Actuators À La Mode: Modal Actuations for Soft Body Locomotion

OTMAN BENCHEKROUN, University of Toronto, Canada and Roblox Research, USA
KAIXIANG XIE, McGill University, Canada and Roblox Research, Canada
HSUEH-TI DEREK LIU, Roblox Research, Canada
EITAN GRINSPUN, University of Toronto, Canada
SHELDON ANDREWS, École de Technologie Supérieure, Canada and Roblox Research, USA
VICTOR ZORDAN, Roblox Research, USA

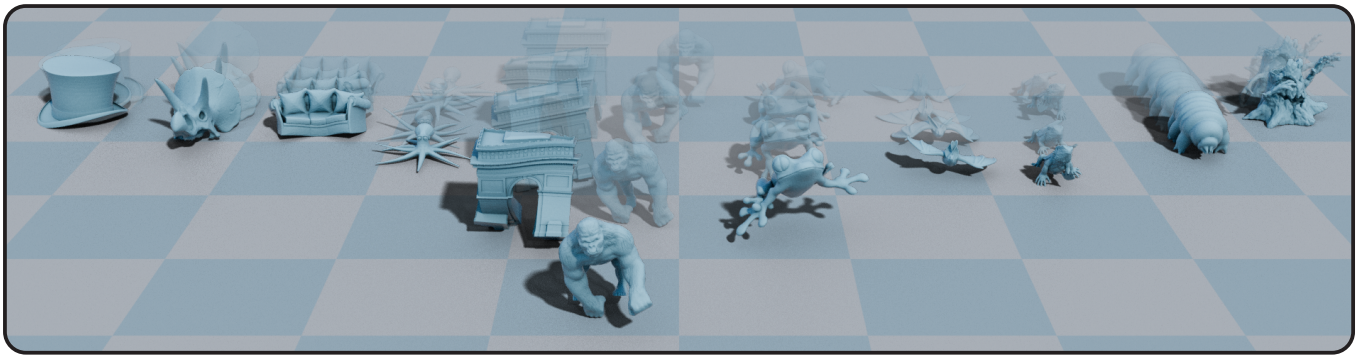


Fig. 1. We can generate locomotions for a wide variety of complex, high resolution, deformable character geometries by combining a spatio-temporal actuation subspace with a reduced order simulation.

Traditional character animation specializes in characters with a rigidly articulated skeleton and a bipedal/quadripedal morphology. This assumption simplifies the design of physically based animations, like locomotion, but comes with the price of excluding characters of arbitrary deformable geometries.

To remedy this, we propose a spatio-temporal actuation subspace built off the natural vibrations of the character geometry. We show this actuation subspace is well suited for designing natural locomotion, without requiring user-provided guidance keyframes as is common in prior work. The resulting actuation is coupled to a reduced fast soft body simulation, allowing us to optimize for locomotions for a wide variety of high resolution deformable characters.

1 INTRODUCTION

The world is comprised of a rich diversity of deformable organisms. These organisms leverage their unique geometry and deformable nature for complex interactions with their environment in order to locomote, manipulate objects, or take flight. They also serve as inspiration for the many fantastical creatures and virtual characters that populate video games, films, and virtual environments. Having simulation and control methods that are able to faithfully capture the essence of these organisms and their behaviors is key to filling virtual worlds with these creative characters. However, most work on developing controllers for virtual characters is predicated on an extremely narrow set of morphologies that assumes an underlying rigid skeletal structure, and further imposes that they be bipedal or quadrupedal.

These assumptions preclude a wide range of characters, which are neither bipedal or quadrupedal, from being animated. For instance, the motion of an octopus with tentacles as shown in Fig. 2 could not be accurately modeled using rigid segments, an issue that applies to a wide variety of deformable organisms. These limitations motivate the need to develop locomotion controllers for soft-body characters with a wide range of highly detailed deformable mesh geometry, yet without the requirement of a piecewise rigid skeleton.

Unfortunately, such controllers are difficult to generate for three reasons. First, there exists *significantly* less motion data for arbitrary geometries than there is for bipedal or quadripedal characters. This data is key for creating natural-looking motions and its absence results in behaviors that exhibit unrealistic jittery motion [Heess et al. 2017].

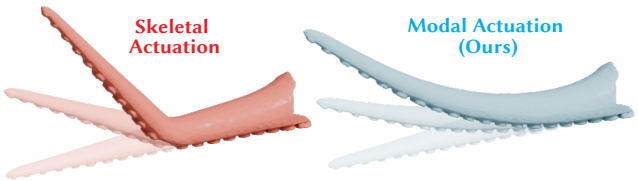


Fig. 2. Traditional animation techniques an underlying skeleton structure, which is actuated via joint torques. The resulting actuation exhibits kinky, piecewise rigid deformation. Our actuation properly models the octopus tentacle as a deformable body, allowing for a naturally smooth deformation.

Authors' addresses: Otman Benchekroun, otman.benchekroun@mail.utoronto.edu, University of Toronto, Canada and Roblox Research, USA; Kaixiang Xie, , McGill University, Canada and Roblox Research, Canada; Hsueh-Ti Derek Liu, , Roblox Research, Canada; Eitan Grinspun, eitan@cs.toronto.edu, University of Toronto, Canada; Sheldon Andrews, , École de Technologie Supérieure, Canada and Roblox Research, USA; Victor Zordan, , Roblox Research, USA.

Second, deformable characters make use of a complex organization of various organic tissues, all interacting in harmony to carry out even the simplest of tasks [Keller et al. 2023]. Designing this musculo-skelature for many characters becomes a tedious, unintuitive ordeal.

Finally, high resolution deformable characters usually come with the price of an expensive simulation, which scales in complexity with the resolution of the character geometry.

We propose a very simple, scalable pipeline for generating locomotion controllers for deformable characters of arbitrary user-defined geometry. Our method defines a small data-free spatio-temporal actuation subspace based off the geometry of the character alone. An actuation signal in this subspace corresponds to a shape that the deformable character naturally wants to take on. We then define a plasticity-based actuation energy, that guides our simulated character towards this actuated target shape.

Unlike the linear actuation used in prior work, our actuation is always rotation and translation invariant, ensuring our character is not allowed non-physical control over its own net torque and force.

We then show how to couple this plastic actuation to a *reduced* deformable simulation, allowing us to perform the entire soft-body controller optimization within the reduced spaces of actuation and simulation, decoupled from the mesh resolution. This allows us to achieve a variety of crawling, running and hopping-like locomotion behaviors with an off the shelf CMAES optimizer, without requiring user-assisted guiding motions or keyframes.

We evaluate the effectiveness of our actuation subspace on a wide variety of character geometries, obtaining varied locomotions for highly detailed characters within minutes, while also providing avenues for intuitive motion control and design.

2 RELATED WORK

2.1 Soft Body Controllers

Soft body characters bring with them many technical challenges in the design of locomotion controllers. In particular, the physical model used for simulation of soft bodies is often characterized by a large number of degrees of freedom, bottle-necking the controller optimization.

A limited amount of previous work addresses constructing soft body controllers, with the significant majority modeling the full space of the character geometry. Due to the increased complexity from using high resolution models, most only animate simple 2D or coarse 3D characters [Coros et al. 2012; Jain and Liu 2011; Tan et al. 2012]. Even with simpler physical models and character geometries, Bhatia et al. [2021]; Hu et al. [2019]; Huang et al. [2024]; Lin et al. [2020]; Min et al. [2019]; Rojas et al. [2021] all report that training controllers for soft-body creatures with fewer than several thousand degrees of freedom can require hours, or even days, of training time, with simulations running much slower than real-time.

The large number of degrees of freedom (DOFs) required of soft-body simulations has motivated the development of reduced order models(ROMs) in order to improve simulation performance [Benchekroun et al. 2023; Brandt et al. 2018; Trusty et al. 2023], with most of the prior work focusing on accelerating *passive* simulation.

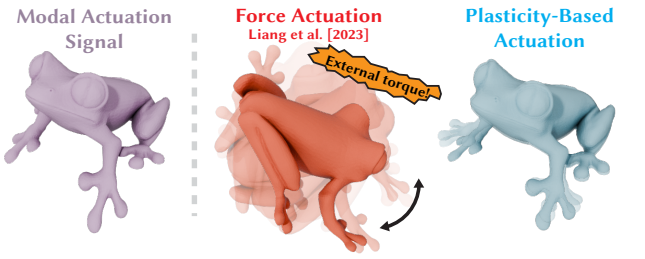


Fig. 3. Our plasticity based actuation energy conserves angular momentum, while taking on the expected target shape. The force-based actuation from Liang et al. [2023] does not conserve angular momentum, creating super-natural rotational motion upon actuation .

ROMs have been leveraged to accelerate high resolution trajectory optimization tasks [Pan and Manocha 2018], but their use remains limited for more general controllers. While Liang et al. [2023] accelerate a closed loop Model Predictive Control (MPC) controller, their approach still remains limited to simple and coarse character geometries, while exhibiting artifacts upon actuation Fig. 3.

Modal spaces for actuation have been leveraged for control problems guided by user-specified keyframes and objectives [Barbič et al. 2009; Li et al. 2013], pushing the optimization towards a user-designed solution. Our insight shows these modal spaces, when sinusoidally activated and coupled to a plasticity-based actuator, provide a basis expressive enough to discover natural locomotion in a physical environment, even without the need for user provided data.

2.2 Designing Character Musculature

Beyond fast simulation, an important step in designing controllers for a character lies in defining a viable musculoskelature for activating the physical system, and for arbitrary soft body characters, this is far from trivial. By far the most common method of constructing this structure for rigid articulated characters is through the use of joint torques applied at the intersection of adjacent bones [Xu et al. 2023; Yin et al. 2007]. These joint torques are either actuated directly, or indirectly through desired joint angles via PD controllers [Reda et al. 2020; Tan et al. 2011b]. Similar joint torques have been used to model soft-body characters whose motion is *skeleton* driven [Kim and Pollard 2011a,b; Liu et al. 2013], but this does not generalize to arbitrary soft body characters where a skeleton is either not obvious or non-existent.

Instead of joint torques, Geijtenbeek et al. [2013] use biomechanically inspired muscles, resembling springs with adjustable rest lengths, which ease the creation of contractible spring-like structures in deformable bodies [Rojas et al. 2021].

Of course, the configuration and placement of muscles play an important contribution to the type of control that can be achieved. Keller et al. [2023]; Lee et al. [2019, 2009, 2014]; Saito et al. [2015] all propose highly detailed musculoskeletal models based on the real-world human anatomy. However, as we explore different character morphologies, realistic anatomical models become ill-defined, and many methods rely on a user to manually design the placement of muscle fibers. In the absence of an anatomical model, designers are

left to their own devices, often opting to draw muscles directly onto the character [Min et al. 2019; Tan et al. 2012].

Alternatively, others shift the burden to an optimization procedure to design the control actuation structure [Bhatia et al. 2021; Liang et al. 2023; Lin et al. 2020; Ma et al. 2021]. The resulting optimization has a large search space, and has again been limited to coarse character geometries.

Yet another set of approaches from Coros et al. [2012]; Ichim et al. [2017]; Pan and Manocha [2018] propose a musculature defined by a plasticity-like deformation. The character is encouraged to take on a changing target shape, as prescribed by a coarse cage, face scan data, or user-prescribed constraints and objectives. Our approach similarly encourages the shape towards a new rest shape, but in our case, this rest shape is described by the natural vibrations of the character geometry.

2.3 Motion Priors

State-of-the art character controllers often crucially make use of motion priors in order to guide the character to more natural behaviors. This is done by specifying the expected characteristics of the desired motion.

The copious amounts of motion data available for articulated characters lends itself well to this task, with many modern controllers making use of this data as reference motion that characters are encouraged to imitate [Peng et al. 2018, 2021; Xu and Karamouzas 2021]. Aside from imitation, another effective method of leveraging this motion data is by constructing a motion subspace (or motion manifold [Starke et al. 2022]), either with PCA [Chai and Hodgins 2005] or more modern Deep Learning-based autoencoders [Bergamin et al. 2019; Holden et al. 2020, 2015].

Unfortunately, for soft body control of arbitrary geometries, motion data is not readily available. Thus, we must turn to alternative principles for guiding controllers, without the use of motion data.

For example, Ranganath et al. [2021] derive an actuation subspace via a Principal Component Analysis (PCA) applied to their own synthetically generated data. They show their deep reinforcement learning framework can animate rigidly articulated character with widely varying morphologies.

