SoftCon: Simulation and Control of Soft-Bodied Animals with Biomimetic Actuators

SEHEE MIN, Seoul National University
JUNGDAM WON, Seoul National University
SEUNGHWAN LEE, Seoul National University
JUNGNAM PARK, Seoul National University
JEHEE LEE, Seoul National University

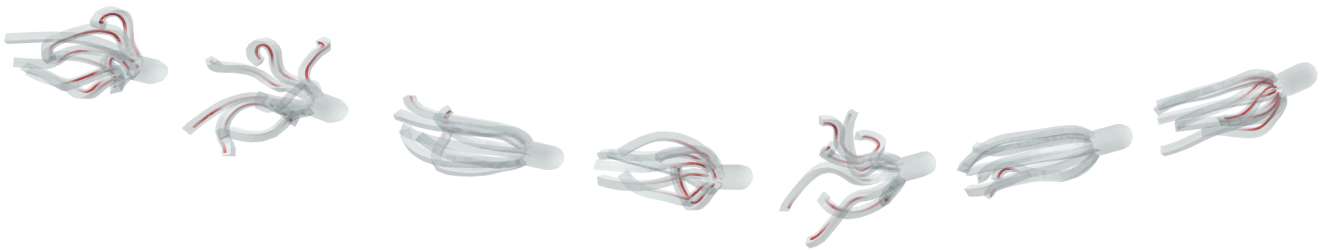


Fig. 1. The octopus swims by actuating muscles embedded in the soft tissues.

We present a novel and general framework for the design and control of underwater soft-bodied animals. The whole body of an animal consisting of soft tissues is modeled by tetrahedral and triangular FEM meshes. The contraction of muscles embedded in the soft tissues actuates the body and limbs to move. We present a novel muscle excitation model that mimics the anatomy of muscular hydrostats and their muscle excitation patterns. Our deep reinforcement learning algorithm equipped with the muscle excitation model successfully learned the control policy of soft-bodied animals, which can be physically simulated in real-time, controlled interactively, and resilient to external perturbations. We demonstrate the effectiveness of our approach with various simulated animals including octopuses, lampreys, starfishes, stingrays and cuttlefishes. They learn diverse behaviors such as swimming, grasping, and escaping from a bottle. We also implemented a simple user interface system that allows the user to easily create their creatures.

CCS Concepts: • **Computing methodologies** → *Animation*; *Physical simulation*; *Reinforcement learning*; Volumetric models.

Additional Key Words and Phrases: character animation, deformable character, soft-bodied animal, finite element method, physics-based control, optimal control, reinforcement learning

Authors' addresses: Sehee Min, Department of Computer Science and Engineering, Seoul National University, sehee@mrl.snu.ac.kr; Jungdam Won, Department of Computer Science and Engineering, Seoul National University, nonaxis@gmail.com; Seunghwan Lee, Department of Computer Science and Engineering, Seoul National University, lsw9021@mrl.snu.ac.kr; Jungnam Park, Department of Computer Science and Engineering, Seoul National University, jungnam04@mrl.snu.ac.kr; Jehee Lee, Department of Computer Science and Engineering, Seoul National University, jehee@mrl.snu.ac.kr.

ACM Reference Format:
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 (November 2019), 12 pages. https://doi.org/10.1145/3355089.3356497

1 INTRODUCTION

Soft-bodied animals are common types of animals on Earth. Most soft-bodied animals on the ground are small (e.g., worms) because they do not have internal supporting structures that help them resist gravity, whereas soft-bodied animals under the water can grow bigger (e.g., giant squids). Soft-bodied animals lack hard parts, such as skeletons and shells, and their bodies are mainly composed of fleshes and muscles. There have been a series of studies in biology as to how soft-bodied animals, such as octopuses and squids, move their bodies and perform all the functions that are usually performed by skeletons in vertebrates [Yekutieli et al. 2003]. Soft-bodied manipulators, such as octopus arms, elephant trunks, and vertebrate tongues, are called *muscular hydrostats*. Compared to articulated arms, muscular hydrostats are *hyper-redundant* manipulators having a large number of degrees of freedom. For example, the octopus arm has long muscle fibers along the entire length of the arm, and every single segment can be activated independently to have a continuum of controllable muscle fibers.

The passive simulation of deformable objects has long been an active research topic in computer graphics and employed in film making, in game development, and even in material science for simulating various material and relevant physical phenomena [Smith et al. 2018]. However, there have been only a few studies on active simulation of deformable soft-bodied animals and their control. There are several technical challenges in creating interactively controllable soft-bodied creatures. The challenges include the physics-based

modeling and simulation of soft tissues, muscles, and their contraction dynamics, the robust control of highly redundant dynamics systems, and mimicking the biological motion of live creatures.

In this paper, we present a novel and general framework for the design and control of soft-bodied animals. We are particularly interested in underwater animals with a variety of body shapes and actuation structures. The soft tissues and muscles are modeled by tetrahedral Finite Element Method (FEM) meshes. Nerve cords deliver excitation signals to cause muscle contraction. We developed a new muscle excitation model, which we call *Adaptive Propagation Muscle Excitation Model* (AP-MEM). Our model is biomimetic in the sense that it mimics the way muscle excitation signals propagate along muscular hydrostats. The new muscle excitation model is a key component that allows us to reproduce swimming patterns that resemble those of real creatures. We incorporated the muscle excitation model into a Deep Reinforcement Learning (DRL) algorithm to construct the controllers of soft-bodied animals, which can be physically simulated in real-time, controlled interactively, and resilient to an external perturbation such as collisions.

We will demonstrate various simulated animals including lampreys, starfishes, octopuses, stingrays, and cuttlefishes. The simulated animals learned to swim and perform goal-driven (task-based) maneuvers. We also developed a simple user interface system that allows the user to easily create their creatures.

