VIPER: Volume Invariant Position-based Elastic Rods

BAPTISTE ANGLES, Google, Université de Toulouse, Electronic Arts / SEED, and University of Victoria

DANIEL REBAIN, Google and University of Victoria

MILES MACKLIN, NVIDIA and University of Copenhagen

BRIAN WYVILL, University of Victoria

LOIC BARTHE, Université de Toulouse and IRIT / CNRS

JP LEWIS and JAVIER VON DER PAHLEN, Electronic Arts / SEED

SHAHRAM IZADI, JULIEN VALENTIN, and SOFIEN BOUAZIZ, Google

ANDREA TAGLIASACCHI, Google, University of Waterloo, and University of Victoria

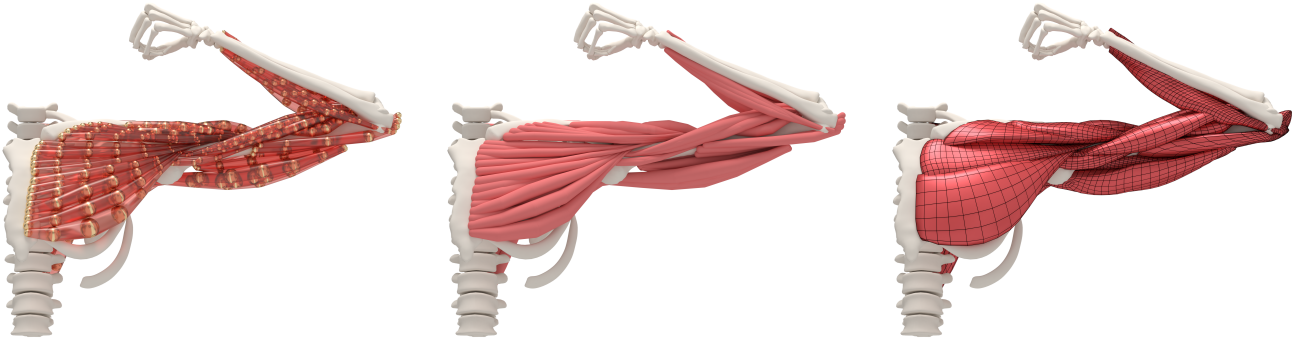


Fig. 1. (left) We compactly model muscles as a collection of generalized rods, where volume conservation is expressed by a radius function defined on curve's vertices – vis sphere's radii. (middle) In the limit, this discretization represents the smooth rods illustrated here, for which physical constraints due to volume invariance can be expressed analytically. (right) The rods create a subspace on which physics is solved, and its effects later propagated to the muscle mesh via linear blend skinning; please see the animation in our **supplemental video**. The "Max" anatomical model is courtesy of ZIVA Dynamics.

We extend the formulation of position-based rods to include elastic *volumetric* deformations. We achieve this by introducing an additional degree of freedom per vertex – isotropic scale (and its velocity). Including scale enriches the space of possible deformations, allowing the simulation of volumetric effects, such as a reduction in cross-sectional area when a rod is stretched. We rigorously derive the continuous formulation of its elastic energy potentials, and hence its associated position-based dynamics (PBD) updates, to realize this model, enabling the simulation with of up to 15000 DOFs at 100 Hz in our GPU implementation. We further show how rods can provide a compact alternative to tetrahedral meshes for the representation of complex muscle deformations, as well as providing a convenient representation for collision detection. This is achieved by modeling a muscle as a *bundle* of rods, for which we also introduce a technique to automatically convert a muscle surface mesh into a rods-bundle. Finally, we show how rods and/or bundles can be skinned to a surface mesh to drive its deformation, resulting in an alternative to cages for real-time volumetric deformation. The source code of our physics engine will be openly available upon publication.

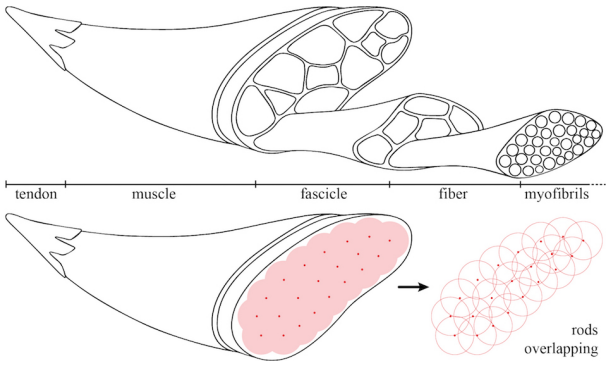
CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Volumetric models**; Collision detection;

Additional Key Words and Phrases: Cosserat rods, soft-body deformation.

ACM Reference Format:

Baptiste Angles, Daniel Rebain, Miles Macklin, Brian Wyvill, Loic Barthe, JP Lewis, Javier von der Pahlen, Shahram Izadi, Julien Valentin, Sofien Bouaziz, and Andrea Tagliasacchi. 2018. VIPER: Volume Invariant Position-based Elastic Rods. 1, 1 (October 2018), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

XXXX-XXXX/2018/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



fig/fibers+ziva/

Fig. 2. (top) Skeletal striated muscle as a collections of nested fascicles, fibres, and myofibrils; base image courtesy of [Lee et al. 2010]. (bottom) We abstract the fascicles as a collection of rods. These can be overlapping, and their rest-pose structure is controlled by *shape-matching* constraints.

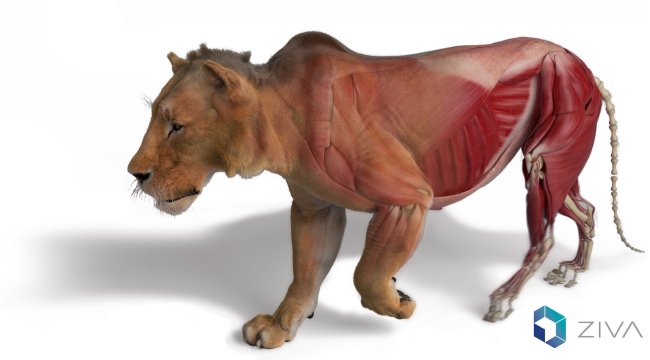


Fig. 3. Bio-mechanically accurate simulation of volumetric anatomical structure is the most effective way to simulate secondary motions (e.g. skin sliding on muscles) and deliver true realistic appearance to dynamic virtual characters; image courtesy of [Ziva Dynamics 2018].

1 INTRODUCTION

In recent years, the computer graphics community has invested exceptional efforts in adapting the (non real-time) physical simulation algorithms at the core of cinematic special effects (e.g.: [Ziva Dynamics 2018]) to the realm of (real-time) interactive applications (e.g.: games, AR/VR). Many of these advancements have been possible thanks to a new class of physics solver, pioneered by Müller et al. [2007], realized on top of *Verlet*-class integrators [Bender et al. 2015]. These *position-based* solvers are capable of elegantly modeling constrained Newtonian dynamics, including rigid-body, cloth, ropes, rods and fluids in a unified framework. A primary example is the NVIDIA FLEX system [Macklin et al. 2014], capable of modeling complex and varied physical phenomena in real-time by leveraging modern GPU hardware.

Rods with volume. Within this technological landscape, of particular relevance to our work is the modeling of elastic "rods" [Kugelstadt and Schömer 2016; Spillmann and Teschner 2007; Umetani et al. 2014]. These models extend "ropes" by augmenting each segment composing the rod with an orthogonal coordinate frame, hence allowing the modeling of *torsion* on top of stretching/bending. Our VIPER rods extend these formulations by accounting for volume preservation, a phenomena not modeled by existing position-based rod models. Many interesting phenomena require this constraint (e.g. soft-bodies, fluids). For example, water is the largest constituent of most animal tissues ($\approx 80\%$ in muscles) hence modeling quasi-incompressible phenomena is of critical importance to achieve believable motion. We address this problem by adding a *per-vertex scaling* degree of freedom – a measure of the local rod cross-section – and optimizing for this quantity within the physics solve. Our rod segments are *hybrid* surface representations, they are *explicitly* parameterized by the position and scale/radius of their vertices, but their surface is defined *implicitly*. This hybrid structure makes them particularly well suited for efficient collision detection/resolution [Green 2010].

Anatomical modeling. Such a physical model not only satisfies our fixation in efficiently simulating rubber bands, but has immediate applications towards the modeling of muscles. As illustrated in Figure 2, *striated skeletal muscles*[1] in human bodies can be represented as a collection of fibers surrounded by connective tissue (i.e. fasciae). Simulating these muscle types efficiently is a fundamental problem, as they represent from 36% to 42% (according to gender) of the average human body mass. In this paper, we propose to efficiently model muscles as a structured collection of volume-preserving rods. This new model can also be interpreted as a generalization of the static volumetric primitives in *Implicit Skinning* [Vaillant et al. 2013], where skin can then be efficiently modeled as a triangular mesh sliding on an implicit iso-surface defined by our fibers. The VIPER primitive is designed to be integrated with other existing components to produce a complete character representation. These include representations of the skeleton, fat, skin, etc.

Volumetric simulation. In the industry, volume-preserving simulation is typically performed by discretizing the interior of the object with tetrahedra or using an approximating cage. However, to the best of our knowledge, even modestly sized models require a lengthy preprocessing (e.g. a large scale eigen-decomposition for computing the deformation modes; see [Barbič and James 2005]) before real-time simulation becomes possible. For example, while visually striking, computing the simulation in Figure 3 requires a two-pass optimization (① muscle, ② skin). ① Muscles are discretized with $51k$ tets sharing $21k$ vertices, and each of their 4 steps of simulation requires 3.2 seconds. ② Skin ($78k$ vertices) is simulated as cloth layered on top of fat (having $68k$ tets sharing $18k$ vertices), where each of the necessary 4 substeps of simulation requires ≈ 35 seconds of compute. Overall, this cumulates to ≈ 2.5 minutes of compute/frame. Offline simulation can be exploited to learn sub-spaces, which then enables dynamic deformations in real-time; see [Xu and Barbič 2016]. However, once learnt, the dynamic behavior of the model is "baked". Clearly, this is an obstacle towards the ultimate

[1] From now on, we will refer to "striated skeletal muscle" simply as "muscles".

goal of truly interactive physics simulation, and, consequently, interactive modeling. The VIPER primitive not only allows the real-time volumetric simulation of complex anatomical structures, but also provides a viable alternative to cages as a compact control structure for soft-body deformations.

Automatic fiber-bundle modeling. Rod-based representations of muscle fascia are not commonly available – typical asset databases contain tetrahedral mesh models instead. While artists sometimes model main characters to the level of interior muscles, this effort is expensive and not justified for background characters. Thus we also introduce a technique to convert existing assets with minimal user intervention. Our solution builds fiber bundles by first creating a set of slices through the muscle then performing an iterative optimization to interpolate these slices with a given number of rods such that they approximate the input surface well.

Contributions. Our fundamental contribution is the design of a novel real-time physics engine for soft-body dynamics. Our system presents several sub-contributions:

- We enrich position-based solvers by introducing a new volume-preserving cosserat rods model and associated constraints.
- We demonstrate how these primitives, when assembled into fiber-bundles, are effective in efficiently modeling muscles.
- We propose a technique for conveniently creating fiber-bundles models from existing simulation assets.
- We introduce the use of VIPER rods as efficient deformation proxies for soft-body deformation.