Heess et al. [2017] instead turn to a more rigorous training routine to fill the hole, observing that training a controller on a diverse set of environments leads to more robust rigid body controllers that better satisfy user defined objectives, albeit with some undesired jittery motion.

Alternatively, periodicity has shown itself to be a particularly powerful prior for locomotion, with Yu et al. [2018] generating locomotion controllers that reward periodic and symmetric behaviors for articulated-bodies. This can be taken one step further by imbuing periodicity into our controller actuation space itself; Central Pattern Generators (CPG) [Guertin 2009] model character motions with simple periodic control signals that are propagated across the character according to a low-dimensional set of parameters. These kinds of temporal motion subspaces have been effective for generating swimming and crawling motions [Ma et al. 2021; Min et al. 2019; Tan et al. 2011a; Tu and Terzopoulos 1994] and have been injected into modern deep reinforcement learning architectures to create rich encodings of human motions [Holden et al. 2017; Starke et al. 2022]

On top of periodicity, others also turn to energy efficiency to guide their animations. For example, Kry et al. [2009] and Nunes et al. [2012] show that the natural vibrations of articulated characters can be used to generate kinematic locomotion animations for rigid characters of varying morphology.

Our approach extends these energy efficient, periodic subspaces demonstrated in prior works, and shows how these same principles can be used to find natural soft body locomotion, without user guidance.

3 METHOD OVERVIEW

Our goal is to derive an actuation for deformable characters that generates motion satisfying certain objectives, while remaining physically plausible. We cast this as a controller optimization problem,

$$\min_{\boldsymbol{d}(t)} \quad J(\boldsymbol{x}(t)), \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{x}(t) = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ E(\boldsymbol{x}, \boldsymbol{d}(t)) \quad \forall t \in [t_0, t_1], \tag{2}$$

where we are solving for a time-varying actuation $\boldsymbol{d}(t) \in \mathbb{R}^{3n}$ that corresponds to the target shape our character should take. $J(\boldsymbol{x}(t))$ is a task-specific objective on the vertex positions $\boldsymbol{x}(t) \in \mathbb{R}^{3n}$ that rewards desired motions (e.g for a locomotion task, it can reward motion of the center of mass along a target direction).

The constraint Eq. (2) ensures the resulting motion obeys the laws of soft-body physics with contact; The vertex positions at any point in time $\boldsymbol{x}(t) \in \mathbb{R}^{3n}$ must be the minimizers of the total energy $E(\boldsymbol{x}(t), \boldsymbol{d}(t))$ of the physical system. Temporarily dropping the dependence on time for clarity, this per-timestep physical energy can be split into two components,

$$E(\boldsymbol{x}, \boldsymbol{d}) = E_p(\boldsymbol{x}) + E_a(\boldsymbol{x}, \boldsymbol{d}), \tag{3}$$

a passive component $E_p(\boldsymbol{x})$ that captures motion according to inertia, external forces, and elastic forces, and an active component $E_a(\boldsymbol{x}, \boldsymbol{d})$ that encourages the elastic body to take on the shape described by $\boldsymbol{d}$.

On top of being non-linear in both $\boldsymbol{x}(t)$ and $\boldsymbol{d}(t)$, Eq. (1) becomes especially difficult to solve for high resolution characters because the problem scales in complexity with the increased dimensionality of $\boldsymbol{x}(t)$ and $\boldsymbol{d}(t)$. The sour combination of high non-linearity and high dimensionality makes solving this problem an extremely slow process prone to large null spaces.

Our solution to this problem is to construct an actuation subspace for $\boldsymbol{d}(t)$ based on the natural elastic vibration modes of the character [Pentland and Williams 1989] and then then define an actuation in this space via a plasticity-based actuation energy [Ichim et al. 2017] .

We combine this actuation reduction with a separate simulation subspace for $\boldsymbol{x}(t)$, built off Skinning Eigenmodes [Benchekroun et al. 2023]. We can then carry out simulation within these reduced spaces, including within the novel non-linear plastic actuation energy , while remaining independent from the resolution of the mesh. This fully reduced-space framework allows us to quickly evaluate simulations of arbitrarily high resolution characters.

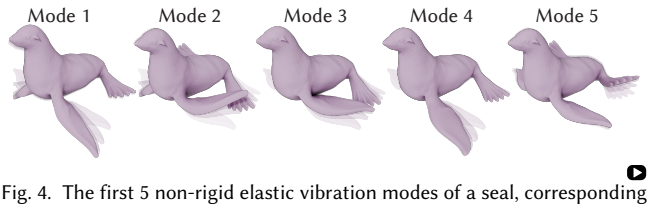


Fig. 4. The first 5 non-rigid elastic vibration modes of a seal, corresponding to reasonable low-energy motions one could expect to see from a seal.

This speedup in simulation evaluation lends itself well to the use of a Covariance Matrix Adaptation Evolution Strategy(CMAES) [Hansen 2006] optimizer in order to solve our control optimization Eq. (1). This derivative-free genetic algorithm iteratively samples populations of $\boldsymbol{d}(t)$, which are used to drive multiple simulations. The resulting objective J is measured for each simulation, and is used by CMAES to inform the collection of a next, more optimal generation of $\boldsymbol{d}(t)$. The process is repeated until convergence, or until a maximum number of iterations are achieved.

3.1 Modal Actuation Subspace

Without access to motion data for arbitrary meshed characters, the guiding principles for defining our spatial-temporal actuation subspace become energy efficiency and periodicity.

We draw inspiration from modal analysis [Pentland and Williams 1989] and build our subspace starting with the vibration modes of the character. These are deformations of the character that induce the *least* elastic energy, describing motions that the character *prefers* to take as visualized on a seal in Fig. 4. While these vibration modes are most commonly used for accelerating passive simulations [Barbič and James 2005; Trusty et al. 2023], we show that with the right actuation energy they form an effective subspace for soft body character actuation.

Vibration modes are the solutions to the generalized eigenvalue problem of the elastic energy Hessian $\boldsymbol{H} \in \mathbb{R}^{3n\times 3n}$,

$$\boldsymbol{H}\boldsymbol{D} = \boldsymbol{M}\boldsymbol{D}\boldsymbol{\Lambda}\,, \tag{4}$$

where the eigenvectors $\boldsymbol{D} \in \mathbb{R}^{3n\times m}$ form an m-dimensional basis of *spatial actuation modes*. Above, $\boldsymbol{M} \in \mathbb{R}^{3n\times 3n}$ is the diagonal vector-mass matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues.

Deformed target positions $\boldsymbol{d} \in \mathbb{R}^{3n}$ in this basis of actuation modes can be compactly represented by a low dimensional time-varying actuation vector $\boldsymbol{a}(t) \in \mathbb{R}^m$ as well as the rest geometry $\boldsymbol{x}_0 \in \mathbb{R}^{3n}$,

$$\boldsymbol{d} = \begin{bmatrix}\boldsymbol{D} & \boldsymbol{x}_0\end{bmatrix}\begin{bmatrix}\boldsymbol{a}(t)\\ 1\end{bmatrix} = \bar{\boldsymbol{D}}\bar{\boldsymbol{a}}(t). \tag{5}$$

Where the bar denotes a homogeneous expression of this actuation space $\bar{\boldsymbol{D}} = [\boldsymbol{D}\ \boldsymbol{x}_0] \in \mathbb{R}^{3n\times(m+1)}$ and $\bar{\boldsymbol{a}} = [\boldsymbol{a}(t)\ 1]^T \in \mathbb{R}^{m+1}$, which we employ to simplify notation on future expressions.

However, interpreting this actuation subspace as a linear space for forces [Barbič et al. 2009; Li et al. 2013; Liang et al. 2023] allows the character for non-physical control over its own net force and net torque Fig. 3. We show in Sec. 3.2.1 how to use the natural vibrations to construct a plasticity based actuation, which guarantees rotation and translation invariance.

With the intuition that most organisms make use of periodic motion patterns for locomotion, we imbue our actuation subspace with periodicity. Our temporal actuation subspace is defined via a sum of k sinusoids,

$$\boldsymbol{a}_i(t) = \sum_j^k \boldsymbol{A}_{ij}\sin\left(2\pi\left(\frac{t}{\boldsymbol{T}_{ij}} + \boldsymbol{\theta}_{ij}\right)\right). \tag{6}$$

where $\boldsymbol{A}_{ij}, \boldsymbol{T}_{ij}, \boldsymbol{\theta}_{ij}$ are the amplitude, period and phase shift for each actuation mode i and sinusoidal function j, which are parameters that are either learned by Eq. (1), or set by a user.

3.2 Reduced Simulation

With a low-dimensional actuation subspace in hand, we can actuate a soft body independently of its mesh resolution. However, this resolution independence does not transfer over to the simulation of the deformable, impeding forward simulation of high resolution characters. Worse yet, optimizing the parameters of the actuation subspace by solving Eq. (1) requires many such forward simulations, becoming intractable as resolution increases.

For this reason, we follow prior work [Barbič et al. 2009; Liang et al. 2023] and introduce a separate reduction of the soft body . We use a linear Skinning Eigenmode subspace [Benchekroun et al. 2023] for our vertex positions,

$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z} \tag{7}$$

where $\boldsymbol{z} \in \mathbb{R}^r$ are the reduced space vertex positions and $\boldsymbol{B} \in \mathbb{R}^{3n\times r}$ being our Skinning Eigenmode subspace matrix. We refer the reader to App. A for a refresher on the construction of this subspace.

With our two subspaces for simulation and actuation in hand, we must now rewrite the simulation optimization problem described in Eq. (2) entirely in our reduced spaces. We restate the full-space optimization problem in question,

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ E_p(\boldsymbol{x}) + E_a(\boldsymbol{x}, \boldsymbol{d})$$

with the goal of finding a method for solving this optimization problem, without ever touching the full resolution of the mesh. Prior work [Barbič and James 2005; Benchekroun et al. 2023; Brandt et al. 2018; Jacobson et al. 2012] discusses extensively how to reduce passive soft body simulation and we refer the reader to App. D for details on the passive simulation, summarizing that we largely follow the method of Benchekroun et al. [2023] and extend it with a simple projective ground-plane damping contact model App. E to arrive at a reduced passive approximation,

$$E_p(\boldsymbol{x}) \approx \tilde{E}_p(\boldsymbol{z}). \tag{8}$$

which can be evaluated entirely in a reduced space of r skinning eigenmodes and a set of $|C_p|$. passive rotation clusters.

3.2.1 Reducing the Actuation Energy $E_a(\boldsymbol{x}, \boldsymbol{d})$. We focus the remainder of our discussion on the actuation energy $E_a(\boldsymbol{x}, \boldsymbol{d})$. This energy aims to make our simulated character $\boldsymbol{x}$ take on a target shape $\boldsymbol{d}$, and has the following form [Ichim et al. 2017]:

$$\begin{aligned} E_a(\boldsymbol{x}, \boldsymbol{d}) &= \sum_e^{|\mathcal{T}|} \min_{\Omega_e} \gamma_e V_e ||\boldsymbol{F}_e(\boldsymbol{x}) - \Omega_e \boldsymbol{Y}_e(\boldsymbol{d})||_F^2 \\ &s.t.\ \Omega_e \in \mathcal{SO}(3). \end{aligned} \tag{9}$$

where $\boldsymbol{F}(\boldsymbol{x})$ corresponds to the deformation Jacobian of the simulated character $\boldsymbol{x}$, while $\boldsymbol{Y}(\boldsymbol{d})$ is the target deformation Jacobian of the actuation shape $\boldsymbol{d}$. γ_e and V_e are the per-element actuation stiffness and volume respectively, the former of which can be tuned to achieve a stronger/weaker character.

The matrix $\boldsymbol{\Omega}_e$ is a locally defined best fit rotation matrix, essential for filtering out rotations from our actuation; an actuation signal $\boldsymbol{d}$ should lead to the same motion regardless of how the simulated element shape $\boldsymbol{F}_e(\boldsymbol{x})$, and the target element shape $\boldsymbol{Y}_e(\boldsymbol{d})$ are oriented.

This also imposes that our actuator conserves angular momentum, ensuring the character cannot supernaturally create an external torque. This is not the case for a simple force-based actuation [Liang et al. 2023] which can introduce spurious linear and angular forces as shown in Fig. 3.