2 RELATED WORK

Physically Simulated Characters. The research on physically simulated characters has pursued improving physical realism, responsiveness to the user's control, and robustness against external perturbation. Although human-like bipeds have been at the center of interests for decades [Coros et al. 2010; da Silva et al. 2008; Hodgins et al. 1995; Laszlo et al. 1996; Lee et al. 2018, 2014; Liu et al. 2016; Sok et al. 2007; Wang et al. 2012; Ye and Liu 2010; Yin et al. 2007], quadruped [Coros et al. 2011], multi-legged [Fang et al. 2013], swimming [Grzeszczuk et al. 1998; Pan and Manocha 2018; Tan et al. 2011; Tu and Terzopoulos 1994], flying [Ju et al. 2013; Pan and Manocha 2018; Wu and Popović 2003], and even imaginary creatures [Coros et al. 2012; Tan et al. 2012] have also been studied and demonstrated. Recently, DRL has gained great attention in physics-based character control due to its powerful capability of learning controllers [Liu and Hodgins 2018; Peng et al. 2018a, 2016, 2017, 2018b; Won et al. 2017, 2018; Yu et al. 2018]. It has been reported that DRL is adept at dealing with high-dimensional state and action spaces without selecting hand-crafted features. Since the dynamic states of soft-bodied animals and their continuum muscle fibers are inherently high-dimensional, DRL suits well for learning the control policies of soft-bodied animals.

Passive Simulation of Deformable Objects. The passive simulation of deformable objects (e.g., clothes, hairs, rubbers, or muscles) has long been an important research topic in computer graphics. Solving the equation of motion based on implicit integration has been regarded as a de facto method that achieves the stability and efficiency in deformable object simulation. Baraff and Witkin [1998] presented a fast simulation technique for mass-spring cloth models by solving the implicit formulation of the dynamics system, where the potential energy induced by springs is approximated by Taylor series expansion. Muller et al. [2007] reproduced the physical phenomena of various objects only in the domain of their positions, which is called *position-based dynamics*. Martin et al. [2011] showed that solving the equation of motion by implicit integration can be rewritten as a non-convex optimization of the potential energy, which leads to an example-based simulation by constructing potential energies from examples. Bouaziz et al. [2014] proposed *projective dynamics*, which introduces auxiliary projection variables to solve the non-convex optimization formulation more efficiently. It has been reported that *projective dynamics* substantially improves the stability and efficiency of deformable object simulation. Recently, Brandt and his colleagues [2018] developed a model reduction technique for *projective dynamics*, which enables real-time simulation of large-scale deformable meshes. We adopted *projective dynamics* as a basic simulation method for our soft-bodied animals.

Active Deformable Character Control. Although most of the physics-based characters have been skeleton-driven in computer animation, deformable characters have also been studied based on optimal control theory. Coping with high-dimensionality of deformable objects is a key technical challenge and several approaches have been explored to circumvent the dimensionality issue. A straightforward approach is to embed a skeleton into soft deformable tissues in such a way that the character is directly driven by the skeleton [Kim and Pollard 2011; Liu et al. 2013]. Alternatively, model reduction techniques have been studied to reduce the dimensionality and computational complexity of the dynamics model explicitly [Barbič et al. 2009; Barbič and Popović 2008; Fan et al. 2014; Pan and Manocha 2018; Schulz et al. 2014]. The two-dimensional deformable object simulation and contractile elements were used to animate plushies [Bern et al. 2017]. Coros et al. [2012] used the change of rest shapes to control soft deformable characters, which was done by cage-based or example-based adaption. Ijiri et al. [2009] animated jellyfishes using procedural deformation. Propagation of signal is considered a source of character control as in our work. The difference comes from how the contraction of the soft-body is defined. They designed a linear transformation of the local orientation field for the soft-body contraction. Linear force actuators play a role in our method. Tan et al. [2012] formulated the muscle-driven control of soft deformable bodies into Quadratic Programming (QP) with complementary constraints. They embedded line muscle segments into soft tissues to move the body. Our simulated animals also use muscles as base actuators. The key difference is that our novel muscle excitation model imitates soft-bodied animals in nature. Furthermore, the use of DRL makes substantial improvements over QP-based trajectory optimization in such a way that our animals are simulated in real-time and controlled interactively.

Muscles in Soft-bodied Animals. Muscles are the only source of actuation for the animals in nature. The contraction dynamics of muscles produces different actuation patterns from artificial actuators such as motors. The skeletal animation driven by muscles are well-understood in computer graphics [Geijtenbeek et al. 2013; Lee et al. 2018, 2014; Wang et al. 2012]. Skeletal muscles in vertebrae are attached to bones and the functional role of each individual muscle is clearly defined by the attachment sites and its route between

them. Muscles of soft-bodied animals in nature are quite different. For example, the octopus arm is densely packed with muscle fibers arranged in three different groups (longitudinal, transverse, and oblique) and the axial nerve cord runs along the center of the arm [Laschi et al. 2012]. The central motor signal originating from the brain runs down the nerve cord to activate muscle fibers sequentially from central to peripheral. Unlike vertebrates, for which all motor signals originate from the brain and spinal nerves, octopus arms can move even after the arms are cut off from the body. This observation implies that motor signals may originate in the nerve cords along the arm. The harmony of motor signals propagated from central and originated from peripheral produces distinctive wriggling movements of soft-bodied animals.

3 ANIMAL DESIGN

We assume the soft flesh is hyperelastic and use the Finite Element Method (FEM) to simulate the soft flesh and muscle fibers. Assuming n discrete vertices on the mesh, its dynamic states can be represented by the positions of all vertices $\mathbf{q} \in \mathbb{R}^{3n}$, and their velocities $\mathbf{v} \in \mathbb{R}^{3n}$. A discrete-time dynamics system with implicit Euler integration updates positions and velocities as follows:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + h\mathbf{v}_{t+1} \tag{1}$$

$$\mathbf{v}_{t+1} = \mathbf{v}_t + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}}(\mathbf{q}_{t+1}) + \mathbf{f}_{\text{ext}}) \tag{2}$$

where h is the time-step size, $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$ is a mass matrix, $\mathbf{f}_{\text{int}}(\mathbf{q}_{t+1}) = -\nabla E(\mathbf{q}_{t+1}) = -\nabla \sum_i \mathcal{V}_i \Psi_i(\mathbf{q}_{t+1})$ is an internal force computed from the summation of all element-wise elastic strain energy densities $\mathcal{V}$ and corresponding volumes Ψ, and $\mathbf{f}_{\text{ext}}$ is an external force such as hydrodynamic and buoyant force. With implicit Euler integration, both internal force and external force may depend on velocities $\mathbf{v}_{t+1}$ at the next time step as well as positions $\mathbf{q}_{t+1}$. In our case, the hydrodynamics force is supposed to be a function of nodal velocities. We use a mixed implicit-explicit method to simplify the formulation such that both internal and external forces are functions of time t and $t+1$. The drawback of this simplification is the potential instability of the dynamics system that requires the use of smaller time steps. Our formulation of the hydrodynamics circumvents the instability issue to a certain degree since drag force modeling viscous damping of water prevents excessively large hydrodynamics force. Even with larger time steps, the use of constant system matrices derived from *Projective Dynamics* [Bouaziz et al. 2014] achieves better computational performance than the standard FEM simulation with implicit Euler integration.