2 RELATED WORK

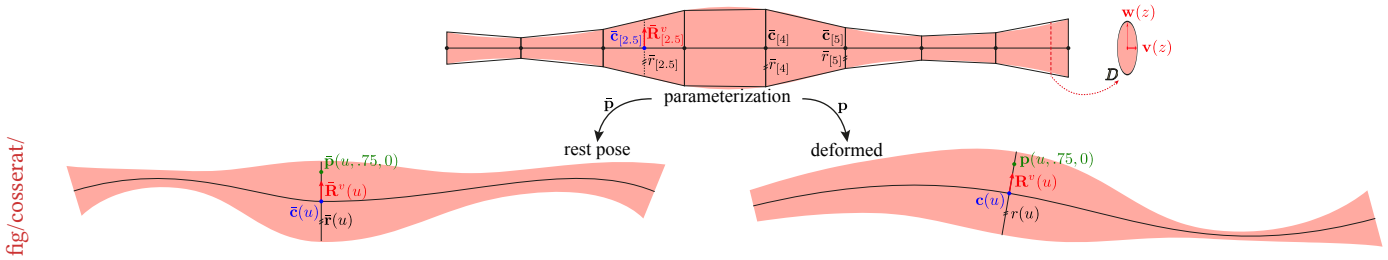
We overview the literature from different angles. We recap example-based modeling frameworks that are commonly used in digital production, as well as recent efforts towards the use of simulated anthropomorphic models. We also review methods that attempt to "emulate" them via geometric processes, and finally processes to calibrate a given model to a target.

Example-based deformation. Digital characters are often modeled via their skin (i.e. *skinning*), with no consideration of underlying volumetric structures, often resulting in non-physically realistic effects such as the *candy wrapper* problem of (LBS) Linear Blend Skinning [Jacobson et al. 2014, Fig.3]. While these artifacts can be resolved [Kavan et al. 2007; Le and Hodgins 2016], skinning solutions lack details such as tendons, muscle bulges, wrinkles, and volume preservation. Example-based approaches such as (PSD) *Pose-Space Deformation* [Kurihara and Miyata 2004; Lewis et al. 2000] and *BlendShapes* [Lewis et al. 2014] interpolate artist-sculpted shapes to *emulate* all these effects. However, "*producing effects such as skin sliding over underlying structures, or the collision effects visible parts of the body press together, can require considerable skill and weeks to months of sculpting depending on the required quality*" [Yuen 2018]. Such data-driven models can be learnt from measurements for both static [Loper et al. 2015], as well as dynamic [Pons-Moll et al. 2015] humans, but they hardly generalize outside of their corresponding training domains.

Physically-based anthropomorphic models. Physically-based simulation of characters has a long history [Scheepers et al. 1997; Sifakis et al. 2005; Terzopoulos and Waters 1990] but, due to its high computational cost, it has only recently began to see practical use. For skeletal muscle deformation, Lee et al. [2010] provides an excellent overview of the field, in regards to which Saito et al. [2015], with its ability to reach *near-interactive runtime*, can be considered the state-of-the-art. Similarly to blendshape generation, training data can be exploited to generate efficient low-dimensional simulations [Bouaziz et al. 2014; Schumacher et al. 2012; Xu and Barbič 2016], enabling physically-based digital characters in production settings [Clutterbuck and Jacobs 2010; Ziva Dynamics 2018]. A shortcoming of these methods is the requirement that the training set encompasses samples of all configurations of the object/character that that will be needed. Physically based approaches also permit a decomposition into *layers* – skin, muscle, fat, and bones – enabling appropriate algorithms to be used for each. Highly relevant to our work is the simulation of skin layered over volumetric primitives pioneered by Li et al. [2013], and its realization in commercial software [Vital Mechanics 2018], as well as other existing research [Saito and Yuen 2017], industry [Ziva Dynamics 2018], and proprietary solutions [Clutterbuck and Jacobs 2010] to this complementary problem. Recently, Romeo et al. [2018] proposed the use of PBD for muscle simulation, but its $\approx$40s/frame of processing time makes it unsuitable to interactive applications.

Non physically-based anthropomorphic models. Recent efforts have been made to create alternative representations for sub-skin volumetric models (i.e. representations of muscle, fat, and bones). For example, Maya Muscle [Comet 2011] represents muscles via NURBS that drive the skin via LBS, whose volume is artist-driven in a PSD fashion. However, due to the *explicit* nature of NURBS, collisions are expensive, hence the performance of the system does not scale well in the complexity of the model. Rather than driving skin via LBS, *Implicit Skinning* [Vaillant et al. 2013] lets it slide on top of implicit surfaces via optimization. These surfaces are defined by blending components that abstract entire body parts (i.e. union of bone, muscle, and fat). In contrast to our work, note that this model is purely kinematic (i.e. no physics). Another relevant class of methods simplifies anthropomorphic components even further. For example representing an entire arm as two *tapered capsules* is advantageous for arm vs. cloth collision detection [Muller 2008, Cloth/Collision]. Sphere-Meshes generalize these representations, and have recently been used to approximate geometry [Thiery et al. 2013], and track its movement in real-time [Tkach et al. 2016].

Calibrating anthropomorphic (volumetric) models. Ali et al. [2013] pioneered the transfer of anatomical structures from a template to a target human via approximate deformation models of soft tissues, and Zhu et al. [2015] calibrated these models from a set of RGBD images capturing a human in motion. Analogously to these method, physically inspired models [Saito et al. 2015] can be calibrated to a set of 3D surface scans [Kadleček et al. 2016]. Of particular relevance to our method is the fiber estimation technique pioneered by Choi and Blemker [2013] employed in [Saito



fig/cosserat/

Fig. 4. (top) The parameterization of a VIPER rod, and its discretization. (left) Its rest configuration, and (right) a deformed configuration. [TODO: notation]

et al. 2015, Sec. 3.1.1]. While Saito et al. [2015] is interested in deriving the anisotropic deformation frame, we require an explicit decomposition of the muscle in fiber bundles.

3 GENERALIZED ROD PARAMETERIZATION

Our physical model of Cosserat rods consists of a smooth parametric curve in 3D space $\mathbf{c}(z):[z_0,z_1]\rightarrow\mathbb{R}^3$. An orthogonal frame $\mathbf{D}(z)\in\mathbb{R}^{3\times3}$ is attached to every point $\mathbf{c}(z)\in\mathbb{R}^3$ on the curve. The orthogonal frame $\mathbf{D}(z)=s(z)\mathbf{R}(z)$ is a combination of a uniform scale $s(u)$, and an orthonormal matrix $\mathbf{R}(z)=[\mathbf{u}(z),\mathbf{v}(z),\mathbf{w}(z)]$; see Figure 4. Note that our model generalizes classical elastic rods [Kugelstadt and Schömer 2016; Spillmann and Teschner 2007], as in those models the scale is kept *constant* along the curve. As shown in Figure 4, any point in the parametrized volume of the rod can be transformed to the rest configuration by a function $\bar{\mathbf{p}}(x,y,z):\mathbb{R}^3\rightarrow\mathbb{R}^3$:

$$\bar{\mathbf{p}}(x,y,z)\equiv\bar{\mathbf{p}}(\mathbf{q},z)=\bar{\mathbf{c}}(z)+\bar{s}(z)\bar{\mathbf{R}}(z)\mathbf{q} \tag{1}$$

where $\mathbf{q}=[x,y,0]^T$ is a point on a disc $D(z)$ of radius $r(z)$, aligned with the xy plane, and centered at its center of mass. Similarly, a second function maps the rod from parameterization to its deformed configuration:

$$\mathbf{p}(x,y,z)\equiv\mathbf{p}(\mathbf{q},z)=\mathbf{c}(z)+s(z)\mathbf{R}(z)\mathbf{q} \tag{2}$$

4 VARIATIONAL IMPLICIT EULER SOLVER

Our solver is based on the variational form of implicit Euler integration [Martin et al. 2011]. The physical model evolves through a discrete set of time samples, with simulation step size h. At time t the deformed volume is defined as $\mathbf{p}_t(x,y,z)$ and the velocity as $\dot{\mathbf{p}}_t(x,y,z)$. The rest pose is defined as $\bar{\mathbf{p}}(x,y,z)=\mathbf{p}_0(x,y,z)$. The mass m is assumed to be uniform over the rod. The sum of the external forces is denoted as $\mathbf{f}_{\text{ext}}(x,y,z)$. We will now drop the indexing (x,y,z) whenever possible to improve readability. We consider position dependent internal forces such that the sum of the internal forces is

$$\mathbf{f}_{\text{int}}(x,y,z)=-\tfrac{1}{2}\textstyle\sum_i\nabla_{\mathbf{p}}\|\mathbf{W}_i(\mathbf{p},\bar{\mathbf{p}})\|^2_{\mathbf{K}_i}, \tag{3}$$

where $\mathbf{W}_i(\mathbf{p},\bar{\mathbf{p}})$ is a potential energy function, and $\mathbf{K}_i$ is a stiffness matrix which we assume to be uniform over the rod, and the notation $\|\mathbf{x}\|^2_{\mathbf{A}}$ means $\mathbf{x}^T\mathbf{A}\mathbf{x}$. We can then write implicit Euler as an optimization describing the compromise between an inertia potential and the elastic potentials:

$$\min_{\{\mathbf{c}_t,s_t,\mathbf{R}_t\}}\int_{z_0}^{z_1}\iint_{D(z)}\underbrace{\tfrac{m}{2h^2}\|\mathbf{p}_t-\hat{\mathbf{p}}_t\|_2^2}_{inertia}+\underbrace{\tfrac{1}{2}\textstyle\sum_i\|\mathbf{W}_i(\mathbf{p}_t,\bar{\mathbf{p}})\|^2_{\mathbf{K}_i}}_{elastic}\,dx\,dy\,dz \tag{4}$$

With $\hat{\mathbf{p}}_t$ we indicate the *inertial prediction* for $\mathbf{p}_t$, i.e., its next position in absence of internal forces:

$$\hat{\mathbf{p}}_t=\mathbf{p}_{t-1}+h\dot{\mathbf{p}}_{t-1}+\tfrac{h^2}{m}\mathbf{f}_{\text{ext}}, \tag{5}$$

where $\dot{\mathbf{p}}(\mathbf{q},z)=\dot{\mathbf{c}}(z)+\left(\dot{s}(z)\mathbf{R}(z)+s(z)\dot{\mathbf{R}}(z)\right)\mathbf{q}$.