Plugging our subspaces for simulated positions $\boldsymbol{x} = \boldsymbol{Bz}$ and actuation target $\boldsymbol{d} = \boldsymbol{Da}$ into this energy,

$$E_a(\boldsymbol{z}, \boldsymbol{a}) = \frac{1}{2} \sum_{e}^{|\mathcal{T}|} \min_{\Omega_e} \gamma_e V_e ||\boldsymbol{F}_e(\boldsymbol{z}) - \boldsymbol{\Omega}_e \boldsymbol{Y}_e(\boldsymbol{a})||_F^2, \quad s.t.\ \boldsymbol{\Omega}_e \in \mathcal{SO}(3), \tag{10}$$

leads us to a disappointing result; evaluating this energy requires the computation of the per-element best fit rotation matrix Ω_e, requiring computation that scales with the number of tetrahedra in the character mesh.

3.2.2 Clustering $\boldsymbol{\Omega}_e$ for Mesh Independence of $E_a(\boldsymbol{z}, \boldsymbol{a})$. To avoid recomputing Ω_e for each element in the mesh every simulation step, we derive a clustering scheme for our actuation energy, providing a reduction from the full number of elements $|\mathcal{T}|$. Specifically, we allow for multiple elements to share the same rotation matrix $\boldsymbol{\Omega}_{c(e)}$, where $c(e) \in |C_a|$ identifies the actuation cluster c, in the set of actuation clusters $|C_a|$ to which an element e belongs. Our approximation for the actuation energy becomes

$$E_a(\boldsymbol{x}, \boldsymbol{d}) \approx \tilde{E}_a(\boldsymbol{x}, \boldsymbol{a}) = \frac{1}{2} \min_{\Omega_c} \sum_{e}^{|\mathcal{T}|} \gamma_e V_e ||\boldsymbol{F}_e - \boldsymbol{\Omega}_{c(e)} \boldsymbol{Y}_e||_F^2 \quad s.t.\ \boldsymbol{\Omega}_{c(e)} \in \mathcal{SO}(3)\ \forall c \in |C_a|. \tag{11}$$

We prove in App. G that the optimal clustered rotation Ω_c minimizing the objective above can be found via a polar decomposition of the γV-weighed sum of the covariance matrix $\boldsymbol{F}_e \boldsymbol{Y}_e^T$,

$$\boldsymbol{\Omega}_c = \text{polar}\left(\sum_{e}^{|\mathcal{T}(c)|} \gamma_e V_e \boldsymbol{F}_e \boldsymbol{Y}_e^T \right) = \text{polar}\left(\boldsymbol{H}_c\right), \tag{12}$$

where $\text{polar}(\boldsymbol{H}_c)$ is a function performing polar decomposition on $\boldsymbol{H}_c$ in order to achieve its rotational component $\boldsymbol{\Omega}_{c(e)}$.

While evaluating this term once again involves a per-element sum, we notice that this sum is a bilinear form in our two low-dimensional time-varying quantities, $\boldsymbol{z}$ and $\bar{\boldsymbol{a}}$. With this intuition, we show in App. F that by precomputing the appropriate tensor we can efficiently evaluate this sum at run-time in a manner scaling only in complexity with our small $\boldsymbol{z}$ and $\bar{\boldsymbol{a}}$,

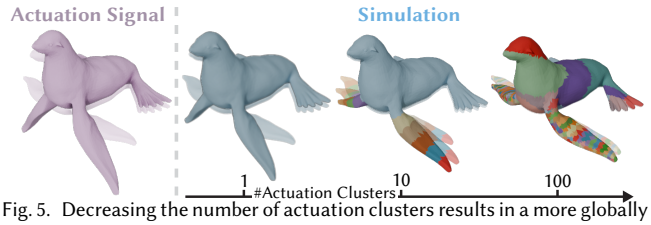


Fig. 5. Decreasing the number of actuation clusters results in a more globally actuated shape, accelerating the speed at which the simulation matches the target actuation signal.

$$\boldsymbol{H}_c = \sum_{u}^{r} \sum_{v}^{m+1} \mathcal{H}_{cuv} z_u \bar{a}_v \tag{13}$$

Above, we have introduced the tensor $\mathcal{H} \in \mathbb{R}^{|C| \times 3 \times 3 \times r \times (m+1)}$, which we derive in App. F, that when multiplied against $\boldsymbol{z}$ and $\bar{\boldsymbol{a}}$, computes our per-cluster covariance matrix $\boldsymbol{H} \in \mathbb{R}^{|C_a| \times 3 \times 3}$. Note that evaluating $\boldsymbol{H}_c$ every stimulation step via this tensor product only requires operations that scale with the number of clusters $|C_a|$, the dimension of our reduced positional DOFs $\boldsymbol{z}$, and the size of our actuation $\bar{\boldsymbol{a}}$.

As shown in Fig. 5, a low number of actuation clusters allows for a more globally defined rotation invariance, with every character tetrahedron sharing the same rest frame. This results in a simulated character that responds quickly to the actuation signal prescribed. Increasing the number of clusters actuates each tetrahedron locally, independently of its neighbors, resulting in a slower response to actuation.

Finally, with our newfound ability to quickly evaluate our rotations, we arrive at the final form of our reduced and clustered simulation optimization problem.

$$\boldsymbol{z} = \underset{\boldsymbol{z}}{\text{argmin}}\ \tilde{E}_p(\boldsymbol{z}) + \tilde{E}_a(\boldsymbol{z}, \boldsymbol{a}) \tag{14}$$

Where we have followed prior work in reducing per-element nonlinearities that appear in the passive term [Jacobson et al. 2012].

3.3 Local Global Solver

We make use of a local global solver to solve our reduced simulation optimization problem. This kind of solver comes with the advantage of maintaining a constant energy Hessian, allowing us to compute a factorization for this matrix once, then reuse it throughout the entire rest of the optimization pipeline. This is comprised of two main stages, a local step and a global step, that are repeated until a convergence criterion is met.

The *local* step first holds $\boldsymbol{z}$ fixed and optimizes for $\boldsymbol{\Omega}_c$ using Eq. (13) and Eq. (12). Then, the *global* step holds $\boldsymbol{\Omega}_t$ fixed, optimizing for $\boldsymbol{z}$. This is resolved using a linear solve we derive in App. D,

$$\boldsymbol{Q}\boldsymbol{z} = \boldsymbol{f}, \tag{15}$$

with $\boldsymbol{f} \in \mathbb{R}^r$ and $\boldsymbol{Q} \in \mathbb{R}^{r \times r}$ given by,

$$\boldsymbol{Q} = \boldsymbol{Q}^p + \boldsymbol{Q}^a \qquad \boldsymbol{f} = \boldsymbol{f}^p + \boldsymbol{f}^a + \boldsymbol{f}^c. \tag{16}$$

The constant system matrix $\boldsymbol{Q}$ is a sum of the passive energy Hessian $\boldsymbol{Q}^p$ and our actuation energy Hessian $\boldsymbol{Q}^a$. Details on the derivation of $\boldsymbol{Q}^p$ and $\boldsymbol{Q}^a$ can be found in App. D.

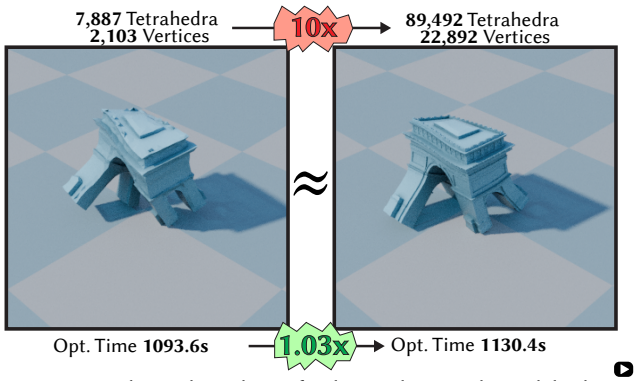


Fig. 6. Varying the mesh resolution for the *Arc de Triomphe* mesh leads to a very similar locomotion style, with negligible extra optimization time.

The right hand side to the system in Eq. (15) $\boldsymbol{f}$ is a sum of the passive forces f^p, actuation forces $f^a(\Omega)$, and contact forces $\boldsymbol{f}^c(\boldsymbol{z})$. These contact forces are found in a projective fashion such that the resulting simulation after the global step Eq. (15) satisfies our no interpenetration, damping friction contact conditions. Collision detection and handling is performed on a subset of the mesh vertices to maintain reduced complexity, as visualized on a tardigrade in the appendix Fig. 13. More details on the computation of this contact force can be found in App. E.

4 RESULTS & DISCUSSION

We showcase the rich variety of locomotions our actuation subspace can generate by specifying the control optimization problem with a very simple objective Eq. (1)

$$J = J_{disp} \cdot J_{align}$$
$$J_{disp} = (\boldsymbol{x}_{COM}(t_0) - \boldsymbol{x}_{COM}(t_1)) \cdot \hat{\boldsymbol{v}} \quad J_{align} = \min_t \hat{\boldsymbol{u}}(t) \cdot \hat{\boldsymbol{v}}, \tag{17}$$

where J_{disp} measures the distance travelled along a target direction $\hat{\boldsymbol{v}}$ throughout the episode, and J_{align} encourages the forward direction of the character, $\hat{\boldsymbol{u}}$ to point in that same target direction $\hat{\boldsymbol{v}}$. We solve the control optimization problem with open-loop controllers using an off-the-shelf implementation of the CMAES[Hansen 2006], using the `pycma` library [Hansen et al. 2019] and aim to directly predict the temporal actuation parameters, $\boldsymbol{A}$, and $\boldsymbol{T}$, $\boldsymbol{\theta}$. We can easily pick a reasonable sampling range for CMAES to explore these parameters as described in App. H .

Controller Optimization Speed. Our controller optimization is carried out entirely in a reduced space. This allows us to obtain locomotions for characters of arbitrarily high resolutions, with the slowest mesh we collect statistics on in Table. 1, the octopus, taking just under an hour to locomote. In contrast, using simulation statistics from a state of the art full space simulator of Trusty et al. [2022] on the smaller sized gecko mesh (a mesh smaller than all our examples), and using the fastest locomotion optimization parameters we've tested (200 simulation steps, 200 CMAES iterations, population size of 16), would take an expected optimization time of at least 17 hours. Using this same analysis, all the examples in Table. 1 are generously at least 17x faster than if we had used a full space simulation to optimize for their locomotion.

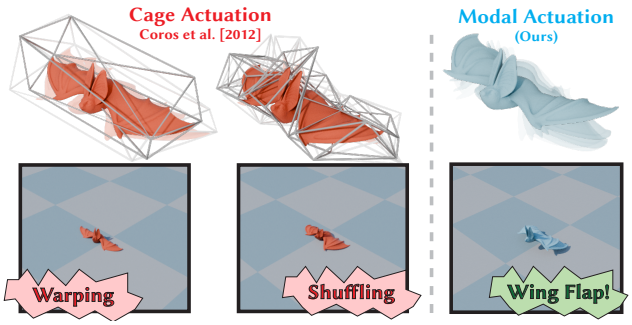


Fig. 7. Using a cage-based actuation introduces warping effects for low-resolution cages, or shuffling behaviors at high-resolution cages. Our modal actuation allows the bat to take more semantically intuitive motions to locomote, such as flapping its wings.

Independence on Mesh Resolution. Building further on our reduced space complexity, Fig. 6 shows the optimized locomotions of two different *Arc de Triomphe* meshes: a visibly coarse one, and another that is finer by a factor of 10. Despite large differences in resolutions the optimization time, as averaged over 5 different trials, is negligible between the two meshes. In contrast, traditional full space local-global simulators [Bouaziz et al. 2014] have an $O(n^2)$ complexity to the resolution of the mesh, and an increase in resolution by a factor of 10 would result in a corresponding optimization time hit by a factor of 100. Moreover, because our actuation subspace is defined via the elastic energy Hessian, which is convergent to the continuum of the geometry, we also get qualitatively similar locomotions across mesh resolutions.

Comparison to Different Actuation Subspaces. Fig. 7 compares our actuation subspace with one defined by a coarse embedding cage, as proposed by Coros et al. [2012]. A bat making use of these embedding cages can indeed find locomotions, however with a cage that is too low-dimensional, the locomotions make use of unnatural warping actuations, scaling/shearing the bat excessively. Increasing the resolution of the cage to something that better conforms to the bat's geometry quickly increase dimension of the actuation subspace (for 32 cage vertices, the actuation subspace has 96 DOFs). This increased dimensionality makes the bat prone to finding locomotions with small, shuffling steps, an observation echoed by [Pan and Manocha 2018]. In contrast, our modal actuations allow the bat to take more semantically meaningful deformations, such as flapping its wings, with only 3 actuation degrees of freedom. Another factor that sets us apart from Coros et al. [2012] is our use of a reduced order physics model, which allows us to find locomotions for 3D examples of much higher resolution than theirs.