Projective Dynamics. We briefly overview the key idea of *projective dynamics*. Putting the equation (1) into equation (2) results in a system of linear equation:

$$\left[\mathbf{M} - h^2 \frac{\partial \mathbf{f}}{\partial \mathbf{q}}\right] \mathbf{v}_{t+1} = \mathbf{M}\mathbf{v}_t + h(\mathbf{f}_{\text{int}}(\mathbf{q}_t) + \mathbf{f}_{\text{ext}}) \tag{3}$$

with Taylor expansion of $\mathbf{f}_{\text{int}}(\mathbf{q}_{t+1}) \simeq \mathbf{f}_{\text{int}}(\mathbf{q}_t) + \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q}_{t+1} - \mathbf{q}_t)$ and the Hessian $\frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ of E evaluated at $\mathbf{q} = \mathbf{q}_t$. Once $\mathbf{v}_{t+1}$ is computed by solving the linear system, the positions $\mathbf{q}_{t+1}$ can be computed from equation (1). Typical remediation of Taylor approximation alternates iteratively between solving the linear system and linearizing the forces during the time step $[t, t+1]$, requiring the inverse of the non-constant matrix and evaluating the Hessian at every iteration. Alternatively, we adopt *projective dynamics* which provides an efficient and robust solver for the equivalent problem. The solver divides the problem into parallelizable local solvers followed by a global quadratic problem. Let C_i be *Constraint Manifold* on which the elastic energy density function Ψ_i vanishes. The local minimization problem is formulated as

$$\Psi_i(\mathbf{q}) = \min_{\mathbf{p}_i \in C_i} \frac{k}{2} \|\mathbf{A}_i\mathbf{q} - \mathbf{p}_i\|^2 \tag{4}$$

where k is stiffness, $\mathbf{A}_i$ is a metric for the constraint, and $\mathbf{p}_i$ is the projection on manifold C_i of the current measure $\mathbf{A}_i\mathbf{q}$. The form of C_i and $\mathbf{A}_i$ depends on the choice of the energy model, which will be described in Section 3.1. Depending on the shape of the constraint manifold, the local problem can be non-linear and computationally expensive to find the projection $\mathbf{p}_i$. Fortunately, once projections are all determined at local steps, the global step can be very efficient because the system matrix of equation (3) is constant. Rearranging the equation (3) results in:

$$(\mathbf{M} + h^2\mathbf{L})\mathbf{q}_{t+1} = \mathbf{M}(\mathbf{q}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}) + \mathbf{J}\mathbf{z} \tag{5}$$

where $\mathbf{L} = \sum_i \mathbf{A}_i^T\mathbf{A}_i$, $\mathbf{J} = \sum_i \mathbf{A}_i^T\mathbf{S}_i$, $\mathbf{z} = \sum_i \mathbf{S}_i^T\mathbf{p}_i$, and $\mathbf{S}_i$ are selector matrices, such that $\mathbf{p}_i = \mathbf{S}_i\mathbf{z}$. We factorize the matrix using the Cholesky decomposition in a pre-processing step.

Hydrodynamics. We use a simplified model of hydrodynamics to simulate the interaction between the character and the water. The simplified model computes drag and thrust forces on the surface of the mesh. The drag force approximates viscous damping of water and acts in the direction of the relative velocity to the water. The thrust force is a source of locomotion and acts in the opposite of the surface normal [Tu and Terzopoulos 1994].

$$\begin{aligned} \mathbf{f}_{\text{drag}} &= \tfrac{1}{2}\rho A C_d(\Phi)\|\mathbf{v}_{\text{rel}}\|^2\mathbf{d}, \\ \mathbf{f}_{\text{thrust}} &= -\tfrac{1}{2}\rho A C_t(\Phi)\|\mathbf{v}_{\text{rel}}\|^2\mathbf{n}, \end{aligned} \tag{6}$$

where A is the area of the surface triangle, ρ is the density of fluid, which is approximately $1000kg/m^3$ for water, $\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{water}} - \frac{1}{3}(\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2)$ is the surface velocity relative to the fluid (See figure 2). We omit the index of the surface for simplification. $\mathbf{d} = \frac{\mathbf{v}_{\text{rel}}}{\|\mathbf{v}_{\text{rel}}\|}$ and $\mathbf{n}$ are the direction of the surface velocity and the surface normal, respectively. $C_d(\Phi)$ and $C_t(\Phi)$ are coefficients, which depend on the angle of attack $\Phi = \frac{\pi}{2} - cos^{-1}(\mathbf{n} \cdot \mathbf{v}_{\text{rel}})$. The coefficient plot of the drag force is symmetric, while the coefficient plot of the thrust force is asymmetric. We lump all the drag and thrust forces on the surface into a single vector $\mathbf{f}_{\text{ext}} = \mathbf{f}_{\text{hydro}}$.

3.1 Muscular Hydrostat Anatomy

The core of the FEM simulation is designing material property. Simulating muscular hydrostats requires appropriate material models according to their functional roles. The background elasticity is a fundamental component that preserves the original shape as close as possible. This elasticity is passive and independent of muscle excitation. On top of background elasticity, the contraction and relaxation of muscle fibers generate internal force.