Discretization. We discretize the curve in the parametrized domain using a set of $m+1$ points $\{z_{[0]},\ldots,z_{[m]}\}$ connected using m *piecewise linear elements* of length $\{l_{[1]},\ldots,l_{[m]}\}$; see Figure 4. We can approximate the curve integral by integrating over these piecewise linear elements using the midpoint rule. For the integration we also define a set of m midpoints $\{z_{[.5]},\ldots,z_{[m-.5]}\}$. A point on a midpoint cross section is then parametrized as:

$$\mathbf{p}(\mathbf{q},z_{[j-.5]})=\mathbf{c}(z_{[j-.5]})+s(z_{[j-.5]})\mathbf{R}(z_{[j-.5]})\mathbf{q} \tag{6}$$

$$\mathbf{c}(z_{[j-.5]})\equiv\tfrac{1}{2}\left[\mathbf{c}(z_{[j-1]})+\mathbf{c}(z_{[j]})\right] \tag{7}$$

$$s(z_{[j-.5]})\equiv\tfrac{1}{2}\left[s(z_{[j-1]})+s(z_{[j]})\right] \tag{8}$$

This is similar to the staggered grid discretization of previous work [Grégoire and Schömer 2006; Spillmann and Teschner 2007], where the frames $\mathbf{R}$ are stored at the *midpoints*. Contrary to previous approaches, our model also has a scale that we store at the *endpoints* of the linear elements. We can now rewrite Equation 4 as:

$$\min_{\{\mathbf{c}_t,s_t,\mathbf{R}_t\}}\sum_{j=1}^{m}l_{[j]}\iint_{D(z_{[j-.5]})}\tfrac{m}{2h^2}\|\mathbf{p}_t-\hat{\mathbf{p}}_t\|_2^2+\tfrac{1}{2}\textstyle\sum_i\|\mathbf{W}_i(\mathbf{p}_t,\bar{\mathbf{p}})\|^2_{\mathbf{K}_i}\,dx\,dy \tag{9}$$

5 ELASTIC POTENTIALS

In this section we detail the elastic potentials used to simulate our volume preserving rods.

Strain potential. We define the strain at a midpoint as

$$\mathrm{E}_{\text{strain}}(z_{[j-.5]})=\iint_{D(z_{[j-.5]})}\|\mathbf{R}^T\nabla\mathbf{p}-\bar{\mathbf{R}}^T\nabla\bar{\mathbf{p}}\|^2_{\mathbf{K}_s}\,dx\,dy, \tag{10}$$

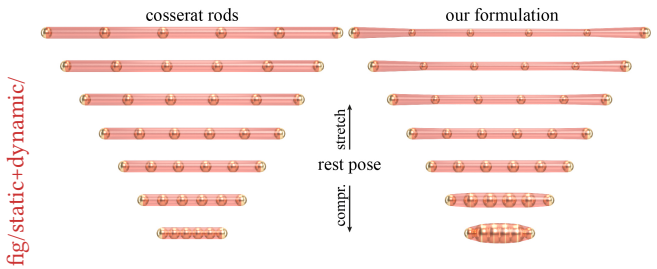


Fig. 5. **Static behavior** – We compare the deformation of a standard cosserat rod to our volumetric invariant version. Our additional degrees of freedom allow us to model the buckling (resp. bulging) caused by the stretching (resp. compression) of the rod.

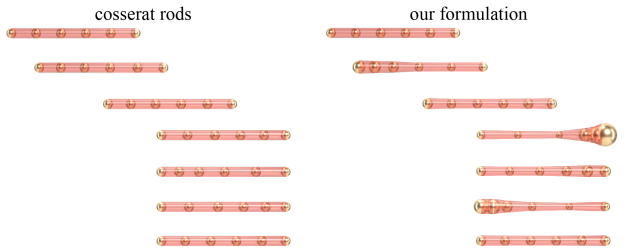


Fig. 6. **Dynamic behavior** – Our generalized rods do not only capture static volumetric deformations, but the scale's velocity allows us to model volume dynamics. In this example, note the rod length is unchanged, but our formulation models a volumetric shockwave travelling through the rod.

where $\mathbf{K}_s = [\, k_s^x \mathbf{e}^x \;\; k_s^y \mathbf{e}^y \;\; k_s^z \mathbf{e}^z \,]$ is a diagonal stiffness matrix and $[\mathbf{e}^x \mathbf{e}^y \mathbf{e}^z]$ is the standard basis. $\nabla \mathbf{p}$ and $\nabla \bar{\mathbf{p}}$ denote the deformation gradients, i.e., the Jacobian matrices of the deformation functions [Sifakis and Barbic 2012]. As $\mathbf{p}$ and $\bar{\mathbf{p}}$ map $\mathbb{R}^3$ to $\mathbb{R}^3$, the Jacobian matrices are 3×3. The Jacobians are not rotational invariant so we rotate them back to the parametrization domain to be able to compare them on a common ground. As explained in Appendix A integrating the strain energy (10) leads to

$$\mathrm{E}_{\mathrm{strain}}(z_{[j-.5]}) = \pi r^2 k_s^z \|\nabla_z \mathbf{c} - \mathbf{w}\bar{\mathbf{w}}^T \nabla_z \bar{\mathbf{c}}\|_2^2 \tag{11}$$

$$+ \pi r^2 (k_s^x + k_s^y)(s - \bar{s})^2 \tag{12}$$

$$+ \tfrac{1}{4}\pi r^4 (k_s^x + k_s^y)(\nabla_z s - \nabla_z \bar{s})^2 \tag{13}$$

$$+ \|s\boldsymbol{\Omega} - \bar{s}\bar{\boldsymbol{\Omega}}\|_{\mathbf{H}_s}^2, \tag{14}$$

where $\mathbf{H}_s = [\, \pi r^4 k_s^z \mathbf{e}^x \;\; \pi r^4 k_s^z \mathbf{e}^y \;\; \pi r^4 (k_s^x + k_s^y) \mathbf{e}^z \,]$ is the second moment of area of a disc scaled by the stiffness, and the Darboux vector [Kugelstadt and Schömer 2016] is denoted by $\boldsymbol{\Omega} = [\Omega^u, \Omega^v, \Omega^w]^T$. Note that we retrieved similar energies in previous works augmented by our additional scale degree of freedom. The energies (11) and (12) respectively measure the stretch along the curve and the cross section, while (13) measures the variation of scale across sections, and (14) measures bending/twisting. Interestingly, Equation 13 can also be interpreted as a measure of surface stretch.

We use an additional energy measuring the second order variation of scale complementing (13) with a measure of surface *bending*

$$\mathrm{E}_{\mathrm{bending}}(z_{[j-.5]}) = \tfrac{1}{4}\pi r^4 \left(k_b^x + k_b^y\right) \left(\nabla_z^2 s - \nabla_z^2 \bar{s}\right)^2 \tag{15}$$

where $\nabla^2 = \nabla \cdot \nabla = \Delta$ is the Laplacian operator. Note that this energy can also been seen as an *approximation* of:

$$\iint_{D(z_{[j-.5]})} \|\mathbf{R}^T \nabla^2 \mathbf{p} - \bar{\mathbf{R}}^T \nabla^2 \bar{\mathbf{p}}\|_{\mathbf{K}_b}^2 \, dx dy, \tag{16}$$

where $\mathbf{K}_b = \left[\, k_b^x \mathbf{e}^x \;\; k_b^y \mathbf{e}^y \;\; k_b^z \mathbf{e}^z \,\right]$. This energy compares the Laplacians of the deformation functions giving a second order measure of the deformation.

Volume potential. By denoting the determinant with $|\cdot|$, and stiffness by k_v, we define the volume preservation at midpoints as:

$$\mathrm{E}_{\mathrm{vol}}(z_{[j-.5]}) = \iint_{D(z_{[j-.5]})} k_v \left(|\nabla \mathbf{p}| - |\nabla \bar{\mathbf{p}}|\right)^2 dx \, dy \tag{17}$$

Note that the determinant is rotational invariant, hence it is not necessary to rotate the Jacobians. As explained in Appendix A integrating the volume energy (17) leads to:

$$\mathrm{E}_{\mathrm{vol}}(z_{[j-.5]}) = \pi r^2 k_v \|s^2 \nabla_z \mathbf{c} - \bar{s}^2 \mathbf{w}\bar{\mathbf{w}}^T \nabla_z \bar{\mathbf{c}}\|_2^2 \tag{18}$$

$$+ \tfrac{1}{2}\pi r^4 k_v \left(s^3 \Omega^u - \bar{s}^3 \bar{\Omega}^u\right)^2 \tag{19}$$

$$+ \tfrac{1}{2}\pi r^4 k_v \left(s^3 \Omega^v - \bar{s}^3 \bar{\Omega}^v\right)^2. \tag{20}$$

6 OPTIMIZATION

To approach the non-linear optimization problem in (9), we linearize the inertia and non-linear elastic terms, and then solve the optimization iteratively in a Gauss-Newton fashion. To warm start the optimization, we first compute a *prediction step* by ignoring the elastic potentials and by solely minimizing the inertia term – this simply provides an initial guess. We then compute a (set of) *correction steps* that also include the elastic potentials.

Prediction step. By denoting by $\boldsymbol{\theta}$ the angles parametrizing the rotation matrix and $\mathcal{I}$ is the second moment of area of a disc in world-space, the predictions for the different degrees of freedom of our model are computed as:

$$\hat{\mathbf{c}} = \mathbf{c} + h\dot{\mathbf{c}} + \frac{h^2}{\pi r^2 m} \boldsymbol{\xi}_{\mathrm{ext}}, \qquad \text{(center prediction)} \tag{21}$$

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + h\dot{\boldsymbol{\theta}} + \frac{\mathcal{I}^{-1} h^2}{s^2 m} \boldsymbol{\tau}_{\mathrm{ext}}, \qquad \text{(frame prediction)} \tag{22}$$

$$\hat{s} = s + h\dot{s} + \frac{2h^2}{\pi r^4 m} \boldsymbol{\gamma}_{\mathrm{ext}}, \qquad \text{(scale prediction)} \tag{23}$$

where $\boldsymbol{\xi}_{\mathrm{ext}}$ is the sum of the external forces which act on the disc, $\boldsymbol{\tau}_{\mathrm{ext}}$ is the sum of the external torques, and $\boldsymbol{\gamma}_{\mathrm{ext}}$ is a quantity which can be seen as the counterpart of the total external torque measuring the sum of the external forces projected on the position vectors; see Appendix B for more details. Note that the center and frame predictions are similar to the rigid-body equations of motion for a

fig/octopus/

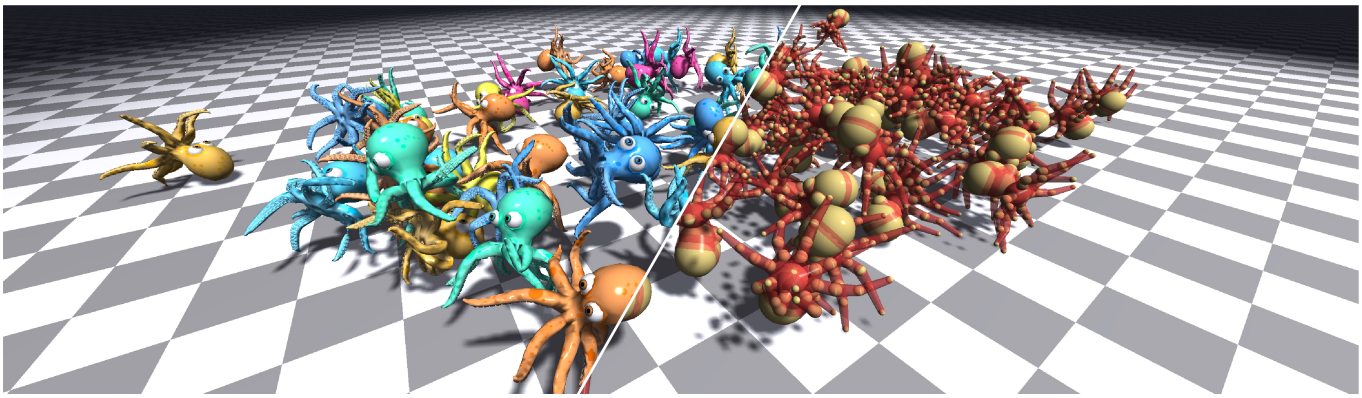


Fig. 7. Being built on VIPER primitives, our physics engine can simulate soft-body deformation and dynamic interactions between hundres of models in *real-time*. A peculiarity of our engine is that *both* collision and simulation are executed on the *same* geometry; see our video in the **additional material**.

stretched disc. On top of these equations, we get a scale prediction describing how the scale of the disc is affected by the velocity and the external forces.