Exotic Actuation Modes. Although we define our actuation modes through modal analysis, we also benefit from decades of work extending and generalizing modal analysis to obtain vibration modes with different qualities, such as sparsity [Brandt and Hildebrandt 2017], locality [Melzi et al. 2017], discretization independence [Chang et al. 2023] and non-linearity [Duenser et al. 2022; Sharp et al. 2023]. Our method can leverage such advancements; We use the method of Melzi et al. [2017] to design the actuation modes of the octopus to be locally bound to each individual tentacle. Fig. 2 shows such a

Table 1. Locomotion optimization statistics for some of the locomotions shown in Fig. 1. All these locomotions are found through 200 iterations of CMAES with a population size of 16, with a single actuation cluster. Below, **m** is the number of spatial actuation modes, **k** is the number of temporal sinusoids used, **r** is the number of skinning Eigenmodes, $|C_p|$ the number of passive clusters, and $|\mathcal{I}|$ the number of contact samples.

Mesh	#Vertices	# Tets	m	k	r	$\lvert C_p\rvert$	$\lvert\mathcal{I}\rvert$	J	Sim. Steps	Opt. Time(s)
Couch	13, 920	64,154	10	2	5	5	20	-0.63	300	$9.84e^2$
Creepy Tree	42,015	200,487	6	2	5	5	40	-0.39	300	$3.3e^3$
Bat	9,266	34,720	3	2	5	10	20	-1.25	300	$9.00e^2$
Bearded Dragon	45,041	180, 406	6	2	7	10	30	-0.97	200	$1.27e^3$
Octopus	13,893	48,514	16	2	6	20	20	-1.07	300	$3.59e^3$
Treefrog	13,771	54,154	5	3	5	10	30	- 1.22	200	$1.28e^3$

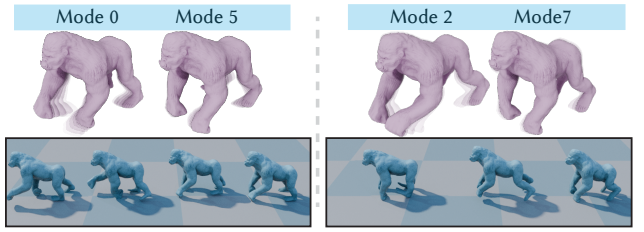
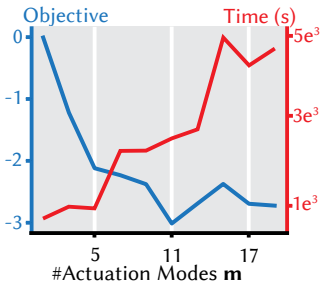


Fig. 8. Using an actuation subspace composed of spatial modes moving the gorilla's limbs in sync (Left) vs. out of sync (Right). The synchronity of the limbs in the actuation subspace is reflected in the final optimized gait.

local modal actuation, and Fig. 9 shows the resulting locomotions these can generate.

Varying Actuation Subspace Dimension. We motivate the importance of making use of a low dimensional actuation subspace in our optimization. We optimize a gorilla's locomotion parameters using an increasing number of spatial actuation modes, and track the resulting objective and total computation time until convergence. The inset shows that an increased dimensionality for our spatial actuation prior is met with a plateauing of the character locomotion's objective, while requiring more optimization time until convergence.



Motion Style Control. Aside from allowing a user to select local bounds to each actuation mode, our actuation subspace provides other intuitive avenues for control.

By selecting which vibration modes to construct the actuation subspace, a user can generate different locomotion gaits. For example, Fig. 8 shows a user picking different sets of actuation modes, one set that have the gorilla's limbs moving asynchronously, and another with limbs moving in synchrony. The resulting synchronicity of the limbs in the actuation subspace is intuitively reflected in the locomotion generated.

Yet another way to imbue user guidance into our framework is by having users constrain different parts of the temporal actuation parameters. Fig. 9 shows a user specifying target periods our temporal sinusoids take on in order to generate locomotions varying in

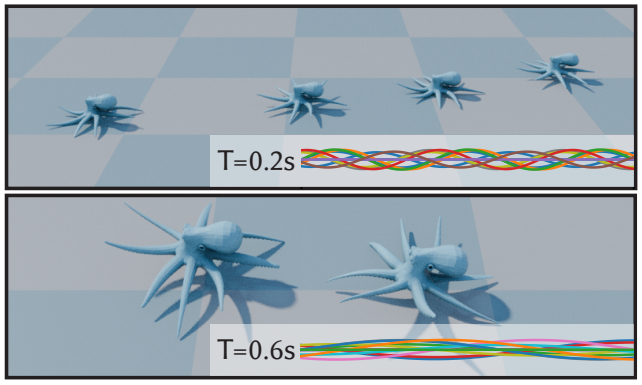


Fig. 9. A user can design faster-paced gaits by specifying the frequencies with which to actuate the actuation modes, and letting the optimizer identify the sinusoidal amplitudes $\boldsymbol{A}$ and phase shifts $\boldsymbol{\theta}$.

style. Larger periods create more of a sluggish crawl for the octopus, while the smaller period makes the octopus skitter across the terrain with smaller steps.

Co-Optimizing for Actuation Modes. While we have found selecting the first 10 actuation modes as a default to usually generate viable locomotions, we can also ask our CMAES optimization to select *which* subset of the top m actuation modes to use. This requires a simple tweak in our optimization, where we add to our optimization degrees of freedom an actuation mode participation vector $\boldsymbol{\sigma} \in \mathbb{R}^m$, whose highest valued entries chose the selected actuation modes used in the optimization. Note that these can rapidly be queried and updated throughout the optimization without requiring any full-space matrix recomputations. Fig. 12 shows the resulting locomotion for the bat, changing the number of desired queried control modes from 2 to 3. While both provide locomotion, querying for three modes unexpectedly converges to a less optimal solution than querying only for two. This happens due to the non-linear nature of our optimization problem with respect to the actuation parameters, making it susceptible to local minima.

Penalizing Work Done. As shown in Fig. 10 We can obtain slower styles of motion by appending an objective to $J(\boldsymbol{x}(t))$ that penalizes energy expenditure from actuation:

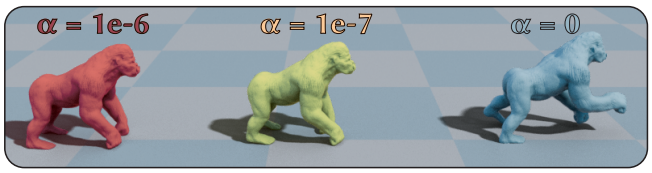


Fig. 10. Varying weighting of the work done objective penalizes energy expenditure by the gorilla. Too strong of a penalization (left), results in no motion after 200 timesteps. Decreasing α towards 0 allows the gorilla to travel further in less time.

$$J_{work} = \alpha \sum_{t_i=t_0}^{t_1} (z(i) - z(t_{i-1}))^T (f^a(t_i) - Q^a z(t_i)) dt \quad (18)$$

A low value for α ignores energy expenditure, allowing the gorilla to locomote as fast as possible. A high value for α penalizes energy expenditure, and inhibits how far the gorilla can move. Setting this parameter too high can result in no locomotion at all.

Jumping. While our focus is on locomotion, Fig. 11 shows our actuation subspace can also be used to generate jumping motions for the bat. The only change required in our controller optimization lies in replacing the J_{disp} objective with an objective that rewards mean center of mass height, $J_{height} = \text{mean}(x_{com_y})$.

Note on Realism. The locomotions we obtain for many biologically inspired characters like the octopus are not *realistic*, in the sense that a real living octopus wouldn't move forward by pushing against its back tentacles as shown in Fig. 9. This isn't surprising, our actuation structure by design has no concept of the real-world anatomy of an octopus. Instead, our actuation forms a fictional *virtual* musculature, generating motions that are *geometrically* plausible, requiring little manual effort from the user to achieve a viable locomotion.

Robustness Across Geometries. Finally, and most importantly, our method provides a framework for designing locomotions for characters of arbitrary geometry, without burdening the user with meticulous design of muscle fibers and bone connectivity [Geijtenbeek et al. 2013; Min et al. 2019; Tan et al. 2012]. Fig. 1 shows locomotion on 11 very different high resolution soft body characters, each with their own unique actuation space and locomotion behavior.

Discussion on choice of CMAES. Our method reduces the controller optimization problem from Eq. (1) to one with a small, low dimensional set of open-loop control parameters.

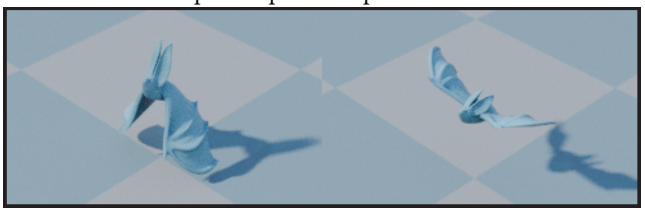


Fig. 11. Using only a single control mode to parameterize our motion prior, we can create an open loop jumping controller for the bat.

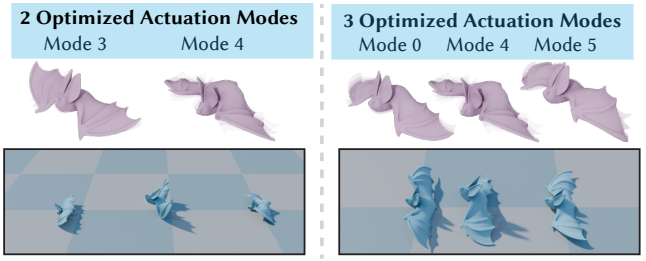


Fig. 12. We can optimize for which control modes the bat should use to locomote, instead of selecting them arbitrarily.

CMAES is known for its ability to solve such low-dimensional optimization problems [Hansen 2006], and has already been used for similar control tasks in graphics [Tan et al. 2011a].

In contrast, while Deep Reinforcement Learning (DRL) can learn more sophisticated high-dimensional optimization problems it is notoriously *sample inefficient* [Jain et al. 2020] and hyper-parameter sensitive [Hämäläinen et al. 2020], requiring more simulation evaluations to converge to a viable locomotion.

Alternatively, gradient-based optimization techniques [Shen et al. 2024] guarantee convergence to a locally optimal solution, but require much user guidance [Mordatch et al. 2012; Pan and Manocha 2018], either in the form of a strong initial guess, or through the use of guiding data. These methods further require the availability of simulation gradients [Geilinger et al. 2020], where differentiating through contact remains an open challenge, with even less work providing solutions for a reduced-space regime. While extending our actuation subspace to a DRL or gradient-based optimization scheme remains exciting future work, the off-the-shelf simplicity of CMAES highlights the effectiveness of our actuation space for generating natural locomotion.

5 CONCLUSION AND FUTURE WORK

We have shown that the pairing of a periodic, energy efficient actuation subspace with a reduced space soft body simulation provides a simple but powerful framework for designing locomotion for highly detailed characters with arbitrary deformable geometries.

We have many exciting avenues for future work. First, our actuation subspace is built entirely off the elastic vibration modes of the character in isolation. However, we believe a stronger actuation subspace could be built if we additionally made us of prior information regarding expected contact points and forces.

While our local-global solver is fast for softer characters, these solvers are known to converge slowly for highly stiff elastic materials [Brown and Narain 2021]. If one wanted to model locomotions for characters composed of extremely stiff materials, like steel and bones, we would recommend switching to other reduced solvers that are better equipped for dealing with such stiffnesses [Trusty et al. 2023], which would come at the cost of abandoning our fast constant system factorization or introducing additional discretization [Li et al. 2020] .

Finally, we are especially excited about the possibility of combining our framework for more general and robust control tasks in order to make soft-body characters replace articulated rigid ones as the default in the field of character animation.

ACKNOWLEDGMENTS

We would like to thank Towaki Takikawa for proof reading, and Sarah Fernandez for emotional support. This research was made possible with the administrative help of our lab system administrator John Hancock and financial officer Xuan Dam. This research was funded by an NSERC Discovery grant and Roblox Research.

REFERENCES

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph.* 28, 3, Article 53 (jul 2009), 9 pages. https://doi.org/10.1145/1531326.1531359

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (jul 2005), 982–990. https://doi.org/10.1145/1073204.1073300

Otman Benchekroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. 2023. Fast Complementary Dynamics via Skinning Eigenmodes. arXiv:2303.11886 [cs.GR]

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Trans. Graph.* 38, 6, Article 206 (nov 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. 2021. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems* 34 (2021).