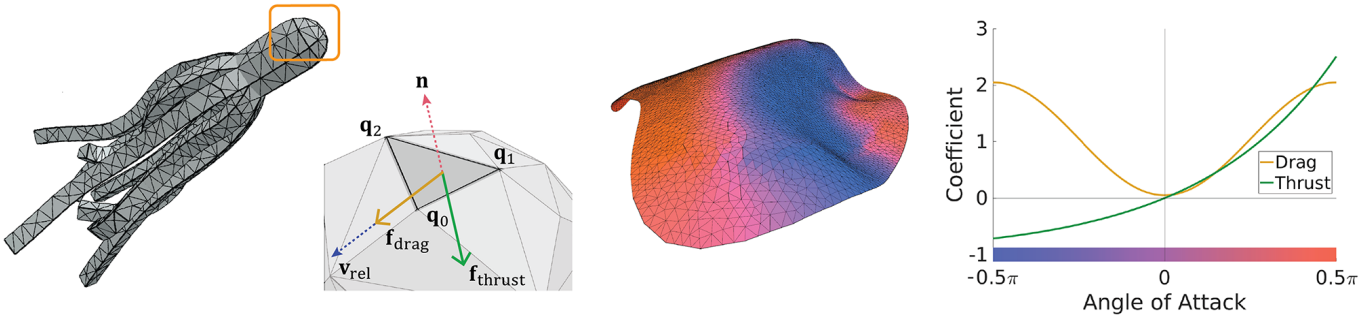


Fig. 2. Simplified Hydrodynamics. (Left) The drag and thrust forces. (Right) The triangles on the curved surface of the swimming stingray are colored by the angle of attack.

Background Elasticity. We choose a corotational elastic strain energy model for background elasticity.

$$\Psi_{\text{passive}}(\mathbf{F}) = \frac{k_{\text{p}}}{2}\|\mathbf{F}-\mathbf{R}\|^2 \tag{7}$$

where $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^\top$ is the deformation gradient of the element, and $\mathbf{R} = \mathbf{U}\mathbf{V}^\top$ is rotational part of the deformation gradient. In the formulation of *projective dynamics*, $\mathbf{A}_i$ maps the nodal position $\mathbf{q}$ to the deformation gradient $\mathbf{F}$, and $\mathbf{p}_i \in SO(3)$ is the projection of the deformation gradient onto the rigid transformation of the rest shape. The element-wise volume preservation energy model projects the deformation gradient to the manifold $\{\mathbf{D} \in \mathbb{R}^{3\times3} | \det(\mathbf{D}) = 1\}$.

$$\Psi_{\text{volume}}(\mathbf{F}) = \frac{k_{\text{v}}}{2}\|\mathbf{F}-\mathbf{D}\|^2 \tag{8}$$

where $\mathbf{D} = \mathbf{U}\Sigma^*\mathbf{V}^\top$, and $\det(\Sigma^*) = 1$. We find $\mathbf{D}$ using constrained quadratic programming. We refer to the work of Bouaziz et al. [2014] for the details on the relevant algebra.

Muscle fibers and cords. The anatomy of the soft-bodied arm comprises several muscle groups with numerous muscle fibers. Physiologically, muscles contract and relax along the direction of muscle fibers. We approximate a continuum of aligned muscle fibers by FEM mesh elements (either tetrahedrons or triangles) and a nerve cord. The muscle excitation signal u travels through the nerve cord, which is discretized into a sequence of segments u_i, and activates nearby FEM elements within a user-specified radius. (See Figure 3) The activated element contracts along the direction of the nearest cord segment. The level of activation of element e is inversely proportional to the distance to the cord l, which can be formulated as $e = u_i \exp(-\sigma l)$. The contraction of an FEM element generates internal force $\mathbf{f}_{\text{muscle}}(e) = -\nabla E_{\text{muscle}}$, where E_{muscle} is a muscle energy model, which depends on the type of muscle fibers. We consider two types, *contractile* and *bending*. The muscle excitation model will be discussed in the next section.

Contractile fibers. The contractile muscle generates anisotropic muscle force $\mathbf{f}_{\text{muscle}}(e) = -f_{\text{muscle}}(e)\mathbf{m}$ where $\mathbf{m}$ is the muscle fiber direction. To generate such an anisotropic muscle force $f_{\text{muscle}}(e)$, we adopt a strain energy model $E_{\text{muscle}} = \mathcal{V}_{\text{muscle}}\Psi_{\text{muscle}}$ [Lee et al. 2018].

$$\Psi_{\text{muscle}}(\mathbf{F}, e) = \frac{k_{\text{m}}}{2}\|(1-r)\mathbf{F}\mathbf{m}\|^2 \tag{9}$$

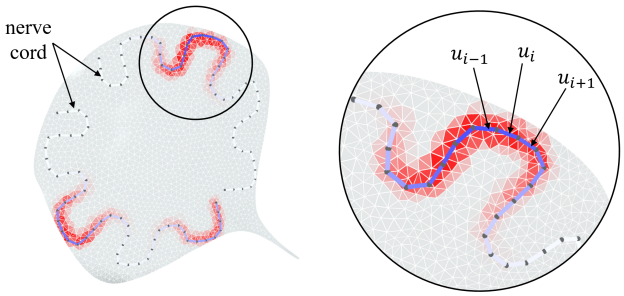


Fig. 3. The nerve cord delivers muscle excitation signal u through muscle fibers approximated by FEM elements. The activated elements are shown in red.

where k_{m} is muscle stiffness, $e \in [0, 1]$ is the level of activation, $r = (1-e)/l$ is the projection of the cord segment. The energy model results in linear force $f_{\text{muscle}}(e) = -k(l-(1-e))$ with a muscle stretch factor $l = \|\mathbf{F}\mathbf{m}\|$. From the viewpoint of *projective dynamics*, r is the projection of the contraction axis given activation e. The global solver balances background elasticity and contractile muscle forces by solving a quadratic optimization problem.

