8.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 81–90.

SLOAN, P., ROSE, C., AND COHEN, M. 2001. Shape by example. In *Proc. I3D '01*, 135–143.

SUEDA, S., KAUFMAN, A., AND PAI, D. 2008. Musculotendon simulation for hand animation. *ACM Trans.Graph. 27*, 3 (August), 83:1–83:8.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 181–190.

TERZOPOULOS, D., AND WATERS, K. 1990. Physically-based facial modeling, analysis and animation. *J. Vis. Comp. Anim. 1*, 73–80.

TWIGG, C., AND KAČIĆ-ALESIĆ, Z. 2010. Point Cloud Glue: constraining simulations using the procrustes transform. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 45–54.

WANG, C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proc. 2002 ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 129–138.

WU, X., AND TENDICK, F. 2004. Multigrid integration for interactive deformable body simulation. *Med. Sim. 3078*, 92–104.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 16:1–16:18.