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4, Article 154 (jul 2014), 11 pages. https://doi.org/10.1145/2601097.2601116

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201387

Christopher Brandt and Klaus Hildebrandt. 2017. Compressed vibration modes of elastic bodies. *Computer Aided Geometric Design* 52-53 (2017), 297–312. https://doi.org/10.1016/j.cagd.2017.03.004 Geometric Modeling and Processing 2017.

George E Brown and Rahul Narain. 2021. WRAPD: weighted rotation-aware ADMM for parameterization and deformation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Jinxiang Chai and Jessica K. Hodgins. 2005. Performance animation from low-dimensional control signals. *ACM Trans. Graph.* 24, 3 (jul 2005), 686–696. https://doi.org/10.1145/1073204.1073248

Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2023. LiCROM: Linear-Subspace Continuous Reduced Order Modeling with Neural Fields. arXiv:2310.15907 [cs.GR]

Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable Objects Alive! *ACM Trans. Graph.* 31, 4, Article 69 (jul 2012), 9 pages. https://doi.org/10.1145/2185520.2185565

Simon Duenser, Bernhard Thomaszewski, Roi Poranne, and Stelian Coros. 2022. Nonlinear Compliant Modes for Large-Deformation Analysis of Flexible Structures. *ACM Trans. Graph.* 42, 2, Article 21 (nov 2022), 11 pages. https://doi.org/10.1145/3568952

Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph.* 39, 6, Article 190 (nov 2020), 15 pages. https://doi.org/10.1145/3414685.3417766

Pierre A Guertin. 2009. The mammalian central pattern generator for locomotion. *Brain research reviews* 62, 1 (2009), 45–56.

Nikolaus Hansen. 2006. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms* (2006), 75–102.

Niksolaus Hansen, Youhei Akimoto, and Petr Baudis. 2019. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634. https://doi.org/10.5281/zenodo.2559634

Nicolas Heess, Dhruva Tb, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Trans. Graph.* 39, 4, Article 53 (aug 2020), 13 pages. https://doi.org/10.1145/3386569.3392440

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Trans. Graph.* 36, 4, Article 42 (jul 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs* (Kobe, Japan) *(SA '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 4 pages. https://doi.org/10.1145/2820903.2820918

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2019. DiffTaichi: Differentiable Programming for Physical Simulation. *CoRR* abs/1910.00935 (2019). arXiv:1910.00935 http://arxiv.org/abs/1910.00935

Suning Huang, Boyuan Chen, Huazhe Xu, and Vincent Sitzmann. 2024. DittoGym: Learning to Control Soft Shape-Shifting Robots. arXiv:2401.13231 [cs.RO]

Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. 2020. PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6. https://doi.org/10.1109/MLSP49062.2020.9231618

Alexandru-Eugen Ichim, Petr Kadleček, Ladislav Kavan, and Mark Pauly. 2017. Phace: physics-based face modeling and animation. *ACM Trans. Graph.* 36, 4, Article 153 (jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073664

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Trans. Graph.* 31, 4, Article 77 (jul 2012), 10 pages. https://doi.org/10.1145/2185520.2185573

Sumit Jain and C Karen Liu. 2011. Modal-space control for articulated characters. *ACM Transactions on Graphics* 30, 5 (2011), 118.

Vidhi Jain, Simin Liu, and Ganesh Iyer. 2020. Coping with sample inefficiency of deep-reinforcement learning (DRL) for embodied AI.

Marilyn Keller, Keenon Werling, Soyong Shin, Scott Delp, Sergi Pujades, C. Karen Liu, and Michael J. Black. 2023. From Skin to Skeleton: Towards Biomechanically Accurate 3D Digital Humans. *ACM Trans. Graph.* 42, 6, Article 253 (dec 2023), 12 pages. https://doi.org/10.1145/3618381

Junggon Kim and Nancy S Pollard. 2011a. Direct control of simulated nonhuman characters. *IEEE Computer Graphics and Applications* 31, 4 (2011), 56–65.

Junggon Kim and Nancy S Pollard. 2011b. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 1–19.

Theodore Kim and David Eberle. 2022. Dynamic Deformables: Implementation and Production Practicalities (Now with Code!). In *ACM SIGGRAPH 2022 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 7, 259 pages. https://doi.org/10.1145/3532720.3535628

P.G. Kry, L. Reveret, F. Faure, and M.-P. Cani. 2009. Modal Locomotion: Animating Virtual Characters with Natural Vibrations. *Computer Graphics Forum* 28, 2 (2009), 289–298. https://doi.org/10.1111/j.1467-8659.2009.01368.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01368.x

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–13.

Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.* 28, 4, Article 99 (sep 2009), 17 pages. https://doi.org/10.1145/1559755.1559756

Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–11.

Jing Li, Tiantian Liu, and Ladislav Kavan. 2020. Soft articulated characters in projective dynamics. *IEEE Transactions on Visualization and Computer Graphics* 28, 2 (2020), 1385–1396.

Siwang Li, Jin Huang, Mathieu Desbrun, and Xiaogang Jin. 2013. Interactive elastic motion editing through space–time position constraints. *Computer Animation and Virtual Worlds* 24, 3-4 (2013), 409–417.

Chen Liang, Xifeng Gao, Kui Wu, and Zherong Pan. 2023. Learning Reduced-Order Soft Robot Controller. arXiv:2311.01720 [cs.RO]

Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. 2020. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. *CoRR* abs/2011.07215 (2020). arXiv:2011.07215 https://arxiv.org/abs/2011.07215

Libin Liu, KangKang Yin, Bin Wang, and Baining Guo. 2013. Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph.* 32, 6, Article 215 (nov 2013), 8 pages. https://doi.org/10.1145/2508363.2508427

Pingchuan Ma, Tao Du, John Z Zhang, Kui Wu, Andrew Spielberg, Robert K Katzschmann, and Wojciech Matusik. 2021. DiffAqua: A Differentiable Computational Design Pipeline for Soft Underwater Swimmers with Shape Interpolation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 132.

Simone Melzi, Emanuele Rodolà, Umberto Castellani, and Michael M. Bronstein. 2017. Localized Manifold Harmonics for Spectral Shape Analysis. *CoRR* abs/1707.02596 (2017). arXiv:1707.02596 http://arxiv.org/abs/1707.02596

Sehee Min, Jungdam Won, Seunghwan Lee, Jungnam Park, and Jehee Lee. 2019. SoftCon: Simulation and Control of Soft-Bodied Animals with Biomimetic Actuators. *ACM Trans. Graph.* 38, 6, Article 208 (nov 2019), 12 pages. https://doi.org/10.1145/3355089.3356497

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 4, Article 43 (jul 2012), 8 pages. https://doi.org/10.1145/2185520.2185539

Rubens F. Nunes, Joaquim B. Cavalcante-Neto, Creto A. Vidal, Paul G. Kry, and Victor B. Zordan. 2012. Using Natural Vibrations to Guide Control for Locomotion.

In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Costa Mesa, California) *(I3D '12)*. Association for Computing Machinery, New York, NY, USA, 87–94. https://doi.org/10.1145/2159616.2159631

Zherong Pan and Dinesh Manocha. 2018. Active Animations of Reduced Deformable Models with Environment Interactions. *ACM Trans. Graph.* 37, 3, Article 36 (aug 2018), 17 pages. https://doi.org/10.1145/3197565

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.* 37, 4, Article 143 (jul 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics* 40, 4, Article 1 (July 2021), 15 pages.

A. Pentland and J. Williams. 1989. Good Vibrations: Modal Dynamics for Graphics and Animation. *SIGGRAPH Comput. Graph.* 23, 3 (jul 1989), 207–214. https://doi.org/10.1145/74334.74355

Avinash Ranganath, Avishek Biswas, Ioannis Karamouzas, and Victor Zordan. 2021. Motor Babble: Morphology-Driven Coordinated Control of Articulated Characters. In *Proceedings of the 14th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Virtual Event, Switzerland) *(MIG '21)*. Association for Computing Machinery, New York, NY, USA, Article 6, 10 pages. https://doi.org/10.1145/3487983.3488291

Daniele Reda, Tianxin Tao, and Michiel van de Panne. 2020. Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Virtual Event, SC, USA) *(MIG '20)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3424636.3426907

Junior Rojas, Eftychios Sifakis, and Ladislav Kavan. 2021. Differentiable implicit soft-body physics. *arXiv preprint arXiv:2102.05791* (2021).

Shunsuke Saito, Zi-Ye Zhou, and Ladislav Kavan. 2015. Computational bodybuilding: Anatomically-based modeling of human bodies. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

Peter Hans Schoenemann. 1964. *A solution of the orthogonal Procrustes problem with applications to orthogonal and oblique rotation.* University of Illinois at Urbana-Champaign.

Nicholas Sharp, Cristian Romero, Alec Jacobson, Etienne Vouga, Paul G Kry, David IW Levin, and Justin Solomon. 2023. Data-Free Learning of Reduced-Order Kinematics. (2023).

Siyuan Shen, Tianjia Shao, Kun Zhou, Chenfanfu Jiang, Sheldon Andrews, Victor Zordan, and Yin Yang. 2024. Elastic Locomotion with Mixed Second-order Differentiation. arXiv:2405.14595 [cs.GR] https://arxiv.org/abs/2405.14595

Olga Sorkine and Marc Alexa. 2007. As-Rigid-as-Possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association, Goslar, DEU, 109–116.

Sebastian Starke, Ian Mason, and Taku Komura. 2022. DeepPhase: periodic autoencoders for learning motion phase manifolds. *ACM Trans. Graph.* 41, 4, Article 136 (jul 2022), 13 pages. https://doi.org/10.1145/3528223.3530178

Jie Tan, Yuting Gu, Greg Turk, and C Karen Liu. 2011a. Articulated swimming creatures. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–12.

Jie Tan, Karen Liu, and Greg Turk. 2011b. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications* 31, 4 (2011), 34–44.

Jie Tan, Greg Turk, and C. Karen Liu. 2012. Soft body locomotion. *ACM Trans. Graph.* 31, 4, Article 26 (jul 2012), 11 pages. https://doi.org/10.1145/2185520.2185522

Ty Trusty, Otman Benchekroun, Eitan Grinspun, Danny M. Kaufman, and David I.W. Levin. 2023. Subspace Mixed Finite Elements for Real-Time Heterogeneous Elastodynamics. In *SIGGRAPH Asia 2023 Conference Papers* (<conf-loc>, <city>Sydney</city>, <state>NSW</state>, <country>Australia</country>, </conf-loc>) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 112, 10 pages. https://doi.org/10.1145/3610548.3618220

Ty Trusty, Danny Kaufman, and David I.W. Levin. 2022. Mixed Variational Finite Elements for Implicit Simulation of Deformables. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 40, 8 pages. https://doi.org/10.1145/3550469.3555418

Xiaoyuan Tu and Demetri Terzopoulos. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 43–50.

Wikipedia contributors. 2023. Orthogonal Procrustes problem — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Orthogonal_Procrustes_problem&oldid=1190317617 [Online; accessed 18-May-2024].

Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3, Article 44 (sep 2021), 22 pages. https://doi.org/10.1145/3480148

Pei Xu, Kaixiang Xie, Sheldon Andrews, Paul G Kry, Michael Neff, Morgan McGuire, Ioannis Karamouzas, and Victor Zordan. 2023. AdaptNet: Policy Adaptation for Physics-Based Character Control. *ACM Transactions on Graphics* 42, 6 (2023). https://doi.org/10.1145/3618375

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: simple biped locomotion control. *ACM Trans. Graph.* 26, 3 (jul 2007), 105–es. https://doi.org/10.1145/1276377.1276509

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Trans. Graph.* 37, 4, Article 144 (jul 2018), 12 pages. https://doi.org/10.1145/3197517.3201397

A SKINNING EIGENMODE POSITIONAL SUBSPACE

We reduce our positional degrees of freedom $\boldsymbol{x}$ with a linear Skinning Eigenmode subspace [Benchekroun et al. 2023],

$$\boldsymbol{x} = \boldsymbol{B}(\boldsymbol{W})\boldsymbol{z} \tag{19}$$

where $\boldsymbol{z} \in \mathbb{R}^r$ are the reduced space vertex positions.