Bending Fibers. Some of the underwater soft-bodied animals have thin body parts such as fins. The whole body of some underwater animals is too thin to model with tetrahedral meshes. Simulating such thin structures with tetrahedral meshes could suffer from an instability issue stemming from the material-space singularity. Instead, we rather utilize a thin-shell model that is actuated by bending muscles. In nature, all muscles produce force by contraction. We consider an imaginary type of muscles that produce force by bending. Bending muscles take a signed excitation signal and the sign indicates the direction of bending. Simulating two-dimensional thin-shells in three-dimensional space requires an energy term on the bending phenomenon. We follow the one-ring bending energy proposed by [Nealen et al. 2006], which uses isotropic deformation represented by the Laplace-Beltrami operator for each edge.

$$\Psi_{\text{bending}}(\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, e) = \frac{k_{\text{b}}}{2}\|\mathbf{X}\mathbf{c} - \mathbf{R}\tilde{\mathbf{X}}(e)\mathbf{c}\|^2 \tag{10}$$

where $\mathbf{q}_0$ and $\mathbf{q}_1$ are vertices constituting an edge and $\mathbf{q}_2$ and $\mathbf{q}_3$ are its adjacent vertices. $\mathbf{X} = (\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3) \in \mathbb{R}^{3\times4}$ is the current position matrix, $\tilde{\mathbf{X}}(e)$ is the rest position matrix deformed by activation $e \in [-1, 1]$, and $\mathbf{c}$ is a set of cotangent weights for the edge. $\mathbf{R} \in SO(3)$ is a projection such that rotating $\tilde{\mathbf{X}}(e)\mathbf{c}$ by $\mathbf{R}$ best matches to $\mathbf{X}(\mathbf{q})\mathbf{c}$. We detail on how activation e updates the rest shape in the Appendix.

3.2 Muscle Excitation Model

We can consider several different models for muscle excitation. The simplest model assumes that all excitation signals on the segments are independent. We call it Independent Muscle Excitation Model (I-MEM). Although I-MEM allows for maximal flexibility in movements and provides the largest degrees of freedom in control, the high-dimensionality of control is a big obstacle for learning motor skills. The direct opposite model to I-MEM assumes that all segments on the entire length of the nerve cord share the same excitation signal. We call it Shared Muscle Excitation Model (S-MEM). Though this model coordinates muscle movements in a structured manner, the model is too simple to reproduce complex behaviors. An alternative to these two extremes is a Central Pattern Generator model (CPG-MEM), which produces cyclic excitation signals from the central nervous system and propagates the signal to peripherals. This model has been popular in computer graphics and robotics because it produces wiggling patterns that look biological [Si et al. 2014; Tian and Lu 2015]. In this model, the central nervous system takes full control over the movements and it modulates the central patterns to change the direction and speed in a strongly-structure manner. We present a new biomimetic muscle excitation model, which we call Adaptive Propagation Muscle Excitation Model (AP-MEM), inspired by signal delivery in muscular hydrostats.

Our model is based on two key observations that can be found in the biology literature. It has been observed from octopuses that nervous signals are produced not only at the brain but also at any part of the limb, as evidenced by amputated octopus arms. The central nervous signals originated from the brain are actively modulated in the peripheral [Levy and Hochner 2017; Richter et al. 2015]. Central nerve signals act as high-level commands from the brain, and the signals are modulated and adjusted to perform skilled control and manipulation. It has also been noted that the speed of neural signal transmission varies in the movement of organisms [Hochner 2012]. The careful modulation of signal propagation speed is as important as signal amplitude modulation. Our AP-MEM is a simple model that satisfies these observations. The AP-MEM is also equipped with a central pattern generator that produces central excitation signals, which are adapted dynamically while propagating to peripherals.

Central Pattern Generator. The various forms of cyclic patterns have been studied and exploited in the composition of center nervous signals. We choose the simple triangular wave function as initial central pattern which is good enough to control all the animals in our examples (see Figure 4). The triangular wave function with range θ_2 and period $t \in [0, \theta_1]$ is

$$TW(t|\theta_1, \theta_2) = \frac{\theta_2}{2}\left[\frac{4}{\theta_1}\left(t - \frac{\theta_1}{4} - \frac{\theta_1}{2}\left\lfloor\frac{2t}{\theta_1}\right\rfloor\right)(-1)^{\left\lfloor\frac{2t}{\theta_1}\right\rfloor} + 1\right], \quad (11)$$

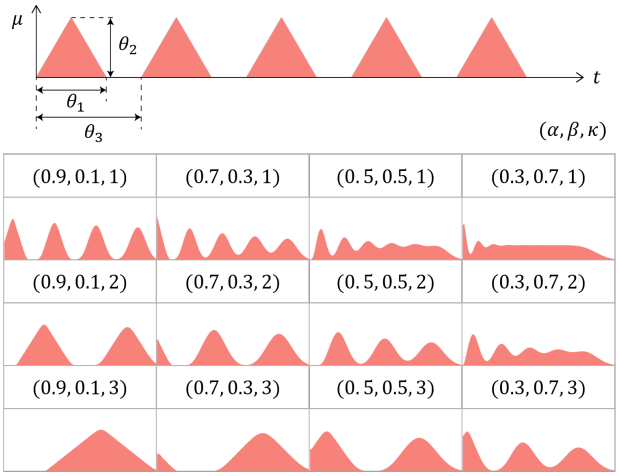


Fig. 4. Triangular central nerve signals at the top propagate through nerve cords when propagation parameters (α, β, κ) are constant.

where $\lfloor\cdot\rfloor$ is the floor function. If the triangular signal occurs repeatedly at the interval $\theta_3 \geq \theta_1$, the central pattern is defined by

$$\mu(t|\theta_1, \theta_2, \theta_3) = \begin{cases} TW(t; \theta_1, \theta_2), & \text{if } t \in [0, \theta_1] \\ 0, & \text{if } t \in (\theta_1, \theta_3]. \end{cases} \quad (12)$$

The presence of delay $(\theta_3 - \theta_1)$ between excitation signals produces a mixture of movement and holding as shown in energetic thrusts in octopus swimming and highly curved S-shapes of ribbon eels.