Correction steps. We then compute a set of correction steps including both inertial/elastic terms. We define

$$\mathbf{X} = [\mathbf{c}_{[0]}^T, s_{[0]}, \boldsymbol{\theta}_{[.5]}^T, \mathbf{c}_{[1]}^T, s_{[1]}, \boldsymbol{\theta}_{[1.5]}^T, \cdots]^T \tag{24}$$

as the vector containing all the degrees of freedom, and $\boldsymbol{\lambda}$ as the vector of Lagrange multipliers. $\mathbf{K}$ is a block diagonal matrix containing the stiffness parameters multiplied by the length of the piecewise elements, $\mathbf{A}$ is a block diagonal matrix stacking the inertia weights multiplied by the length of the piecewise elements, and

$$\mathbf{W}(\mathbf{X}) = [\mathbf{W}_1(\mathbf{X}), \mathbf{W}_2(\mathbf{X}), \cdots]^T \tag{25}$$

stacks the potential energy functions. Denoting the iteration number with k, the the state is then updated as $\mathbf{X}^k = \mathbf{X}^{k-1} + \Delta\mathbf{X}$ and $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1} + \Delta\boldsymbol{\lambda}$, where, as derived in Appendix C:

$$\Delta\mathbf{X} = -h^2\mathbf{A}^{-1}\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\Delta\boldsymbol{\lambda}, \tag{26}$$

$$\Delta\boldsymbol{\lambda} = \left(h^2\|\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\|^2_{\mathbf{A}^{-1}} + \mathbf{K}^{-1}\right)^{-1}\mathbf{W}(\mathbf{X}^{k-1}) - \mathbf{K}^{-1}\boldsymbol{\lambda}^{k-1}. \tag{27}$$

For realtime performance we opt for using an iterative linear system solver such as block Jacobi or Gauss-Seidel. The update for the i-th constraint is:

$$\Delta\boldsymbol{\lambda}_i = \beta\left(h^2\|\mathbf{W}_i(\mathbf{X})^T\|^2_{\mathbf{A}^{-1}} + \mathbf{K}_i^{-1}\right)^{-1}\left(\mathbf{W}_i(\mathbf{X}) - \mathbf{K}_i^{-1}\boldsymbol{\lambda}_i\right) \tag{28}$$

where we dropped the superscripts to improve readability, and β is a relaxation parameter. Note that $\mathbf{A}$ being a block diagonal matrix with block of size at most 3×3, $\mathbf{A}^{-1}$ can be efficiently computed. Note that Equation 28 is a generalization of the XPBD update [Macklin et al. 2016] derived for our volume preserving rod model.

7 REAL-TIME PHYSICS ENGINE

We implemented our real-time solver on a GPU by leveraging the *Thrust* framework provided by the CUDA library, which we execute on a single NVIDIA GTX 1080 graphics card. In our engine, any volumetric object is modeled as a collection of tapered capsule primitives, or "pills" (for brevity), while the floor is represented as a simple halfplane constraint. During simulation, at each time step, we start by first animating kinematic objects (e.g. bones). We then compute the inertial predictions, and perform collision detection (Sec. 7.1) to generate collision resolution constraints (Sec. 7.2). We then solve for all constraints using a Jacobi solver: constraints are computed in parallel, and the resulting positional displacements are averaged out by a reduction. The transformations of VIPERs can then be skinned to any surface mesh model (Sec. 7.4).

Our model provides a viable alternative to cages as a compact control structure for soft-body deformations. We demonstrate this in Figure 7, where we rigged a simple octopus character using rods, and solve for collisions and soft-body deformations in real time. As shown in the accompanying video, our *non-optimized* prototype achieves real-time performance (≈??*ms* sim, ≈??*ms* render) for scenes containing up to 100 octopuses, each rigged with 37 pills, whose deformation is skinned to a triangular surface mesh of ≈ 13k faces. Note how for robust collision detection we need far fewer pills than the volumetric particles used in Macklin et al. [2014].

7.1 Collision detection

To detect collisions between physical primitives, we adopt the approach presented by Green [2010] popularized in the context of GPU particle fluid simulation. Towards this goal, we approximate each pill by its bounding sphere, and (conservatively) detect collisions to be later handled in the resolution loop detailed in Section 7.2. Collisions are detected with the assistance of a uniform grid with cell width chosen as the diameter of the largest bounding sphere, such that collisions between spheres centered in non-neighboring cells are impossible. Extending [Green 2010], we also count the spheres in the neighboring cells of each particle, and use this information to construct *in parallel* a list of all potential collisions. This is in contrast to the original algorithm which for each sphere processes neighbors in series, and therefore degrades to a partially serial algorithm in cases where many particles fall into a single grid cell. Further details regarding this process are provided together

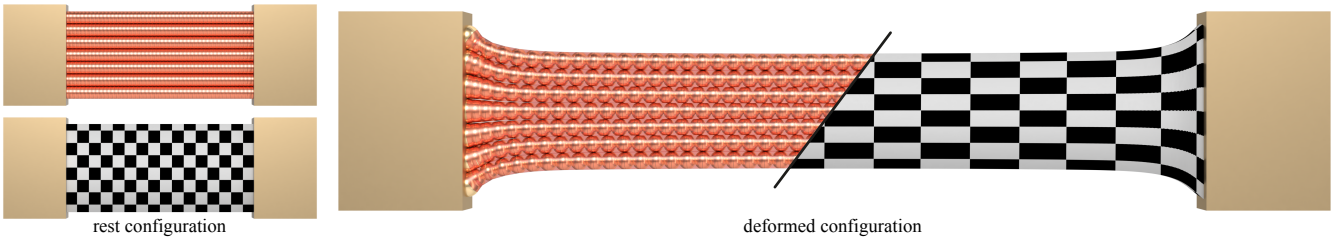


Fig. 8. (left) An elastic band mesh and its VIPER discretization. (right) The VIPER simulation and their deformation "skinned" to rest-pose mesh.

with an executable 1D example in the form of a *Jupyter notebook* in our additional material.

7.2 Collision handling

In a generic physics engine one would implement collision detection/response between any pair of available proxies. For the sake of efficiency, in our framework we only tackle two collision proxies: (kinematic) *half-planes* and (dynamic) *pills*. Within the combinatorial set of collision pairs, the main challenge is pill-to-pill collisions. In what follows, we first compute the meta-parameters of the collisions, that are then resolved in the optimization via PBD constraints [Müller et al. 2007].

Collision metadata. Given two pills $\mathbf{P}_a$ and $\mathbf{P}_b$, each modeled as a pair of spheres, for example, $\mathbf{P}_a = \{(\mathbf{c}_1^a, r_1^a), (\mathbf{c}_2^a, r_2^a)\}$, the fundamental queries we need to answer are: ① is there a collision? ② what is the collision point/normal? ③ what is the inter-penetration amount? As typical in efficient collision resolution, we introduce a *single* PBD constraint modeling collision forces corresponding to the *largest* pill-to-pill inter-penetration. In our solution, we leverage the geometric structure of the problem: ① a pill can be interpreted as the union of infinitely many spheres whose position and radii are linearly interpolated between its endpoints; ② the largest inter-penetration corresponds to the inter-penetration between any pair of spheres, one in pill $\mathbf{P}_a$ and one in pill $\mathbf{P}_b$. By first defining the LERP operator as $\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \gamma) \equiv (1-\gamma)\mathbf{x}_1 + \gamma\,\mathbf{x}_2$, the interpolated sphere $(\mathbf{c}(\gamma), r(\gamma))$ is derived by LERP'ing the endpoint quantities as $\mathbf{c}(\gamma) = \mathcal{L}(\mathbf{c}_1, \mathbf{c}_2, \gamma)$ and $r(\gamma) = \mathcal{L}(r_1, r_2, \gamma)$. The largest inter-penetration is then given by the solution of the *bivariate* optimization problem:

$$\arg\max_{\alpha,\beta} \quad \|\mathbf{c}_a(\alpha) - \mathbf{c}_b(\beta)\|_2 - (r_a(\alpha) + r_b(\beta)) \tag{29}$$

Because a closed-form operator $\Pi_b(\alpha)$ providing the barycentric coordinate of the closest-point projection of a point $\mathbf{c}_a(\alpha)$ onto the pill $\mathbf{P}_b$ is available (see Appendix D) we can further simplify this problem into a *scalar* optimization problem:

$$\arg\max_{\alpha} \quad \|\mathbf{c}_a(\alpha) - \mathbf{c}_b(\Pi_b(\alpha))\|_2 - (r_a(\alpha) + r_b(\Pi_b(\alpha))) \tag{30}$$

which we solve by Dichotomous Search [Antoniou and Lu 2007] with a fixed number of iterations (set to 10 in our engine).