$\boldsymbol{B}(\boldsymbol{W}) \in \mathbb{R}^{3n\times r}$ is a subspace matrix, parameterized by a set of Linear Blend Skinning weights $\boldsymbol{W} \in \mathbb{R}^w$, with $r = 12w$. We obtain these weights from a modal analysis on the elastic energy Laplacian, $\boldsymbol{L} =\in \mathbb{R}^{n\times n}$,

$$\boldsymbol{LW} = \boldsymbol{MW\Lambda}. \tag{20}$$

$\boldsymbol{B}$ is then constructed from $\boldsymbol{W}$ via the equation:

$$\boldsymbol{B} = \boldsymbol{I}_3 \otimes (\boldsymbol{W} \otimes \boldsymbol{1}_4^T) \odot (\boldsymbol{1}_m^T \otimes [\boldsymbol{X}\ \boldsymbol{1}_n]), \tag{21}$$

Where $\boldsymbol{X} \in \mathbb{R}^{n\times 3}$ is a matrix of stacked vertex positions at rest.

B FULL SPACE PASSIVE ENERGY

The passive term is standardly comprised of a kinetic energy E_k, an external potential energy for gravity E_f, and an elastic potential energy which we choose to be ARAP [Kim and Eberle 2022; Sorkine and Alexa 2007] for all our examples E_v,

$$E_p(\boldsymbol{x}) = E_k(\boldsymbol{x}) + E_f(\boldsymbol{x}) + E_v(\boldsymbol{x}) \tag{22}$$

$$\begin{aligned} E_k(\boldsymbol{x}) &= \frac{1}{2h^2}||\boldsymbol{x}-\boldsymbol{y}||_{\boldsymbol{M}}^2, \\ E_f(\boldsymbol{x}) &= \boldsymbol{x}^T\boldsymbol{Mg}, \\ E_v(\boldsymbol{x}) &= \frac{1}{2}\sum_e^{|\mathcal{T}|} \min_{\boldsymbol{R}_e} \mu_e V_e ||\boldsymbol{F}_e(\boldsymbol{x}) - \boldsymbol{R}_e||_F^2 \quad s.t.\ \boldsymbol{R}_e \in \mathcal{SO}(3). \end{aligned} \tag{23}$$

Above, h is the simulation timestep size, $\boldsymbol{y} \in \mathbb{R}^{3n}$ are inertial positions, $\boldsymbol{g} \in \mathbb{R}^{3n}$ is the acceleration due to gravity acting on each vertex, $|\mathcal{T}|$ is the set of all mesh elements, μ_e and V_e are the per-element elastic stiffness and volume respectively. $\boldsymbol{F}_e(\boldsymbol{x})$ is the per-element deformation Jacobian of the simulated character, while $\boldsymbol{R}_e$ is its corresponding best fit rotation matrix.

C REDUCING THE PASSIVE ENERGY

We seek to detach the evaluation of the passive energy $E_p(\boldsymbol{x})$, defined in App. B, with the resolution of the mesh. We follow an approach similar to [Benchekroun et al. 2023], where we approximate our full space positions $\boldsymbol{x} = \boldsymbol{Bz}$, with the Skinning Eigenmode subspace from App. A. To mitigate the evaluation of per-element rotations $\boldsymbol{R}_e$ in the elastic potential energy, we make use of the clustering scheme from [Jacobson et al. 2012], where multiple elements may share the same rotation. We call these clusters *passive*, as they group together rotations from our passive elastic energy term. These are distinct from our *active* clusters, which also allow for elements in a cluster to share the same rotation matrix.

Our active term finds the rotation matrix that aligns $\boldsymbol{F}_e(\boldsymbol{z})$ to $\boldsymbol{Y}_e(\boldsymbol{a})$, whereas the passive term finds the rotation matrix that aligns $\boldsymbol{F}_e(\boldsymbol{z})$ to the identity. A distinction that can incur significant implementational differences in reduction.

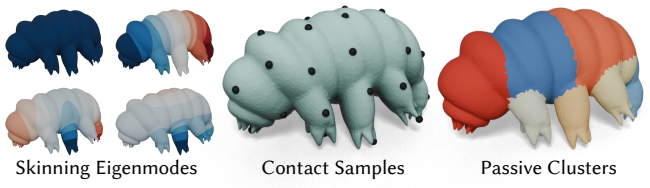


Fig. 13. We visualize the three simulation subspace parameters, namely 4 Skinning Eigenmodes, 10 passive muscle clusters and 30 contact sample points.

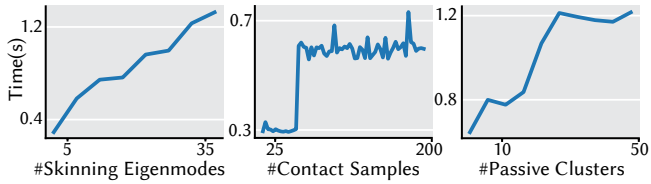


Fig. 14. Timings of 200 simulation steps as we vary three simulation subspace parameters of our reduced simulation, the number of skinning eigenmodes, contact samples, and passive muscle clusters. With the default for all quantities not being varied are 5 skinning eigenmodes, 40 contact samples, and 5 passive clusters.

Plugging in our active positional subspace and our rotation clustering scheme brings us to the reduced form of the passive elastic energy, which can be evaluated entirely in a reduced space.

$$\begin{aligned} \tilde{E}_k(z) &= \frac{1}{2h^2}||\boldsymbol{z}-\boldsymbol{q}||_{\boldsymbol{N}}^2 \\ \tilde{E}_f(z) &= \boldsymbol{z}^T\boldsymbol{B}^T\boldsymbol{Mg}, \\ \tilde{E}_v(z) &= \frac{1}{2}\min_{\boldsymbol{R}_c}\sum_e^{|\mathcal{T}|} \mu_e V_e ||\boldsymbol{F}_e(\boldsymbol{z}) - \boldsymbol{R}_{c(e)}||_F^2 \\ &\quad s.t.\ \boldsymbol{R}_c \in \mathcal{SO}(3) \forall c \in |C_p|. \end{aligned} \tag{24}$$

Fig. 13 visualize our Skinning Eigenmode Subspace and passive clusters on the tardigrade, while Fig. 14 explores how thease two parameters affect simulation computation time.

D GLOBAL STEP DERIVATION FOR LOCAL-GLOBAL SOLVER

To solve for the character vertex locations in the reduced space $\boldsymbol{z}$, we make use of a local-global solver. The local step optimize for a set of rotations both from our passive energy, and from our actuation energy, holding $\boldsymbol{z}$ fixed. These optimal rotations are derived in App. G. The global step requires us to optimize for $\boldsymbol{z}$, while holding our rotation variables fixed.

We state our reduced space optimization problem

$$\begin{aligned} \boldsymbol{z} &= \underset{\boldsymbol{z}}{\text{argmin}}\ \tilde{E}_p(\boldsymbol{z}) + \tilde{E}_a(\boldsymbol{z}, \boldsymbol{a}) \\ &= \tilde{E}_k(\boldsymbol{z}) + \tilde{E}_f(\boldsymbol{z}) + \tilde{E}_v(\boldsymbol{z}) + \tilde{E}_a(\boldsymbol{z}, \boldsymbol{a}) \end{aligned} \tag{25}$$

with all of our reduced space energies defined as,

$$
\begin{aligned}
\tilde{E}_k(\boldsymbol{z}) &= \frac{1}{2h^2}||\boldsymbol{z}-\boldsymbol{q}||^2_{\boldsymbol{N}} \\
\tilde{E}_f(\boldsymbol{z}) &= \boldsymbol{z}^T\boldsymbol{B}^T\boldsymbol{M}\boldsymbol{g}, \\
\tilde{E}_v(\boldsymbol{z}) &= \frac{1}{2}\min_{\boldsymbol{R}_c}\sum_e^{|\mathcal{T}|}\mu_e V_e||\boldsymbol{F}_e(\boldsymbol{z})-\boldsymbol{R}_{c(e)}||^2_F \\
&\quad s.t.\ \boldsymbol{R}_c \in \mathcal{SO}(3)\forall c \in |C_p|. \\
\tilde{E}_a(\boldsymbol{a}) &= \frac{1}{2}\min_{\Omega_c}\sum_e^{|\mathcal{T}|}\mu_e V_e||\boldsymbol{F}_e(\boldsymbol{z})-\Omega_{c(e)}\boldsymbol{Y}_e(\boldsymbol{a})||^2_F \\
&\quad s.t.\ \Omega_c \in \mathcal{SO}(3)\forall c \in |C_a|
\end{aligned}
\tag{26}
$$

The global step assumes we've already found the minimizing rotations $\boldsymbol{R}_c$ and Ω_c for the current iterate, and so we can treat these as constants and drop the two nested minimization problems for these two quantities, as well as the constraints that had appeared in the full energy Eq. (11).

Our strategy will be to expand all energies in Eq. (26) and drop any terms that explicitly do not depend on $\boldsymbol{z}$, as they have no bearing on our optimization task.

D.1 Expanding the Elastic Energy $\tilde{E}_v(\boldsymbol{z})$

Starting with the passive energy in Eq. (26), we expand out the Frobenius norm and obtain,

$$
\begin{aligned}
\tilde{E}_v(\boldsymbol{z}) &= \frac{1}{2}\sum_e^{|\mathcal{T}|}\mu_e V_e||\boldsymbol{F}_e-\boldsymbol{R}_{c(e)}||^2_F, \\
&= \sum_e^{|\mathcal{T}|}\frac{\mu_e V_e}{2}tr(\boldsymbol{F}_e^T\boldsymbol{F}_e)-\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)})+\text{const.}
\end{aligned}
\tag{27}
$$

We deal with the first term $\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{F}_e)$, and rewrite the integrand inside the sum in index notation, reintroducing our definition for $\boldsymbol{F}_{eij} = \mathcal{J}^{\boldsymbol{B}}{}_{eijk}z_k$ from App. F.

$$
\begin{aligned}
\mu_e V_e tr({\boldsymbol{F}_e}^T\boldsymbol{F}_e) &= \mu_e V_e(\mathcal{J}^{\boldsymbol{B}}{}_{eijk}z_k)(\mathcal{J}^{\boldsymbol{B}}{}_{eijl})z_l, \\
&= \mu_e V_e\mathcal{J}^{\boldsymbol{B}}{}_{eijk}\mathcal{J}^{\boldsymbol{B}}{}_{eijl}z_k z_l, \\
&= Q^v_{ekl}z_k z_l,
\end{aligned}
$$

with $Q^v_{ekl} = \mu_e V_e\mathcal{J}^{\boldsymbol{B}}{}_{eijk}\mathcal{J}^{\boldsymbol{B}}{}_{eijl}$. Plugging this back into the sum gives us

$$
\sum_e^{|\mathcal{T}|}Q^v_{ekl}z_k z_l = z_k(\sum_e^{|\mathcal{T}|}Q^v_{ekl})z_l = \boldsymbol{z}^T\boldsymbol{Q}^v\boldsymbol{z},
\tag{28}
$$

where we've exposed the Laplacian matrix for our actuation energy $\boldsymbol{Q}^v = \boldsymbol{B}^T\boldsymbol{J}^T(\boldsymbol{V}^v\otimes\boldsymbol{I}_9)\boldsymbol{J}\boldsymbol{B} \in \mathbb{R}^{r\times r}$, and $\boldsymbol{V}^v \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$ being the diagonal μ-weighed volume matrix $V^v_{ee} = \mu_e V_e$.