Adaptive Propagation. The propagation operator delivers excitation signals, u_i, along with nerve cord segments where $i = 0, 1, \cdots$ indexes segments from proximal to distal. While propagating, the signal u_i is modulated spatially by parameters α and β, and temporally by parameter κ when the central signal μ is given to proximal to the muscle cord.

$$u_i(t+1) = \begin{cases} \mu(t), & \text{if } \ i = 0 \\ \alpha u_{i-1}(t) + \beta u_i(t), & \text{otherwise,} \end{cases} \quad (13)$$

The attenuation factor β determines how much the current level of excitation will remain at the next time. The delivery factor α determines how much excitation will be propagated to the next segment. We repeat this step κ times at every control time step to modulate the speed of the propagation process. The propagation operator at time t is defined by

$$\mathbf{u}(t+1) = \mathbf{P}^{\kappa}\mathbf{u}(t) = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & \mu \\ \alpha & \beta & 0 & \dots & 0 & 0 \\ 0 & \alpha & \beta & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \alpha & \beta \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}^{\kappa} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \vdots \\ u_N(t) \\ 1 \end{bmatrix} \quad (14)$$

where $\mathbf{P}$ is a propagation matrix, and N is the number of the segments. If the parameters α, β and κ are constant over time, our

Fig. 5. Dynamics s_{dyn}, task s_{task}, and excitation s_{ext} states. (Left) The positions of selected vertices are colored in yellow and the blue arrows are their velocities. (Middle) The green and red arrows are the target velocity and average velocity of the character, respectively. (Right) The current excitation pattern is colored in pink.

model generates stationary CPG-MEM signals, whereas it modulates the central patterns $\mu(t)$ to control the full-body movements if the parameters change. Learning a CPG-MEM controller is computationally demanding because the action originating at the beginning of the nerve cord can be evaluated at the end of the propagation. The occurrence of an action and its evaluation are distant spatially and temporally. Our AP-MEM takes a different approach to modulating the propagation parameters to add flexibility and controllability in the middle of a long muscle. We found that our AP-MEM better suits for policy learning in the next section.

4 CONTROLLER LEARNING

In reinforcement learning (RL), an agent (a soft-bodied animal in our study) in state $\mathbf{s}$ takes action $\mathbf{a}$, and its state changes to $\mathbf{s}'$ with transition probability $\mathcal{P}(\mathbf{s}, \mathbf{a}, \mathbf{s}') \in [0, 1]$. RL assumes the Markov property, meaning that the transition probability depends only on the current state. The agent receives a scalar-valued reward $r = \mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ in the transition proportional to the desirability of the transition. The initial state $\mathbf{s}_0$ of the agent is chosen from initial state probability $\rho(\mathbf{s}_0)$. The state changes sequentially by taking actions until it meets a termination condition (e.g., time limit and failure detection of the given task). The sum of discounted rewards collected during this process is called a *return* $\mathcal{G}(\mathbf{s}_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots$, where $\gamma \in [0, 1)$ is a discount factor that makes the sum finite. The goal of reinforcement learning is to find a policy (a.k.a. controller) that maximizes the expected value of the return given the initial state distribution.

The state $\mathbf{s} = (\mathbf{s}_{\text{dyn}}, \mathbf{s}_{\text{task}}, \mathbf{s}_{\text{exc}})$ is composed of the dynamical state, task-specific state, and the excitation state (See Figure 5). The dynamical state $\mathbf{s}_{\text{dyn}} = (\mathbf{x}, \dot{\mathbf{x}})$ includes vertex positions $\mathbf{x}$ and their velocities $\dot{\mathbf{x}}$ of the FEM mesh. We found that it is not necessary to include all vertices in the state, but we randomly select k vertices ($n \gg k$) near the nerve cords. The positions and velocities are represented in a local, moving coordinate system attached to the animal's body. The task state $\mathbf{s}_{\text{task}}$ gives the information of user-provided goals where we use the target velocity represented in the local coordinate system for underwater locomotion. $\mathbf{s}_{\text{exc}}$ is the current excitation pattern $\mathbf{u}(t)$. The user provides with central nerve signals $\mu_0(t)$ parameterized by $(\theta_1, \theta_2, \theta_3)$, and the initial values of propagation parameters $(\alpha_0, \beta_0, \kappa_0)$ for each individual nerve cord j. The action is the modulation of the central pattern and propagation parameters. The actual parameters at muscle j are

$$\mu = \mu_0(t) + \Delta\mu, \quad \alpha = \alpha_0 + \Delta\alpha, \quad \beta = \beta_0 + \Delta\beta, \quad \kappa = \kappa_0 + \Delta\kappa, \tag{15}$$

when action $(\Delta\mu, \Delta\alpha, \Delta\beta, \Delta\kappa)$ is applied.

The reward for the underwater navigation task encourages the agent to move along the target velocity.

$$\begin{aligned} \mathcal{R} = {} & w_1 \exp\Big(- \sigma_1 \|v_{\text{d}} - v\| \Big) \\ & + w_2 \exp\Big(- \sigma_2 \|\mathbf{d}_{\text{d}} - \mathbf{d}\|^2 \Big) \\ & + w_3 \exp\Big(- \sigma_3 \|\theta_{\text{up}}\| \Big) \\ & + r_{\text{task}}. \end{aligned} \tag{16}$$

where $v_{\text{d}} = \|\mathbf{v}_{\text{d}}\| \in \mathbb{R}$ and $\mathbf{d}_{\text{d}} = \frac{\mathbf{v}_{\text{d}}}{v_{\text{d}}} \in \mathbb{S}^2$, respectively, are the length, v is the average body velocity during 1 s. and the direction of the target velocity $\mathbf{v}_{\text{d}} \in \mathbb{R}^3$ and $\theta_{\text{up}} \in \mathbb{R}$ is the angle between the up vector and the global upward direction. r_{task} depends on the choice of task objectives.

Our policy and value functions are deep networks with three fully connected layers with 128 units for each layer. The internal layers use *tanh* units for non-linear activation and the last layer uses linear activation units. We adopt Proximal Policy Optimization (PPO) to directly minimize expected returns through stochastic gradient descent [Schulman et al. 2017].

5 EXPERIMENTAL RESULT

Our implementation of the FEM simulation largely follows the tutorial provided by Sifakis and Barbic [2012] and the work of Bouaziz et al. [2014]. For FEM mesh generation, we used *TetWild* [Hu et al. 2018] for 3D tetrahedralization and Delaunay triangulation for thin plates. We typically generate meshes with 400 to 1000 vertices as a trade-off between computational costs and motor functionality (See Table 1). Each nerve cord is divided into 50 segments, which provide a lot of control freedom. The degrees of simulation freedom proportional to the number of vertices is much larger than its control freedom. Muscle routings are specified by Bézier splines. The segments of nerve cords are assigned to tetrahedra by computing triangle-spline intersections.