Collision constraints. Having detected a collision between two pills $\mathbf{P}_a$ and $\mathbf{P}_b$, and having computed α^* (and hence $\beta^* = \Pi_b(\alpha^*)$) by solving (30), we can express a constraint that correlates radii and positions on the two pills in order to resolve the collision in a least squares sense:

$$E_{\text{collision}} = \|\|\mathbf{c}_a(\alpha^*) - \mathbf{c}_b(\beta^*)\|_2 - \big(r_a(\alpha^*) + r_b(\beta^*)\big)\|_2^2. \tag{31}$$

7.3 Scale-invariant shape matching – "Bundling"

To represent more complex geometry than individual rods, such as that in Figure 2, we can gather a collection of rods in a cross-section, and introduce a constraint to explain their joint deformation. We employ the assumption that muscle fibers in a muscle cross-section contract isotropically. Indexing the rods in a cross-section by i, our rod deformation model can be expressed as a similarity transform

$$\mathbf{T}_i = \begin{bmatrix} s_i\mathbf{R}_i & \mathbf{c}_i \\ \mathbf{0} & 1 \end{bmatrix}. \tag{32}$$

As illustrated in Figure 9, for each muscle cross-section we define a scale invariant shape-matching energy measuring the deviation of the current rod deformations $\mathbf{T}_i$ from a global similarity transform $\mathbf{T}^*$ of the rest deformations $\bar{\mathbf{T}}_i$ as

$$E_{\text{shape}} = \sum_i \|\mathbf{T}^*\bar{\mathbf{T}}_i - \mathbf{T}_i\|_2^2. \tag{33}$$

We treat this energy as a hard constraint by finding the optimal $\mathbf{T}^*$ and setting $\mathbf{T}_i = \mathbf{T}^*\bar{\mathbf{T}}_i$ as a post-processing step after few iterations. The optimal $\mathbf{T}^*$ can be computed following the derivation in [Umeyama 1991]. The optimal rotation $\mathbf{R}^*$ can be found by solving

$$\mathbf{R}^* = \arg\min_{\mathbf{R}\in SO(3)} \|\mathbf{R} - \Sigma_i[s_i\mathbf{R}_i \quad \mathbf{c}_i - \boldsymbol{\mu}][\bar{s}_i\bar{\mathbf{R}}_i \quad \bar{\mathbf{c}}_i - \bar{\boldsymbol{\mu}}]^T\|_2^2, \tag{34}$$

where $\boldsymbol{\mu} = \frac{1}{n}\Sigma_i\mathbf{c}_i$. We compute the optimal rotation $\mathbf{R}^*$ using the robust approach presented in [Müller et al. 2016]. The optimal scale can be computed as

$$s^* = \frac{\sum_i \text{sum}(\mathbf{R}^*[\bar{s}_i\bar{\mathbf{R}}_i \quad \bar{\mathbf{c}}_i - \bar{\boldsymbol{\mu}}] \circ [s_i\mathbf{R}_i \quad \mathbf{c}_i - \boldsymbol{\mu}])}{\sum_i \text{sum}([\bar{s}_i\bar{\mathbf{R}}_i \quad \bar{\mathbf{c}}_i - \bar{\boldsymbol{\mu}}] \circ [\bar{s}_i\bar{\mathbf{R}}_i \quad \bar{\mathbf{c}}_i - \bar{\boldsymbol{\mu}}])}, \tag{35}$$

where $\text{sum}(\cdot)$ adds all entries of the matrix and $\circ$ is the Hadamard product. Finally, the optimal translation can be derived as

$$\mathbf{c}^* = \boldsymbol{\mu} - s^*\mathbf{R}^*\bar{\boldsymbol{\mu}}. \tag{36}$$

7.4 Skinning VIPERs to surface deformation

Our VIPER rods can also be employed as a volumetric *proxy* driving the deformation of a surface mesh; see Figure 1 and Figure 8. In particular, we blend the relative transformations between deformed

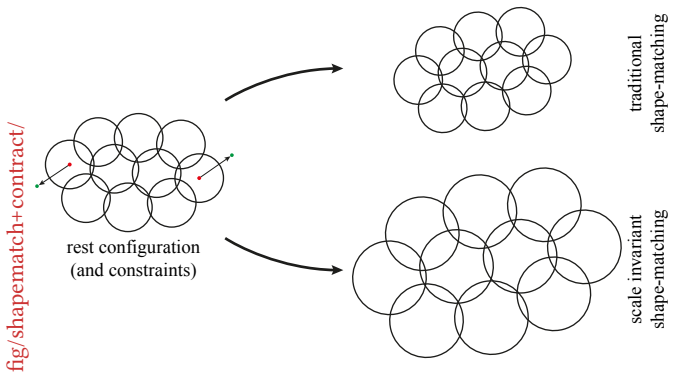


Fig. 9. **Scale-invariant shape matching** – Shape matching can recover a rigidly transformed configuration, while our model allows for a null-space that includes uniform scale. We employ this model as we work under the assumption that muscle fibers in a muscle cross-section contract isotropically.

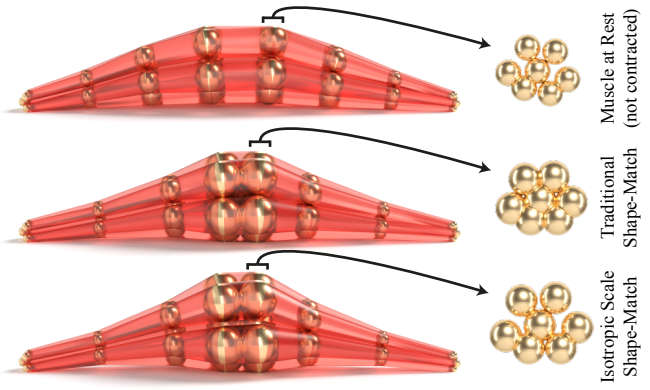


Fig. 10. **Muscle contraction** – (top) Muscle at rest and its excitation (force shortening edge lengths in an area) with traditional shape-matching constraints (middle) vs our novel scaled shape-matching constraints (bottom).

and rest pose configurations via *linear blend skinning*. To achieve this, we utilize a modified version of the LBS weight computation by Thiery et al. [2013]. In that paper, weights were computed by fairing an initial assignment where each vertex was *fully* attached to the nearest element. Instead, we modify this assignment to begin with weights for each vertex that are proportional to the inverse square-distance from the vertex to the surface of each pill. This resolves cases where multiple surfaces are at equal distance, and the nearest neighbor is multiply-defined. Further, when bundling is employed (Section 7.3), we do not employ the rod frames, but rather we employ the one provided by the optimization in Equation 33.

7.5 Energy implementation

The energies derived in Section 5 have been derived using continuous operators. To implement these energies we approximate $\nabla_z \mathbf{c}(z_{[j+.5]})$, $\nabla_z s(z_{[j+.5]})$, and $\nabla_z^2 s(z_{[j+.5]})$ using finite difference such that

$$\nabla_z \mathbf{c}(z_{[j+.5]}) = l_{[j]}^{-1}\left(\mathbf{c}(z_{[j+1]}) - \mathbf{c}(z_{[j]})\right), \tag{37}$$

$$\nabla_z s(z_{[j+.5]}) = l_{[j]}^{-1}\left(s(z_{[j+1]}) - s(z_{[j]})\right), \tag{38}$$

$$\begin{aligned}\nabla_z^2 s(z_{[j]}) = {} & l_{[j]}^{-1}\left(s(z_{[j+1]}) - s(z_{[j]})\right) \\ & - l_{[j-1]}^{-1}\left(s(z_{[j]}) - s(z_{[j-1]})\right).\end{aligned} \tag{39}$$

For simplicity of the derivations we used angles $\boldsymbol{\theta}$ to parametrize the rotation $\mathbf{R}$. However, our implementation uses quaternions. For small angles the corresponding quaternion is $\mathbf{Q} = [\frac{\boldsymbol{\theta}}{2}^T, 1]^T$. We also use quaternions to represent rotations and approximate the Darboux vector using

$$\Omega(z_{[j]}) = 4(l_{[j-1]} + l_{[j]})^{-1}\,\mathrm{Im}(\bar{\mathbf{Q}}_{[j-.5]}\mathbf{Q}_{[j+.5]}), \tag{40}$$

where $\mathrm{Im}(\cdot)$ gives the imaginary part of a quaternion.

8 ANATOMICAL MODELING AND SIMULATION

We now describe how our rods can be used to model complex anatomical structures such as *bones* and *muscles*; see Figure 1. For the former, we use a simplified version of Thiery et al. [2013] where only pill primitives are used – this primarily allows us to reduce the complexity of the collision detection/resolution codebase. We then detail the conversion of digital models of muscles into VIPERs in Section 8.1, and describe a few nuances about the simulation of their motion in Section 8.2.

8.1 Muscles to rods conversion – "Viperization"

We developed a (weakly-assisted) technique to convert a traditional muscle model into a collection of K rods. As outlined in Figure 11, our process involves several phases. We begin by computing a volumetric discretization of the muscle's surface. We then ask the artist to paint annotations on the surface marking the start (sources) and the end (sinks) of the muscle. A harmonic solve alike the one describe in Choi and Blemker [2013] is the executed to compute a field that smoothly varies in the $[0, 1]$ range along the muscle's length. We then sample M iso-levels of this field to be surfaces containing the M vertices of *each* of the K generated VIPERs. Within each iso-level we require: ① VIPERs to have the same radii, and ② to be distributed uniformly on the slice (iso-surface). To obtain this, we perform a *restricted centroidal Voronoi diagram* of K points on each slice [Botsch et al. 2010], while simultaneously penalizing the length of each rod. We alternate this variational optimization with a discrete one that re-assigns spheres to different rods in order to minimize the sum of rod lengths. This process, which we refer to as "combing", starts from one end of the muscle, and executes in order $M-1$ instances of minimum-cost bipartite matching (which we solve in polynomial time via the Hungarian algorithm), where the pairwise costs are the K^2 euclidean distances across rod nodes in two adjacent slices. A few examples of the conversion process are illustrated in Figure 1 and Figure 12.

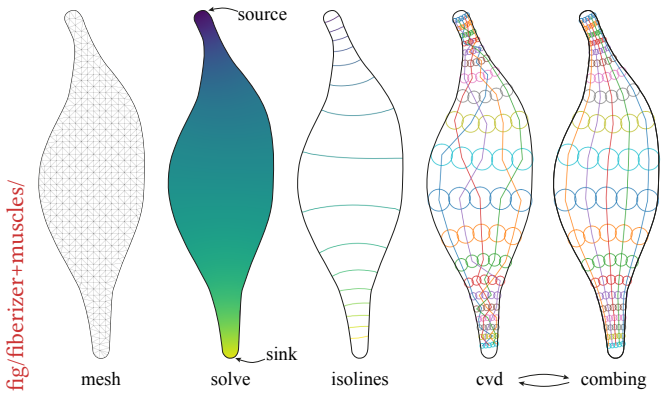


Fig. 11. The mesh-to-VIPER conversion process. Given source/sink constraints, we compute a harmonic function in the volume, and extract a few discrete iso-levels. Within each of these, we execute a restricted CVD to place 5 elements of the same radii on these surfaces. We then execute a combinatorial optimization that connects samples across layers to produce minimal length curves.

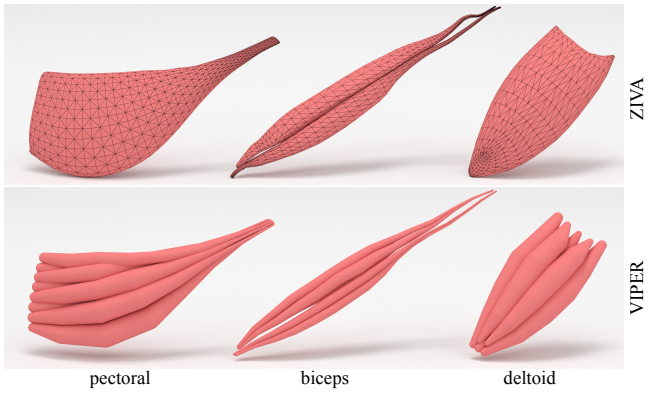


Fig. 12. We illustrate several examples of the VIPERs extracted by our automated process. For the *pectoral*, our VIPER model employs 8 rods, each discretized by 8 elements. In comparison, the surface meshes by ZIVA contain ($|\partial V| = 1584$, $|\partial F| = 3164$) on the boundary and its (volumetric) simulation mesh contains ($|V| = 350$, $|T| = 1089$) tetrahedral elements. Where $\partial \equiv$ "boundary", $F \equiv$ "faces", $V \equiv$ "vertices" and $T \equiv$ "tetrahedra".