We turn our attention to the second term in this energy, $\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)})$. With a similar treatment, we rewrite it in index notation,

$$
\begin{aligned}
\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)}) &= \mu_e V_e(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)})_{ii}, &(29)\\
&= \mu_e V_e F_{ij}R_{c(e)ij}, \\
&= \mu_e V_e(\mathcal{J}^{\boldsymbol{B}}{}_{eijk}z_k)(P_{ec}R_{cij}), \\
&= (\mu_e V_e\mathcal{J}^{\boldsymbol{B}}{}_{eijk}P_{ec})R_{cij}z_k, &(30)
\end{aligned}
$$

where we have made use of a $|\mathcal{T}|\times|C|$ cluster selection matrix $\boldsymbol{P}$ to select out the tetrahedron inside each cluster.

$$
P_{ec} = \begin{cases} 1 & \text{if } e \in \mathcal{T}(c), \\ 0 & \text{otherwise.} \end{cases}.
\tag{31}
$$

With this selection matrix in hand, we can rewrite the second term going through the full summation,

$$
\begin{aligned}
-\sum_e^{|\mathcal{T}|}\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)}) &= -z_k(\sum_e^{|\mathcal{T}|}\mu_e V_e\mathcal{J}^{\boldsymbol{B}}{}_{eijk}P_{ec})R_{cij}, \\
&= z_k K^v_{ijkc}R_{cij}, \\
&= \boldsymbol{z}^T(\mathcal{K}^v : \boldsymbol{R}),
\end{aligned}
\tag{32}
$$

Where we've revealed the tensor $\mathcal{K}^v \in \mathbb{R}^{r\times3\times3\times|C|}$, given by

$$
\mathcal{K}^v = \text{reshape}(-\boldsymbol{B}^T\boldsymbol{J}^T((\boldsymbol{V}^v\boldsymbol{P})\otimes\boldsymbol{I}_9), \{r,3,3,|C|\}).
\tag{33}
$$

Abstracting this away further we arrive at

$$
-\sum_e^{|\mathcal{T}|}\mu_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{R}_{c(e)}) = \boldsymbol{z}^T\boldsymbol{f}^v,
\tag{34}
$$

Finally, we can rewrite the reduced clustered passive muscle energy as a quadratic,

$$
\tilde{E}_v(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{Q}^v\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{f}^v + \text{const},
\tag{35}
$$

D.2 Expanding $\tilde{E}_a(\boldsymbol{z},\boldsymbol{a})$

Starting with the actuation energy in Eq. (11), we expand out the Frobenius norm:

$$
\tilde{E}_a(\boldsymbol{z}) = \sum_e^{|\mathcal{T}|}\gamma_e V_e||\boldsymbol{F}_e-\Omega_{c(e)}\boldsymbol{Y}_e||^2_F,
\tag{36}
$$

$$
= \sum_e^{|\mathcal{T}|}\gamma_e V_e tr(\boldsymbol{F}_e^T\boldsymbol{F}_e) - 2\gamma_e V_e tr(\boldsymbol{F}_e^T\Omega_{c(e)}\boldsymbol{Y}_e)+\text{const.}
\tag{37}
$$

Just like the first section App. D.1, the first term amounts to the simple quadratic form

$$
\sum_e^{|\mathcal{T}|}\gamma_e V_e tr(\boldsymbol{F}_t^T\boldsymbol{F}_t) = \boldsymbol{z}^T\boldsymbol{Q}^a\boldsymbol{z}
\tag{38}
$$

with the actuation muscle Laplacian $\boldsymbol{Q}^a = \boldsymbol{B}^T\boldsymbol{J}^T(\boldsymbol{V}^v\otimes\boldsymbol{I}_9)\boldsymbol{J}\boldsymbol{B} \in \mathbb{R}^{r\times r}$, where $\boldsymbol{V}^v \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$ is the γ weighed diagonal volume matrix such that $V^v_{ee} = V_e\gamma_e$.

For the second term we follow the same procedure as in the previous section, rewriting the integrand in index notation.

$$
\begin{aligned}
-\gamma_t tr(\boldsymbol{F}_t^T\Omega_{c(t)}\boldsymbol{Y}_t) &= -\gamma_t F_{twi}\Omega_{c(t)wk}Y_{tki} \\
&= -\gamma_t(\mathcal{J}^{\boldsymbol{B}}{}_{twid}z_d)(P_{tc}\Omega_{cwk})(\mathcal{J}^{\boldsymbol{D}}{}_{tkil}\bar{a}_l) \\
&= z_d(-\gamma_t\mathcal{J}^{\boldsymbol{B}}{}_{twid}\mathcal{J}^{\boldsymbol{D}}{}_{tkil}P_{tc})\Omega_{cwk}\bar{a}_l
\end{aligned}
$$

Plugging this into the sum allows us to express it in a our canonical form

$$\begin{aligned}
&\sum_{e}^{|\mathcal{T}|} -\gamma_e V_e tr(\boldsymbol{F}_e^T \Omega_{c(e)} \boldsymbol{Y}_e), \\
&= z_d (\sum_{e}^{|\mathcal{T}|} -\gamma_e V_e \mathcal{J}^{\boldsymbol{B}}{}_{ewid} \mathcal{J}^{\boldsymbol{D}}{}_{ekil} P_{ec}) \Omega_{cwk} \bar{a}_l, \\
&= z_d O_{wkdlc} \Omega_{cwk} \bar{a}_l, \\
&= \boldsymbol{z}^T ((O \vdots \Omega) : \bar{a}), \\
&= \boldsymbol{z}^t \boldsymbol{f}^a,
\end{aligned} \tag{39}$$

with $O \in \mathbb{R}^{3\times3\times r\times m_s\times|C|}$. where $\boldsymbol{f}^a = ((O \vdots \Omega) : \bar{a}) \in \mathbb{R}^r$. We can finally rewrite our actuation energy as the quadratic,

$$\tilde{E}^a(\boldsymbol{z}) = \boldsymbol{z}^T \boldsymbol{Q}^a \boldsymbol{z} + 2\boldsymbol{z}^T \boldsymbol{f}^a + \text{const.} \tag{40}$$

D.3 Expanding Inertial Energy $\tilde{E}_k(\boldsymbol{z})$

$$\tilde{E}_k(\boldsymbol{z}) = \frac{1}{2h^2}||\boldsymbol{z} - \boldsymbol{q}||^2_{\boldsymbol{N}} = \frac{1}{2h^2}\boldsymbol{z}^T \boldsymbol{N}\boldsymbol{z} - \frac{1}{h^2}\boldsymbol{z}^T \boldsymbol{N}\boldsymbol{q} + \text{const.} \tag{41}$$

Introducing $\boldsymbol{Q}^k = \frac{1}{h^2}\boldsymbol{N} \in \mathbb{R}^{r\times r}$, as well as $\boldsymbol{f}^k = -\frac{1}{h^2}\boldsymbol{N}\boldsymbol{q} \in \mathbb{R}^{\boldsymbol{r}}$, we can write the kinetic energy simply as the quadratic energy,

$$E_k(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{Q}^k \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{f}^k + \text{const} \tag{42}$$

D.4 Expanding External Potential Energy $\tilde{E}_f(\boldsymbol{z})$

This term is basically all expanded already, we only have to simplify it in our canoncial form,

$$E_f(\boldsymbol{z}) = \boldsymbol{z}^T \boldsymbol{B}^T \boldsymbol{M}\boldsymbol{g}, \tag{43}$$

$$= \boldsymbol{z}^T \boldsymbol{f}^f, \tag{44}$$

with $\boldsymbol{f}^f = \boldsymbol{B}^T \boldsymbol{M}\boldsymbol{g}$.

D.5 Putting it All Together

Assembling all the terms of our energy that depend on $\boldsymbol{z}$ leaves us with the total quadratic energy,

$$\tilde{E}(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T (\boldsymbol{Q}^a + \boldsymbol{Q}^v + \boldsymbol{Q}^k)\boldsymbol{z} + \boldsymbol{z}^T (\boldsymbol{f}^a + \boldsymbol{f}^v + \boldsymbol{f}^k + \boldsymbol{f}^f) + \text{const.} \tag{45}$$

Taking the minimizer of this quadratic energy finally reveals the system (without contact) we need to solve every timestep:

$$\begin{aligned}
(\boldsymbol{Q}^v + \boldsymbol{Q}^a + \boldsymbol{Q}^k)\boldsymbol{z} &= (\boldsymbol{f}^a + \boldsymbol{f}^v + \boldsymbol{f}^k) \\
\boldsymbol{Q}\boldsymbol{z} &= \boldsymbol{f}
\end{aligned} \tag{46}$$

Note that $\boldsymbol{Q}^v + \boldsymbol{Q}^a + \boldsymbol{Q}^k$ are constant matrices, and so the system factorization can be solved once and reused throughout the course of the entire optimization problem.

Fig. 15 shows our iteration step cost breakdown on the Arc de Triomphe character.

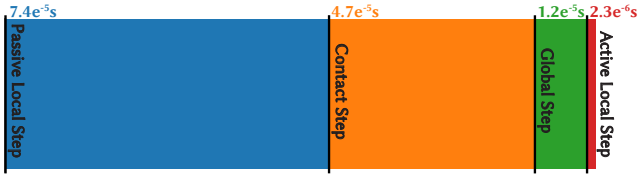


Fig. 15. Timing breakdown for an average simulation step with our default 10 local-global iterations on the Arc de Triomphe character.

E COMPUTING CONTACT FORCES $\boldsymbol{f}_c$

To account for contact, we add an extra force term to the right hand side of Eq. (46), $\boldsymbol{f}_c \in \mathbb{R}^r$,

$$\boldsymbol{Q}\boldsymbol{z} = \boldsymbol{f} + \boldsymbol{f}_c. \tag{47}$$

This extra force term $\boldsymbol{f}_c$ is computed every local-global iteration, before the global step to ensure $\boldsymbol{z}$ upholds the zero interpenetration, damping friction conditions,

$$\boldsymbol{J}^c_N \dot{\boldsymbol{z}} = \boldsymbol{0}, \tag{48}$$

$$\boldsymbol{J}^c_T \dot{\boldsymbol{z}} = \mu \boldsymbol{J}^c_T \dot{\boldsymbol{z}}_{prev}, \tag{49}$$

where $\mu \in [0, 1]$ is a velocity damping coefficient. A low value for μ results in extreme friction with no allowed tangential velocity, while a high value results in slipper no-friction. All our examples use $\mu = 0.2$ $\boldsymbol{J}^c_N \in \mathbb{R}^{|\mathcal{I}_c|\times r}$ is a contact Jacobian that maps from our reduced velocity $\dot{\boldsymbol{z}}$ to per-contact point normal velocities. Likewise $\boldsymbol{J}^c_T \in \mathbb{R}^{2|\mathcal{I}_c|\times r}$ maps to tangential velocities. The above two equations can be summarized by the constraint,

$$\boldsymbol{J}^c \dot{\boldsymbol{z}} = \boldsymbol{v}, \tag{50}$$

where $\boldsymbol{J}^c = [\boldsymbol{J}^c_N\ \boldsymbol{J}^c_T]^T \in \mathbb{R}^{3|\mathcal{I}|\times r}$ is the full contact Jacobian and $\boldsymbol{v} = [\boldsymbol{0}\ \mu \boldsymbol{J}^c_T \dot{\boldsymbol{z}}_{prev}]^T$.

Through finite differences, we rewrite our constraint on the velocity to act on positions DOFs $\boldsymbol{z}$,

$$\begin{aligned}
\boldsymbol{J}^c \frac{\boldsymbol{z} - \boldsymbol{z}_{curr}}{h} &= \boldsymbol{v}, \\
\boldsymbol{J}^c \boldsymbol{z} &= h\boldsymbol{v} + \boldsymbol{J}^c \boldsymbol{z}_{curr}.
\end{aligned} \tag{51}$$

To compute an $\boldsymbol{f}$ that ensures $\boldsymbol{z}$ satisfies this condition, we plug in our global step solve $\boldsymbol{z} = \boldsymbol{Q}^{-1}(\boldsymbol{f} + \boldsymbol{f}_c)$ Eq. (47) into this constraint and obtain,

$$\begin{aligned}
\boldsymbol{J}^c \boldsymbol{Q}^{-1}(\boldsymbol{f} + \boldsymbol{f}_c) &= h\boldsymbol{v} + \boldsymbol{J}^c \boldsymbol{z}_{curr}, \\
\boldsymbol{J}^c \boldsymbol{Q}^{-1} \boldsymbol{f}_c &= h\boldsymbol{v} + \boldsymbol{J}^c \boldsymbol{z}_{curr} - \boldsymbol{J}^c \boldsymbol{Q}^{-1} \boldsymbol{f}, \\
\boldsymbol{L}\boldsymbol{f}_c &= \boldsymbol{b},
\end{aligned} \tag{52}$$

With $\boldsymbol{L} = \boldsymbol{J}^c \boldsymbol{Q}^{-1}$ and $\boldsymbol{b} = h\boldsymbol{v} + \boldsymbol{J}^c \boldsymbol{z}_{curr} - \boldsymbol{J}^c \boldsymbol{Q}^{-1} \boldsymbol{f}$. The matrix $\boldsymbol{Q}^{-1} \in \mathbb{R}^{r\times r}$ is a dense matrix, but it's also small and constant, and so the product $\boldsymbol{L} = \boldsymbol{Q}^{-1} \in \mathbb{R}^{|\mathcal{I}|\times r}$ can be precomputed once and reused throughout the entire optimization procedure. The system described by Eq. (52) can either be over or underdetermined, depending on whether the number of contacting degrees of freedom $3|\mathcal{I}|$ is greater or less than the dimensionalilty of our simulation subspace r.