Since the whole body of our animal model is soft, the local coordinate system attached to the body can be noisy or wiggly under the influence of muscle activation. We pick $C \geq 4$ vertices to attach the coordinate system as far away as possible from muscles (e.g., in the head). The local coordinate system $(\mathbf{R}_{\text{local}}, \mathbf{b}_{\text{local}})$ is determined by solving a minimization problem.

$$\min_{(\mathbf{R}, \mathbf{b}) \in SO(3) \times \mathbb{R}^3} \|\mathbf{X} - (\mathbf{R}\tilde{\mathbf{X}} + \mathbf{b})\|^2 \tag{17}$$

where $\mathbf{X}, \tilde{\mathbf{X}} \in \mathbb{R}^{3 \times C}$ are matrices horizontally stacking the vertices of deformed and material space positions, respectively, while activating muscles by central patterns. This minimization problem can be solved analytically using singular value decomposition. Increasing the Young's modulus around the coordinate system is also helpful in stabilizing the coordinate system.

Our learning algorithm is episodic. It learns a control policy from many simulation episodes. The initial state $\mathbf{s}_0$ of each simulation episode is picked randomly from the pool of states in previous

Table 1. Simulation model data

		Lamprey	Starfish	Octopus	Stingray	Cuttlefish
# of muscles		4	8	32	2	22
# of force actuators	Total	3702	4634	4026	4895	4894
	Passive	1114	1049	2160	2394	1379
	Muscle	2588	3585	1866	2501	3515
# of simulation DoF		1209	1215	2931	1794	3288
# of contour		716	724	1758	997	1470
Muscle stiffness (k_m)		$1.0e^6$	$1.0e^6$	$5.0e^5$	$1.0e^5$	$1.0e^5$
Young's modulus (k_p)		$1.0e^6$	$1.0e^6$	$2.0e^6$	$2.0e^6$	$2.0e^6$
Volume Preservation (k_v)		$1.0e^5$	$2.0e^5$	$1.0e^5$	$2.0e^5$	$2.0e^5$

Table 2. Simulation and learning parameters

Simulation		Learning	
Simulation Hz	240	Policy/Value learning rate	$1.0e^{-5}$
Control Hz	30	Discount factor (γ)	0.95
Max iterations	10	GAE and TD (λ)	0.95
Damping coeff.	0.9999	# of tuples per policy update	512
		Batch size for policy update	64
		Iteration for policy update	10
		Maximum time horizon (sec)	10
		Clip parameter (ϵ)	0.2

episodes. We also adopt an early termination strategy [Peng et al. 2018a] to prune unsuccessful trials. We used the implementation of Proximal Policy Optimization in OpenAI baselines [Dhariwal et al. 2017], where deep network operations are performed by TensorFlow [TensorFlow 2015]. Table 2 shows all parameters used in our implementation. Since the dynamics simulation is the bottleneck of the overall computation, all computations were done on CPU without utilizing GPU acceleration. We use i7-7700 CPUs to generate simulation episodes accelerated by multi-threading. The training requires approximately 4 to 48 hours depending on the number of muscles and tetrahedra and the simulation time step. The low-dimensional, low-energy locomotion for the stingray takes 4 hours, whereas the high-dimensional octopus requires 48 hours until its learning curve plateaus on a single PC. The implementation code is available at https://github.com/seiing/SoftCon.

5.1 Animal Models

The *lamprey* has an eel-like body without backbones. Our lamprey model has four longitudinal muscles along the body (see Figure 6 and Table 1). Tetrahedralization of this simple, long body produces a large dynamics system with 1209 degrees of freedom and 3702 force actuators (passive and muscle). Many degrees of freedom along the continuum of muscle fibers allow the lamprey to bend, tilt, and twist the body about an arbitrary direction. It is possible for the DRL algorithm to learn the locomotion of the lamprey from scratch without any information about muscle excitation patterns. However, the resulting motions would look unpredictable and undesirable. In order to achieve the appearance of structured, biological swimming motions, we provided with central nerve signals that activate two muscles on the left and the other two muscles on the right alternatively. The control policy learns to actuate muscles as agonists and their antagonists and thus produces horizontal wiggling as expected.

The *starfish* is an imaginary soft-bodied animal having four legs that spread out symmetrically. This X-shape animal has also been used in the previous studies [Pan and Manocha 2018; Tan et al. 2012]. Each arm of the starfish includes two longitudinal muscles, which act as an agonist and antagonist pair to produce oscillatory swing patterns. The starfish is the simplest example that learns to swim in a few hours.

Our *octopus* model has eight arms. Each arm has two longitudinal, one transverse, and one oblique muscle (see Figure 8). The agonist and antagonist pair of longitudinal muscles bend and swing the arm. The longitudinal muscles play a primary role in the underwater navigation task. The transverse muscles squeeze the soft tissues and the volume preservation of the soft tissues has the arm stretched. Typical reach movements of the octopus make use of transverse muscles to have its arm stretched. The oblique muscle twists the arm. The octopus anatomy has muscle fibers aligning along a spiral in one direction. The simulated octopus can swim by orchestrating 24 muscles harmoniously and steer the moving direction rapidly and fluidly. Figure 9 shows the muscle excitation patterns when the octopus makes a rapid turn.

The octopus often hides its body in a safe zone such as an abandoned bottle or a narrow valley. We consciously create such an environment (see Figure 7). Collision detection and response is a key technical component that simulates the soft-bodied animal in a cramped environment. Simply pushing the nodal positions that interpenetrate obstacles to collision-free space can avoid the collision. At every iteration of the dynamics simulation, we check all nodal positions whether they are colliding using a Triangle-Triangle intersection algorithm and compute their collision-free space projection. All interpenetrating nodes are projected into the collision-free space after solving the global step.

The *Bottle Escape* scenario starts with the octopus in a bottle. We begin with a swimming policy network learned in free space as an initial guess. The target velocity is set to aim at the entrance of the bottle. Interestingly, the octopus learned to squeeze out of the bottle in only a few iterations. We suspect that the free-space swimming policy is already good enough to push the body forward even when the body is jammed in the narrow passage. In the *Narrow Valley* example, we planed the trajectory using Bézier splines. The subject computes the target velocity at every time step to track the plan by computing the closest points on the spline. This example does not require re-training of the control policy since the free-space swimming policy manages to navigate through narrow passages while coping with the curved trajectory and repeated collisions.