8.2 Muscle simulation

Once a muscle is *viperized* as described in Section 8.1, rod centers within the same cross-section are connected via the *bundling* constraints in Equation 33. Every rod also obeys the constraints described in Section 5. The endpoints of rods are kinematically attached to bones, and intra-muscle collisions are disabled, while inter-muscle collisions are detected and resolved as described respectively in Section 7.1 and Section 7.2. Muscles can also be *excited/contracted* by inserting internal forces, or even simply shortening the length of fibers (while preserving volume) resulting in the bulging effects illustrated in Figure 10. Lo - What about: Muscles can also be *activated* (fiber contractions generating a stronger force, producing a change of shape at constant muscle length and volume) by inserting internal forces, or even simply shortening the length of fibers resulting in the bulging effects illustrated in Figure 10. We also speed-up the solver convergence during fast motion by exploiting the availability of bone transformations. In more detail, each muscle particle has two skinning weights corresponding to the two bones the muscle is attached to. At the beginning of each frame we use the transform of each bone relative to their last frame's transforms to initialize the displacement of the particle using LBS, and later refine this via simulation.

9 CONCLUSIONS & FUTURE WORK

In this paper, we introduced a novel formulation of cosserat rods that considers local volume, and optimizes for its local conservation. The resulting position-based simulation is highly efficient, and is straightforward to implement on graphics hardware. We demonstrated how rod-bundling is a powerful representation for the modeling of volumetric deformation – and in particular for skeletal muscles. Rather than requiring artists to model from scratch, we also introduced an algorithm to procedurally generate VIPERs with minimal user interaction. Finally, by coupling the rod simulation to a surface mesh via skinning, our model can be thought of as a direct alternative to tetrahedral meshes and cages for real-time non-rigid deformation. Most importantly, our generalized rods formulation opens up a number of venues for future work, which we classify in three broad areas, as elaborated below.

Model generalization. While in our rods we discretized the skeletal curve with piecewise linear elements, it would be interesting to investigate whether the use of *continuous* curve models such as splines would be tractable – from both mathematical, as well as implementation standpoints. Similarly to [Müller and Chentanez 2011; Müller and Chentanez 2011], our model could also be extended to model *anisotropic* volume deformations, that is, both the rest pose and deformed rod could have a non-circular cross section. Further, while in this paper we treated the modeling of non-circular cross-sections via bundling, the theory of medial axis [Tagliasacchi et al. 2016] tells us how any geometric object can be approximated via primitives formed as the convex-hulls of three-spheres – what Tkach et al. [2016] called "wedges". Extending our volume-invariant rods models to volume-invariant *wedges* would provide an elegant generalization of our modeling paradigm.

Anatomical modeling. As highlighted by our supplementary video, rod primitives can be exploited for the efficient approximate modeling and simulation of complex structures. Nonetheless, the dynamics of the human body are the result of the complex *interplay* between muscle, fat, and the consequent deformation of skin. Enriching our model to also account for these factors would be an interesting extension. For example, rather than driving the muscle surface via skinning as in Section 7.4, one could represent a muscle as a controllable *implicit blend* [Angles et al. 2017], approximate fat as an elastic offset between muscles and skin, and *simulate skin* as an elastic surface whose vertices lie on a (potentially) user-controlled iso-level of the implicit function. Further, while artistical editing of physically driven anatomical systems can be a difficult due to the

complexity of simulation, our framework could immediately enable *interactive modeling*, in a similar fashion to what ZBrush/ZSphere currently provides for authoring static geometry. By extending the works in [Tkach et al. 2016, 2017], an efficient anatomical model for a particular user could also be constructed by fitting to RGBD data.

Optimization. Our solver has not *yet* directly leveraged the straightforward multi-resolution structure of rod geometry. More specifically, the curve parameterization of rods offers a domain over which designing prolongation/restriction operators needed for a *geometric multi-grid* implementation becomes straightforward. An orthogonal dimension for optimization would be to consider the existence of multi-resolution structures within the *cross-sectional* domain; see Figure 2. This could both be exploited in offering multi-scale interaction for artists in editing our rod models, as well as to produce level-of-details models for efficient simulation at scale. Finally, while we employed out-of-the-box geometry processing tools to convert a triangular surface mesh into a rod model, we believe fitting a fiber-bundle model to a given solid could be achieved without the (often finicky) conversion to volume/tetrahedral mesh, but rather as a direct optimization over fiber placements.

REFERENCES

Dicko Hamadi Ali, Tiantian Liu, Benjamin Gilles, Ladislav Kavan, François Faure, Olivier Palombi, and Marie-Paule Cani. 2013. Anatomy transfer. *ACM Trans. on Graphics* (2013).

Baptiste Angles, Marco Tarini, Loic Barthe, Brian Wyvill, and Andrea Tagliasacchi. 2017. Sketch-Based Implicit Blending. *ACM TOG (Proc. SIGGRAPH Asia)* (2017).

Andreas Antoniou and Wu-Sheng Lu. 2007. *Practical Optimization: Algorithms and Engineering Applications*.

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics* (2005).

Jan Bender, Matthias Müller, and Miles Macklin. 2015. Position-Based Simulation Methods in Computer Graphics.. In *Proc. Eurographics (Technical Course Notes)*.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. 2010. *Polygon mesh processing*. AK Peters/CRC Press.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. on Graphics* (2014).

Hon Fai Choi and Silvia S Blemker. 2013. Skeletal muscle fascicle arrangements can be reconstructed using a laplacian vector field simulation. *PloS one* (2013).

Simon Clutterbuck and James Jacobs. 2010. A Physically Based Approach to Virtual Character Deformations. In ACM SIGGRAPH Talk sessions.

Michael Comet. 2011. Maya Muscle. http://download.autodesk.com/us/support/files/muscle.pdf. (Accessed on Aug. 8th, 2018).

Nicholas Ian Mark Gould. 1986. On the accurate determination of search directions for simple differentiable penalty functions. *IMA J. Numer. Anal.* (1986).

Simon Green. 2010. Particle simulation using CUDA. *NVIDIA whitepaper* (2010).

Mireille Grégoire and Elmar Schömer. 2006. Interactive Simulation of One-dimensional Flexible Parts. In *Proc. of ACM Symposium on Solid and Physical Modeling*.

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J.P. Lewis. 2014. Skinning: Real-time Shape Deformation. SIGGRAPH Course, http://skinning.org/direct-methods.pdf.

Petr Kadleček, Alexandru-Eugen Ichim, Tiantian Liu, Jaroslav Křivánek, and Ladislav Kavan. 2016. Reconstructing personalized anatomical models for physics-based body animation. *ACM TOG (Proc. SIGGRAPH Asia)* (2016).

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. ACM, 39–46.

Tassilo Kugelstadt and Elmar Schömer. 2016. Position and orientation based Cosserat rods.. In *Proc. of the Symposium on Computer Animation (SCA)*.

Tsuneya Kurihara and Natsuki Miyata. 2004. Modeling Deformable Human Hands from Medical Images. In *Proceedings of the 2004 ACM SIGGRAPH Symposium on Computer Animation (SCA-04)*. 357–366.

Binh Huy Le and Jessica K. Hodgins. 2016. Real-time Skeletal Skinning with Optimized Centers of Rotation. *ACM Trans. Graph.* 35, 4, Article 37 (2016), 37:1–37:10 pages.

Dongwoon Lee, Michael Glueck, Azam Khan, Eugene Fiume, and Ken Jackson. 2010. A survey of modeling and simulation of skeletal muscle. *ACM Trans. on Graphics* (2010).

J.P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. 2014. STAR: Practice and Theory of Blendshape Facial Models. In *Eurographics*.

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose Space Deformation: A Unified Approach to Shape Interpolation a nd Skeleton-Driven Deformation. In *Proc. ACM SIGGRAPH*.

Duo Li, Shinjiro Sueda, Debanga R Neog, and Dinesh K Pai. 2013. Thin Skin Elastodynamics. *ACM TOG (Proc. SIGGRAPH)* (2013).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM Trans. on Graphics* (2015).

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *Proc. of the International Conference on Motion in Games*.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified particle physics for real-time applications. *ACM TOG (Proc. SIGGRAPH)* (2014).

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based Elastic Materials. *ACM Trans. on Graphics* (2011).

Matthias Muller. 2008. NVIDIA PhysX SDK 3.4.0 Documentation. https://docs.nvidia.com/gameworks/#gameworkslibrary/physx/physx.htm. (Accessed on Aug. 9th, 2018).

Matthias Müller, Jan Bender, Nuttapong Chentanez, and Miles Macklin. 2016. A Robust Method to Extract the Rotational Part of Deformations. In *Proceedings of the 9th International Conference on Motion in Games (MIG '16)*.

Matthias Müller and Nuttapong Chentanez. 2011. Adding Physics to Animated Characters with Oriented Particles.

Matthias Müller and Nuttapong Chentanez. 2011. Solid simulation with oriented particles. *ACM Trans. on Graphics* (2011).

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* (2007).

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. 2015. Dyna: A Model of Dynamic Human Shape in Motion. *ACM TOG (Proc. SIGGRAPH)* (2015).

Marco Romeo, Carlos Monteagudo, and Daniel SÃąnchez-QuirÃşs. 2018. Muscle Simulation with Extended Position Based Dynamics. In *Spanish Computer Graphics Conference (CEIG)*.

Jun Saito and Simon Yuen. 2017. Efficient and Robust Skin Slide Simulation. In *Proceedings of the ACM SIGGRAPH Digital Production Symposium (DigiPro '17)*. ACM, New York, NY, USA, Article 10, 6 pages.

Shunsuke Saito, Zi-Ye Zhou, and Ladislav Kavan. 2015. Computational bodybuilding: Anatomically-based modeling of human bodies. *ACM TOG (Proc. SIGGRAPH)* (2015).

Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. 1997. Anatomy-Based Modeling of the Human Musculature *(SIGGRAPH '97)*. ACM, New York, NY, USA, 163–172.

Christian Schumacher, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner, and Markus Gross. 2012. Efficient simulation of example-based materials. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*.

Eftychios Sifakis, Igor Neverov, and Ronald Fedkiw. 2005. Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data. In *ACM TOG (Proc. SIGGRAPH)*.

J. Spillmann and M. Teschner. 2007. CORDE: Cosserat Rod Elements for the Dynamic Simulation of One-Dimensional Elastic Objects. In *Proc. of the Symposium on Computer Animation (SCA)*.

Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-Art Report. *Proc. Eurographics (State of the Art Reports)* (2016).

Demetri Terzopoulos and Keith Waters. 1990. Physically-based Facial Modeling, Analysis, and Animation. *Journal of Visualization and Computer Animation* 1, 4 (1990), 73–80.