If it's overdetermined, we solve for $\boldsymbol{f}_c$ in a least squares sense,

$$\begin{aligned} \boldsymbol{f}_c &= \underset{\boldsymbol{f}_c}{\operatorname{argmin}} ||\boldsymbol{L}\boldsymbol{f}_c - \boldsymbol{b}||^2, \\ &\Downarrow \\ \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{f}_c &= \boldsymbol{L}^T\boldsymbol{b}. \end{aligned} \tag{53}$$

If it's underdetermined, we solve for the smallest $\boldsymbol{f}_c$ satisfying our constraint,

$$\begin{aligned} \boldsymbol{f}_c &= \underset{\boldsymbol{f}_c}{\operatorname{argmin}} ||\boldsymbol{f}_c||^2 \quad \text{s.t.} \boldsymbol{L}\boldsymbol{f}_c = \boldsymbol{b}, \\ &\Downarrow \\ \boldsymbol{f}_c &= \boldsymbol{L}^T(\boldsymbol{L}\boldsymbol{L}^T)^{-1}\boldsymbol{b}. \end{aligned} \tag{54}$$

F TENSOR PRECOMPUTATION FOR CLUSTERING ROTATIONS

We start with the assumption that we have access to the simulation Skinning Eigenmode Subspace $\boldsymbol{B} \in \mathbb{R}^{3n\times r}$ Eq. (21), $\boldsymbol{D} \in \mathbb{R}^{3n\times(m_s+1)}$ obtained from Eq. (4), $\boldsymbol{J} \in \mathbb{R}^{9|T|\times 3n}$ the standard deformation Jacobian mapping matrix [Kim and Eberle 2022].

We construct the subspace deformation Jacobian mapping matrix $\boldsymbol{J}^B \in \mathbb{R}^{9|T|\times r}$ and $\boldsymbol{J}^D \in \mathbb{R}^{9|T|\times(m_s+1)}$:

$$\boldsymbol{J}^B = \boldsymbol{J}\boldsymbol{B} \qquad\qquad \boldsymbol{J}^D = \boldsymbol{J}\bar{\boldsymbol{D}}$$

We reorder each of the entries in the form of a new 4-th order tensor $\mathcal{J}^B \in \mathbb{R}^{|T|\times 3\times 3\times r}$ and $\mathcal{J}^D \in \mathbb{R}^{|T|\times 3\times 3\times(m_s+1)}$.

$$\begin{aligned} \mathcal{J}^B &= \text{reshape}(\boldsymbol{J}^B, \{|T|, 3, 3, r\}), \\ \mathcal{J}^D &= \text{reshape}(\boldsymbol{J}^D, \{|T|, 3, 3, m_s+1\}) \end{aligned}$$

Which we can use to express the deformation Jacobian and the control target's deformation Jacobian matrices, $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{T}|\times 3\times 3}$ and $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{T}|\times 3\times 3}$, using index notation,

$$F_{eij} = \mathcal{J}^B_{eijk} z_k, \quad Y_{eij} = \mathcal{J}^D_{eijl} \bar{a}_l, \tag{55}$$

where a repeated index implies a component wise product along that dimension if the index is also present on the other side of the equation, or a standard dot product along that dimension if the index is not repeated on the other side of the equation.

We re-express $\boldsymbol{F}_e\boldsymbol{Y}_e$, the matrix that appears inside the sum of Eq. (72),

$$\begin{aligned} C_{eij} &= F_{eik} Y_{ejk}, \\ &= (\mathcal{J}^B_{eiku} z_u)(\mathcal{J}^D_{ejkv} \bar{a}_v), \\ &= (\mathcal{J}^B_{eiku} \mathcal{J}^D_{ejkv}) z_u \bar{a}_v, \\ &= \mathcal{K}_{eijuv} z_u \bar{a}_v, \end{aligned} \tag{56}$$

revealing the constant tensor $\mathcal{K} \in \mathbb{R}^{|\mathcal{T}|\times 3\times 3\times r\times(m_s+1)}$, which, when multiplied against the changing $\boldsymbol{z}$ and $\bar{\boldsymbol{a}}$, provides us with the per-tet matrix product of $\boldsymbol{F}_e\boldsymbol{Y}_e$ every timestep.

We then introduce a grouping matrix $\boldsymbol{G} \in \mathbb{R}^{|C|\times|\mathcal{T}|}$, a matrix that assembles for each cluster a γ-weighed sum of its member tetrahedra's scalar quantities.

$$G_{ce} = \begin{cases} \gamma_e & \text{if } e \in \mathcal{T}(c) \\ 0 & \text{otherwise} \end{cases} \tag{57}$$

This matrix allows us now to fully express the averaged covariance matrix $\boldsymbol{H} \in \mathbb{R}^{|C|\times 3\times 3}$ in index notation,

$$H_{cij} = G_{ce} C_{eij} \tag{58}$$

Finally, plugging in our expression for C_{eij}, we have

$$\begin{aligned} H_{cij} &= G_{ce} \mathcal{K}_{eijuv} z_u \bar{a}_v \\ &= \mathcal{H}_{cijuv} z_u \bar{a}_v \end{aligned} \tag{59}$$

Where we finally expose the constant tensor $\mathcal{H}_{cijuv}$, which multiplies $\boldsymbol{z}$ and $\bar{\boldsymbol{a}}$ and provides the covariance matrices for all clusters $\boldsymbol{H}$.

G OPTIMAL CLUSTERED ROTATIONS

We want to derive the optimal rotation Ω_c for each cluster c. Starting from Eq. (11), computing the optimal rotation Ω_c for each c amounts to the following optimization problem

$$\Omega_c = \underset{\Omega_c \in SO(3)}{\operatorname{argmin}} \sum_{e}^{|T(c)|} \gamma_t V_e ||\boldsymbol{F}_e - \Omega_c \boldsymbol{Y}_e||_F^2 \tag{60}$$

As presented in Eq. (12), we will show that the optimal solution is given by the polar decomposition:

$$\Omega_c = \text{polar}\left(\sum_{e}^{|\mathcal{T}(c)|} \gamma_e V_e \boldsymbol{F}_e \boldsymbol{Y}_e^T \right). \tag{61}$$

The same derivation for the active clustered rotation Ω_c can be extended to the optimal rotation $\boldsymbol{R}$ for the passive term E_v by setting $\boldsymbol{Y}_e = \boldsymbol{I}_{3\times 3}$ for every tetrahedral element e.

Proof. We start by expanding each term in the summation of Eq. (61) as

$$\gamma_e V_e ||\boldsymbol{F}_e - \Omega_c \boldsymbol{Y}_e||_F^2 \tag{62}$$

$$= \gamma_e V_e tr\left((\boldsymbol{F}_e - \Omega_c \boldsymbol{Y}_e)^T (\boldsymbol{F}_e - \Omega_c \boldsymbol{Y}_e) \right) \tag{63}$$

$$= \gamma_e V_e tr\left(\boldsymbol{F}_e^T \boldsymbol{F}_e + \boldsymbol{Y}_e^T \boldsymbol{Y}_e - \boldsymbol{F}_e^T \Omega_c \boldsymbol{Y}_e - \boldsymbol{Y}_e^T \Omega_c^T F_e \right) \tag{64}$$

$$= -\gamma_e V_e tr\left(2\boldsymbol{F}_e^T \Omega_c \boldsymbol{Y}_e \right) + \text{const} \tag{65}$$

$$= -\gamma_e V_e tr\left(2\Omega_c \boldsymbol{Y}_e \boldsymbol{F}_e^T \right) + \text{const} \tag{66}$$

Above, in the second equality we made use of the identity $||\boldsymbol{A}||_F^2 = tr(\boldsymbol{A}^T\boldsymbol{A})$. In the third equality we expanded the multiplication, and used the orthogonality of rotation matrices $\Omega_c^T\Omega_c = \boldsymbol{I}$. In the fourth equality we dropped terms that do not depend on the variable we are optimizing over and placed them inside the constant c. In the fifth and final equality we have made use of the trace's invariance under permutation of its input. We've also used the identity that the trace is invariant under transposes $tr(A + A^T) = tr(2A)$.

With the above result, we return to our full optimization problem (Eq. (61)) and rewrite it as:

$$\Omega_c = \underset{\Omega_c \in SO(3)}{\text{argmin}} \sum_{e}^{|T(c)|} -\gamma_e V_e 2tr\left(\Omega_c Y_e F_e^T\right) + \text{const} \tag{67}$$

$$= \underset{\Omega_c \in SO(3)}{\text{argmax}} \sum_{e}^{|T(c)|} \gamma_e V_e tr\left(\Omega_c Y_e F_e^T\right) \tag{68}$$

$$= \underset{\Omega_c \in SO(3)}{\text{argmax}}\ tr\left(\sum_{e}^{|T(c)|} \gamma_e V_e \Omega_c Y_e F_e^T\right) \tag{69}$$

$$= \underset{\Omega_c \in SO(3)}{\text{argmax}}\ tr\left(\Omega_c \sum_{e}^{|T(c)|} \gamma_e V_e F_e^T Y_e\right) \tag{70}$$

$$= \underset{\Omega_c \in SO(3)}{\text{argmax}}\ tr\left(\Omega_c \boldsymbol{H}_c\right) \tag{71}$$

In the second equality we have dropped the const and switched to a maximization instead of a minimization, while also dropping the factor of 2 from the optimization. In the third equality we have made use of the linearity of the trace operator and the sum operator, allowing them to commute. In the fourth equality, we finally pull out the per-cluster rotation Ω_c from the sum, as all tets in $|T(c)|$ share the same rotation matrix by definition. This final form is an instance of the *Orthogonal Procrustes* problem, where the target matrix we are trying to match is $\boldsymbol{H}_c$. Based on the proof in Schoenemann [1964]; Wikipedia contributors [2023], the optimal solution to the Orthogonal Procrustes problem can be obtained from the polar decomposition of $\boldsymbol{H}_c$.

$$\Omega_c = \text{polar}(\boldsymbol{H}_c) = \text{polar}\left(\sum_{e}^{|\mathcal{T}(c)|} \gamma_e V_e F_e Y_e^T\right) \tag{72}$$

□

H ACTUATION LIMITS

The controller optimization problem defined by the objective Eq. (17) is ill-defined: there is a large space of undesireable overly energetic motions that allow a character to locomote, such as an extremely large initial actuation that launches the character forward into the air.

As such, our method requires sensible limits to the actuation parameters in Eq. (6). Fortunately, many of these parameters are intuitive to define. We provide typical allowable ranges used throughout all the experiments above below.

$$\boldsymbol{T}_{i,j} \in [0.2, 0.6] \tag{73}$$

$$\boldsymbol{\theta}_{i,j} \in [0, \pi/2] \tag{74}$$

$$\boldsymbol{A}_{i,j} \in [-a_{max_i}(p), a_{max_i}(p)] \tag{75}$$

We derive a method to automatically set the maximum actuation mode amplitude a_{max_i} through a more intuitive unitless scalar parameter p, which can be reused across a wide range of characters, regardless of scale and proportions.

This parameter p describes the maximum amount of unweighed Dirichlet energy measured by any tetrahedron. It measures the maximum amount of scale/compression each of our tetrahedra tolerates.

$$p = \max_{t \in \mathcal{T}} ||F_t(\boldsymbol{a}) - \boldsymbol{I}||_F^2 = \max_{t \in \mathcal{T}} \boldsymbol{a}^T \boldsymbol{D}^T \boldsymbol{J}_t^T \boldsymbol{J}_t \boldsymbol{D}\boldsymbol{a} \tag{76}$$

We re-express this parameter p by further assuming each mode i is actuated independently. Rewriting the above expression, keeping only the relevant column $\boldsymbol{D}_i$ of $\boldsymbol{D}$, and entry a_i of $\boldsymbol{a}$:

$$p = \max_{t \in \mathcal{T}} \boldsymbol{D}_i^T \boldsymbol{J}_t^T \boldsymbol{J}_t \boldsymbol{D}_i a_i^2 = \max_{t \in \mathcal{T}} W_{it} a_i^2 \tag{77}$$

Where we introduced the scalar $W_{it} = \boldsymbol{D}_i^T \boldsymbol{J}_t^T \boldsymbol{J}_t \boldsymbol{D}_i$ that maps from an actuation of mode i to its resulting stretch/compression at tetrahedron t. Finally, given a target p, it's very easy to go the other way and determine a corresponding value for each actuation mode a_{max_i}. This gives:

$$a_{max_i} = \frac{p}{\sqrt{\max_{t \in \mathcal{T}} W_{it}}} \tag{78}$$

A value of 0 indicates we tolerate no scaling/compression, and so a_{max_i} will always give 0. As we increase p, we increase the maximum amount of scaling any one of our character's tetrahedra can tolerate. For all experiments we set $p = 1$.