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: shape approximation using spherical quadric error metrics. *ACM Trans. on Graphics* (2013).

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-Meshes for Real-Time Hand Modeling and Tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)* (2016).

Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. 2017. Online Generative Model Personalization for Hand Tracking. *ACM TOG (Proc. SIGGRAPH Asia)* (2017).

Nobuyuki Umetani, Ryan Schmidt, and Jos Stam. 2014. Position-based elastic rods. In *Proc. of the Symposium on Computer Animation (SCA)*.

Shinji Umeyama. 1991. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* (1991).

Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit Skinning: Real-time Skin Deformation with Contact Modeling. *ACM TOG (Proc. SIGGRAPH)* (2013).
Vital Mechanics 2018. http://www.vital.com.
Hongyi Xu and Jernej Barbič. 2016. Pose-space Subspace Dynamics. *ACM Trans. on Graphics* (2016).
Simon Yuen. 2018. *Personal communication.* Head of Creatures, Method Studios.
Lifeng Zhu, Xiaoyan Hu, and Ladislav Kavan. 2015. Adaptable anatomical models for realistic bone motion reconstruction. *Computer Graphics Forum (Proc. EuroGraphics)* (2015).
Ziva Dynamics 2018. Ziva Dynamics. https://ziva.com.

A ELASTIC POTENTIALS

Strain. As defined in Section 5 we can write the strain energy as

$$\mathrm{E}_{\text{strain}} = \iint_D \|\mathbf{R}^T\nabla\mathbf{p} - \bar{\mathbf{R}}^T\nabla\bar{\mathbf{p}}\|^2_{\mathbf{K}}dxdy, \tag{41}$$

where we drop the subscript from $\mathbf{K}$ for brevity. This equation can be extended as

$$\iint_D \|\mathbf{R}^T\nabla(\mathbf{c}+s\mathbf{R}\mathbf{q}) - \bar{\mathbf{R}}^T\nabla(\bar{\mathbf{c}}+\bar{s}\bar{\mathbf{R}}\bar{\mathbf{q}})\|^2_{\mathbf{K}}dxdy, \tag{42}$$

where

$$\mathbf{R}^T\nabla(\mathbf{c}+s\mathbf{R}\mathbf{q}) = \mathbf{R}^T(\nabla\mathbf{c} + \nabla s\mathbf{R}\mathbf{q} + s\nabla\mathbf{R}\mathbf{q} + s\mathbf{R}\nabla\mathbf{q}), \tag{43}$$

$$= \mathbf{R}^T\nabla\mathbf{c} + \nabla s\mathbf{q} + s\mathbf{R}^T\nabla\mathbf{R}\mathbf{q} + s\nabla\mathbf{q}. \tag{44}$$

We can derive the gradient operator for each part of this summation leading to

$$\mathbf{R}^T\nabla\mathbf{c} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{R}^T\nabla_z\mathbf{c}\end{bmatrix} = \begin{bmatrix}0&0&0\\0&0&0\\0&0&\mathbf{w}^T\nabla_z\mathbf{c}\end{bmatrix}, \tag{45}$$

$$\nabla s\mathbf{q} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \nabla_z s\mathbf{q}\end{bmatrix} = \begin{bmatrix}0&0&\nabla_z sx\\0&0&\nabla_z sy\\0&0&0\end{bmatrix}, \tag{46}$$

$$s\mathbf{R}^T\nabla\mathbf{R}\mathbf{q} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & s\boldsymbol{\Omega}\times\mathbf{q}\end{bmatrix} = \begin{bmatrix}0&0&-s\Omega^w y\\0&0&s\Omega^w x\\0&0&s\Omega^u y - s\Omega^v x\end{bmatrix}, \tag{47}$$

$$s\nabla\mathbf{q} = \begin{bmatrix}s\mathbf{e}^x & s\mathbf{e}^y & \mathbf{0}\end{bmatrix} = \begin{bmatrix}s&0&0\\0&s&0\\0&0&0\end{bmatrix}, \tag{48}$$

where $\boldsymbol{\Omega} = [\Omega^u\Omega^v\Omega^w]^T$ is the Darboux vector; and analogous expressions can be derived for $\nabla\bar{\mathbf{p}}$. We can now observe than (41) can be broken up into the sum of distinct energies

$$\mathrm{E}_{\text{strain}} = \iint_D k^z(\mathbf{w}^T\nabla_z\mathbf{c} - \bar{\mathbf{w}}^T\nabla_z\bar{\mathbf{c}})^2 \tag{49}$$

$$+\|(\nabla_z s - \nabla_z\bar{s})\mathbf{q}\|^2_{\mathbf{K}} \tag{50}$$

$$+\|(s\boldsymbol{\Omega} - \bar{s}\bar{\boldsymbol{\Omega}})\times\mathbf{q}\|^2_{\mathbf{K}} \tag{51}$$

$$+(k^x+k^y)(s-\bar{s})^2dxdy, \tag{52}$$

as the cross terms evaluate to $\mathbf{0}$. After integrating over the disc we can reformulate (49) as

$$\mathrm{E}_{\text{strain}} = \pi r^2k^z\|\nabla_z\mathbf{c} - \mathbf{w}\bar{\mathbf{w}}^T\nabla_z\bar{\mathbf{c}}\|^2_2 \tag{53}$$

$$+\frac{\pi r^4(k^x+k^y)}{4}(\nabla_z s - \nabla_z\bar{s})^2 \tag{54}$$

$$+\|s\boldsymbol{\Omega} - \bar{s}\bar{\boldsymbol{\Omega}}\|^2_{\mathbf{H}} \tag{55}$$

$$+\pi r^2(k^x+k^y)(s-\bar{s})^2, \tag{56}$$

where $\mathbf{H} = \begin{bmatrix}\frac{\pi r^4k^z\mathbf{e}^x}{4} & \frac{\pi r^4k^z\mathbf{e}^y}{4} & \frac{\pi r^4(k^x+k^y)\mathbf{e}^z}{4}\end{bmatrix}$ is the second moment of the area of a disc scaled by the stiffness.

Volume. As defined in Section 5 we can write the volume energy as

$$\mathrm{E}_{\text{vol}} = \iint_D k(|\nabla\mathbf{p}| - |\nabla\bar{\mathbf{p}}|)^2dxdy \tag{57}$$

$$= \iint_D k(|\mathbf{R}^T\nabla\mathbf{p}| - |\bar{\mathbf{R}}^T\nabla\bar{\mathbf{p}}|)^2dxdy, \tag{58}$$

Based on the strain derivation, the determinant can be computed as

$$|\mathbf{R}^T\nabla(\mathbf{c}+s\mathbf{R}\mathbf{q})| = |\begin{bmatrix}s&0&\nabla_z sx - s\Omega^w y\\0&s&\nabla_z sy + s\Omega^w x\\0&0&\mathbf{w}^T\nabla_z\mathbf{c} + s\Omega^u y - s\Omega^v x\end{bmatrix}| \tag{59}$$

$$= s^2\mathbf{w}^T\nabla_z\mathbf{c} + s^3(\Omega^u y - \Omega^v x). \tag{60}$$

We can now integrate over the disc leading to

$$\mathrm{E}_{\text{vol}} = \pi r^2k\|s^2\nabla_z\mathbf{c} - s^2_{\bar{\mathbf{p}}}\mathbf{w}\bar{\mathbf{w}}^T\nabla_z\bar{\mathbf{c}}\|^2_2 \tag{61}$$

$$+\frac{\pi r^4k}{2}(s^3\Omega^u - \bar{s}^3\bar{\Omega}^u)^2 \tag{62}$$

$$+\frac{\pi r^4k}{2}(s^3\Omega^v - \bar{s}^3\bar{\Omega}^v)^2. \tag{63}$$

B PREDICTION STEP

As described in (9), the inertial potential over the disc is of the form

$$\mathrm{E}_{\text{inertia}} = \iint_D \frac{m}{2h^2}\|\mathbf{p}_t - \hat{\mathbf{p}}_t\|^2_2dxdy. \tag{64}$$

We aim at finding the unknowns $\mathbf{x}_t = [\,\mathbf{c}_t^T\ s_t\ \boldsymbol{\theta}_t^T\,]^T$ that minimize (64) giving us the prediction update. We denote by $\boldsymbol{\theta}$ the angles parametrizing the rotation matrix. Because the deformation function $\mathbf{p}_t$ is non linear, i.e., due to the rotational degrees of freedom, we rely on a Gauss-Newton iterative scheme for the minimization. We linearize (64) at $\mathbf{x}_t^k$ leading to

$$\arg\min_{\Delta\mathbf{x}_t^k}\iint_D \frac{m}{2h^2}\|\mathbf{A}\Delta\mathbf{x}_t^k - \mathbf{b}\|^2_2dxdy, \tag{65}$$

where k is the iteration number, and $\Delta\mathbf{x}_t = [\,\Delta\mathbf{c}_t^T\ \Delta s_t\ \Delta\boldsymbol{\theta}_t^T\,]^T$. At each iteration we minimize (65), and then apply the update $\mathbf{x}_t^{k+1} = \mathbf{x}_t^k + \Delta\mathbf{x}_t^k$, where we initialize $\mathbf{x}_t^0 = \mathbf{x}_{t-1}$. The matrix $\mathbf{A}$ can be written as $\mathbf{A} = [\,\mathrm{I}_{3\times3}\ \mathbf{R}_t^k\mathbf{q}\ -s_t^k[\mathbf{R}_t^k\mathbf{q}]_\times\,]$, where $[\cdot]_\times$ is a cross product skew-symmetric matrix. The vector $\mathbf{b}$ is defined as $\mathbf{b} = \mathbf{p}_{t-1} + h\dot{\mathbf{p}}_{t-1} + \frac{h^2}{m}\mathbf{f}_{\text{ext}} - \mathbf{p}_t^k$. To compute the prediction updates we will use a single iteration of Gauss-Newton so $\mathbf{b} = h\dot{\mathbf{p}}_{t-1} + \frac{h^2}{m}\mathbf{f}_{\text{ext}}$ as $\mathbf{p}_t^0 = \mathbf{p}_{t-1}$. We will now drop the superscripts and the subscripts to improve readability. As Equation 65 is quadratic we can be find its minimum by solving the normal equation

$$\left(\iint_D \frac{m}{h^2}\mathbf{A}^T\mathbf{A}dxdy\right)\Delta\mathbf{x} = \iint_D \frac{m}{2h^2}\mathbf{A}^T\mathbf{b}dxdy. \tag{66}$$

Interestingly, the left hand side $\iint_D \frac{m}{2h^2}\mathbf{A}^T\mathbf{A}dxdy$ can be simplified to a block diagonal matrix of the form

$$\begin{bmatrix}\iint_D \frac{m}{2h^2}\mathrm{I}_{3\times3}dxdy & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \iint_D \frac{m}{2h^2}(\mathbf{R}\mathbf{q})^T(\mathbf{R}\mathbf{q})dxdy & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \iint_D \frac{ms^2}{2h^2}[\mathbf{R}\mathbf{q}]_\times^T[\mathbf{R}\mathbf{q}]_\times dxdy\end{bmatrix}, \tag{67}$$

by noticing that $[\mathbf{R}\mathbf{q}]_\times^T(\mathbf{R}\mathbf{q}) = \mathbf{0}$. Moreover, as the center of mass of the disc is placed at the origin $\iint_D m\mathbf{R}\mathbf{q}dxdy = \mathbf{0}$ and $\iint_D m[\mathbf{R}\mathbf{q}]_\times = \mathbf{0}$.

We can now integrate the diagonal elements leading to

$$\begin{bmatrix} \frac{\pi r^2 m}{h^2}\mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\pi r^4 m}{2h^2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{ms^2}{h^2}\mathcal{I} \end{bmatrix}, \tag{68}$$

where $\mathcal{I} = \mathbf{R}\begin{bmatrix} \frac{\pi r^4}{4}\mathbf{e}^x & \frac{\pi r^4}{4}\mathbf{e}^y & \frac{\pi r^4}{2}\mathbf{e}^z \end{bmatrix}\mathbf{R}^T$ is the second moment of area of a disc in world-space. The right hand side $\iint_D \frac{m}{h^2}\mathbf{A}^T\mathbf{b}dxdy$ can be simplified as

$$\begin{bmatrix} \frac{\pi r^2 m}{h}\dot{\mathbf{c}} + \boldsymbol{\xi}_{\text{ext}} \\ \frac{\pi r^4 m}{2h}\dot{s} + \boldsymbol{\tau}_{\text{ext}} \\ \frac{ms^2\mathcal{I}}{h}\dot{\boldsymbol{\theta}} + \boldsymbol{\gamma}_{\text{ext}} \end{bmatrix}, \tag{69}$$

where $\boldsymbol{\xi}_{\text{ext}} = \iint_D \mathbf{f}_{\text{ext}}dxdy$ is the sum of the external forces which act on the disc, $\boldsymbol{\tau}_{\text{ext}} = s\iint_D (\mathbf{Rq})\times\mathbf{f}_{\text{ext}}dxdy$ is the sum of the external torques and $\boldsymbol{\gamma}_{\text{ext}} = \iint_D (\mathbf{Rq})\cdot\mathbf{f}_{\text{ext}}dxdy$ is a quantity which can be seen as the counterpart of the external torque by measuring the external forces applied along the center direction.

Center update. By solving the linear system (66) for $\Delta\mathbf{c}$ we find the prediction update for the center

$$\Delta\mathbf{c} = h\dot{\mathbf{c}} + \frac{h^2}{\pi r^2 m}\boldsymbol{\xi}_{\text{ext}}, \tag{70}$$

As we integrate on a disc located at a midpoint this update is valid for the center of the disc located at this point. We approximate the update over the end points by using the same update rule.

Scale update. Similarly, the scale update can be computed solving the linear system for Δs leading to

$$\Delta\mathbf{s} = h\dot{s} + \frac{2h^2}{\pi r^4 m}\boldsymbol{\gamma}_{\text{ext}}, \tag{71}$$

We also approximate the update over the end points using the same update rule

Frame update. The frame update can be computed by solving the linear system for $\Delta\boldsymbol{\theta}$ leading to

$$\Delta\boldsymbol{\theta} = h\dot{\boldsymbol{\theta}} + \frac{\mathcal{I}^{-1}h^2}{s^2 m}\boldsymbol{\tau}_{\text{ext}}. \tag{72}$$

C CORRECTION STEP

Inertia approximation. From the derivation in Appendix B we can obtain an approximation of the inertia term as

$$\mathrm{E}_{\text{inertia}} \approx \frac{\pi r^2 m}{2h^2}\|\mathbf{c}_t - \hat{\mathbf{c}}_t\|_2^2 + \frac{\pi r^4 m}{4h^2}(s_t - \hat{s}_t)^2 + \frac{s^2 m}{2h^2}\|\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_t\|_{\mathcal{I}}^2, \tag{73}$$

where

$$\hat{\mathbf{c}}_t = \mathbf{c}_{t-1} + h\dot{\mathbf{c}}_{t-1} + \frac{h^2}{\pi r^2 m}\boldsymbol{\xi}_{\text{ext}}, \tag{74}$$

$$\hat{s}_t = s_{t-1} + h\dot{s}_{t-1} + \frac{2h^2}{\pi r^4 m}\boldsymbol{\gamma}_{\text{ext}}, \tag{75}$$

$$\hat{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_{t-1} + h\dot{\boldsymbol{\theta}}_{t-1} + \frac{\mathcal{I}^{-1}h^2}{s^2 m}\boldsymbol{\tau}_{\text{ext}}, \tag{76}$$

are the inertial predictions for the different degrees of freedom. The variational form of implicit Euler (9) can then be written in the form

$$\min_{\mathbf{X}} \tfrac{1}{2h^2}\|\mathbf{X} - \hat{\mathbf{X}}\|_{\mathbf{A}}^2 + \tfrac{1}{2}\|\mathbf{W}(\mathbf{X})\|_{\mathbf{K}}^2, \tag{77}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{c}_{[0]}^T, s_{[0]}, \boldsymbol{\theta}_{[.5]}^T, \mathbf{c}_{[1]}^T, s_{[1]}, \boldsymbol{\theta}_{[1.5]}^T, \cdots \end{bmatrix}^T$ and $\hat{\mathbf{X}}$ are vectors containing all the degree of freedoms and their predictions, $\mathbf{K}$ is a block diagonal matrix stacking the stiffness parameters scaled by the length of the piecewise elements, $\mathbf{A}$ is a block diagonal matrix stacking the inertia weights scaled by the length of the piecewise elements, and $\mathbf{W}(\mathbf{X}) = \begin{bmatrix} \mathbf{W}_1(\mathbf{X}) & \mathbf{W}_2(\mathbf{X}) & \dots \end{bmatrix}^T$ stacks the potential energy functions.

Variational Solver. To solve this optimization we can linearize the elastic potentials and write an iterative Gauss-Newton optimization

$$\min_{\Delta\mathbf{X}} \tfrac{1}{2h^2}\|\mathbf{X}^{k-1} + \Delta\mathbf{X} - \hat{\mathbf{X}}\|_{\mathbf{A}}^2 + \tfrac{1}{2}\|\mathbf{W}(\mathbf{X}^{k-1}) + \nabla\mathbf{W}(\mathbf{X}^{k-1})\Delta\mathbf{X}\|_{\mathbf{K}}^2, \tag{78}$$

where k is the iteration number, $\mathbf{X}^k = \mathbf{X}^{k-1} + \Delta\mathbf{X}$, and we initialize $\mathbf{X}^0 = \hat{\mathbf{X}}$. Since Equation 78 is quadratic in the unknown $\Delta\mathbf{X}$, we can minimize it with a single linear solve

$$\tfrac{\mathbf{A}}{h^2}(\mathbf{X}^{k-1} + \Delta\mathbf{X} - \hat{\mathbf{X}}) + \nabla\mathbf{W}(\mathbf{X}^{k-1})^T\mathbf{K}\left(\mathbf{W}(\mathbf{X}^{k-1}) + \nabla\mathbf{W}(\mathbf{X}^{k-1})\Delta\mathbf{X}\right) = \mathbf{0}.$$

However, the conditioning of this linear system is greatly dependent on how stiff are the elastic potentials. Following the optimization trick presented in [Gould 1986], for elastic potentials with large stiffness a better option is to split the equation above as

$$\begin{cases} \frac{\mathbf{A}}{h^2}(\mathbf{X}^{k-1} + \Delta\mathbf{X} - \hat{\mathbf{X}}) + \nabla\mathbf{W}(\mathbf{X}^{k-1})^T\boldsymbol{\lambda}^k = \mathbf{0}, & (79) \\ \mathbf{K}^{-1}\boldsymbol{\lambda}^k = \mathbf{W}(\mathbf{X}^{k-1}) + \nabla\mathbf{W}(\mathbf{X}^{k-1})\Delta\mathbf{X}. & (80) \end{cases}$$

Note that when the elastic potentials are infinitively stiff $\mathbf{K}^{-1}$ vanishes $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_n]^T$ becomes the vector of Lagrange multipliers. We can now reformulate (79) as

$$\Delta\mathbf{X} = -h^2\mathbf{A}^{-1}\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\boldsymbol{\lambda}^k - (\mathbf{X}^{k-1} - \hat{\mathbf{X}}) \tag{81}$$

$$\approx -h^2\mathbf{A}^{-1}\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\Delta\boldsymbol{\lambda}, \tag{82}$$

by assuming $\nabla\mathbf{W}(\mathbf{X}^k) \approx \nabla\mathbf{W}(\mathbf{X}^{k-1})$, and where $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1} + \Delta\boldsymbol{\lambda}$ and we initialize $\boldsymbol{\lambda}^0 = \mathbf{0}$. This can be proven by induction knowing that $\mathbf{X}^0 = \hat{\mathbf{X}}$ and $\boldsymbol{\lambda}^0 = \mathbf{0}$. By substituting $\Delta\mathbf{X}$ in the Equation 80, we can rewrite the system of equations as

$$\begin{cases} \Delta\mathbf{X} = -h^2\mathbf{A}^{-1}\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\Delta\boldsymbol{\lambda}, \\ (h^2\|\nabla\mathbf{W}(\mathbf{X}^{k-1})^T\|_{\mathbf{A}^{-1}}^2 + \mathbf{K}^{-1})\Delta\boldsymbol{\lambda} = \mathbf{W}(\mathbf{X}^{k-1}) - \mathbf{K}^{-1}\boldsymbol{\lambda}^{k-1}. \end{cases}$$

Therefore, $\Delta\mathbf{X}$ and $\Delta\boldsymbol{\lambda}$ can be found with a single linear solve.

D CLOSED FORM PILL PROJECTION

Given a pill $\mathbf{P} = \{(\mathbf{c}_1, r_1), (\mathbf{c}_2, r_2)\}$, we can compute the closest point distance of a point $\mathbf{x}$ onto $\mathbf{P}$ in close form as

$$d = \|\mathbf{x} - \mathbf{c}_1 + t\hat{\mathbf{j}}l\|_2 - ((1-t)r_1 + tr_2) \tag{83}$$

where $l = \|\mathbf{c}_1 - \mathbf{c}_2\|_2$ is the pill length, $\hat{\mathbf{j}} = l^{-1}(\mathbf{c}_1 - \mathbf{c}_2)$ is the pill versor, $\mathbf{p}_s = \mathbf{c}_1 - \hat{\mathbf{j}}\left((\mathbf{x} - \mathbf{c}_1)\cdot\hat{\mathbf{j}}\right)$ the orthogonal projection onto the pill segment, $\theta = \arcsin(l^{-1}(r_1 - r_2))$ is the pill slope angle, and $t = \min(\max(-l^{-1}(\mathbf{p}_s + o\hat{\mathbf{j}} - \mathbf{c}_1)\cdot\hat{\mathbf{j}}, 0), 1)$ is the barycentric coordinate of the projection, where $o = \|\mathbf{x} - \mathbf{p}_s\|_2^2\tan(\theta)$.