The *stingray* has a thin, wide body, which is modeled as a thin plate of deformable tissues. The bending-type muscles embedded in the thin plate has a zigzag shape along the side fins imitating the

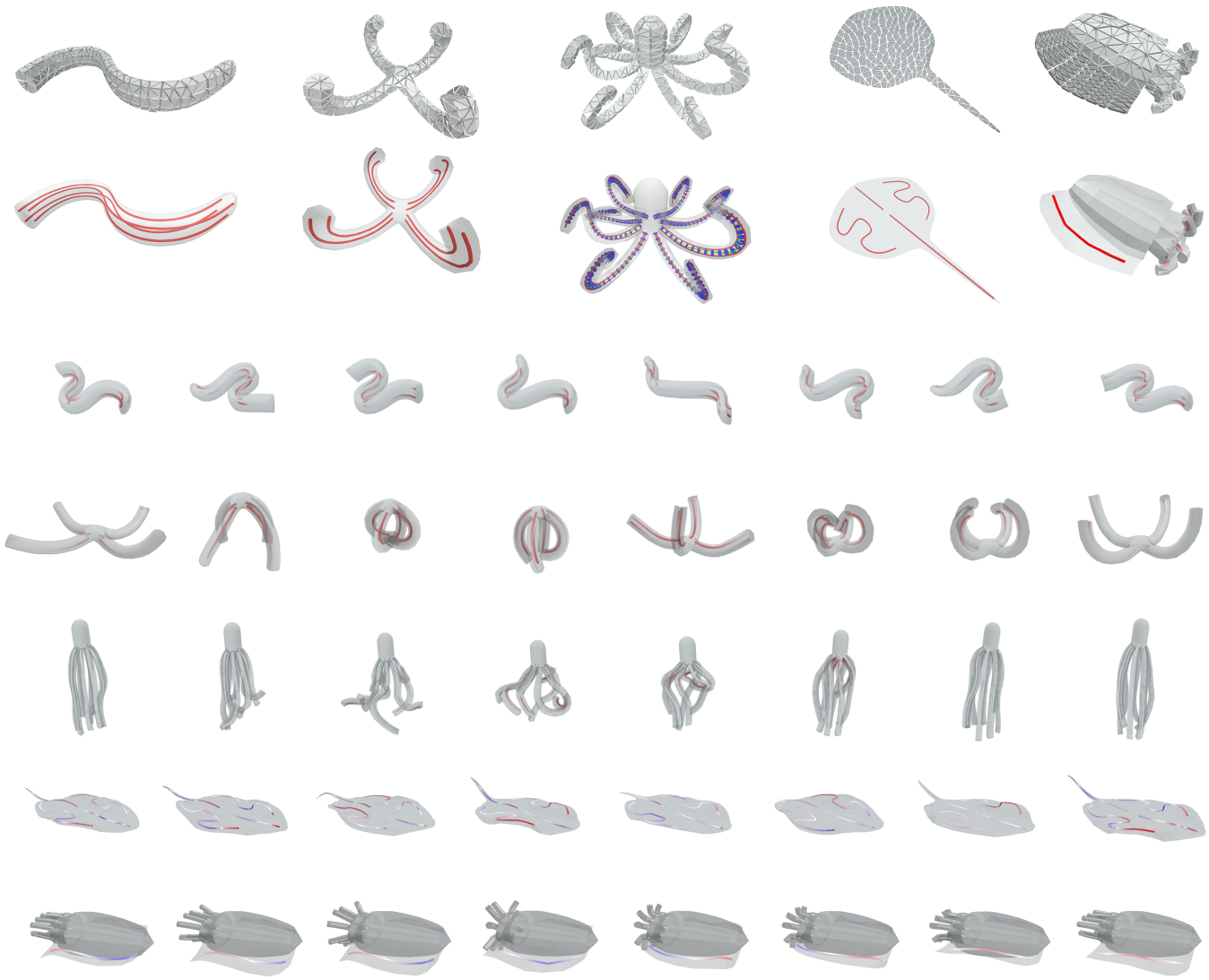


Fig. 6. The FEM meshes and nerve cords of assorted soft-bodied animals. (From left to right) Lamprey, starfish, octopus, stingray and cuttlefish.

Fig. 7. The animals in cramped environments. (Left to Right) The octopus escaping the bottle, the octopus swimming through a narrow valley, and the ribbon eels passing through obstacles.

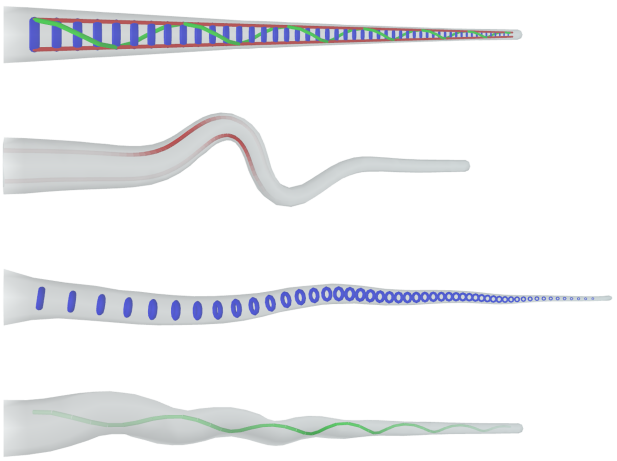


Fig. 8. Octopus arms. The contraction of longitudinal, transverse and oblique muscles respectively bend, lengthen and twist the arm.

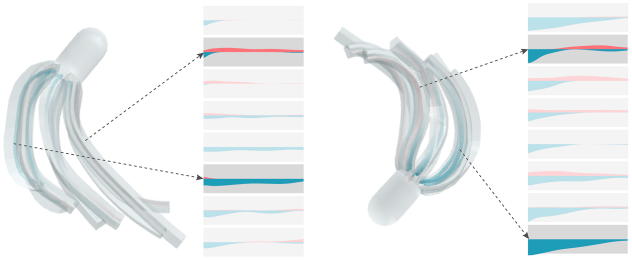


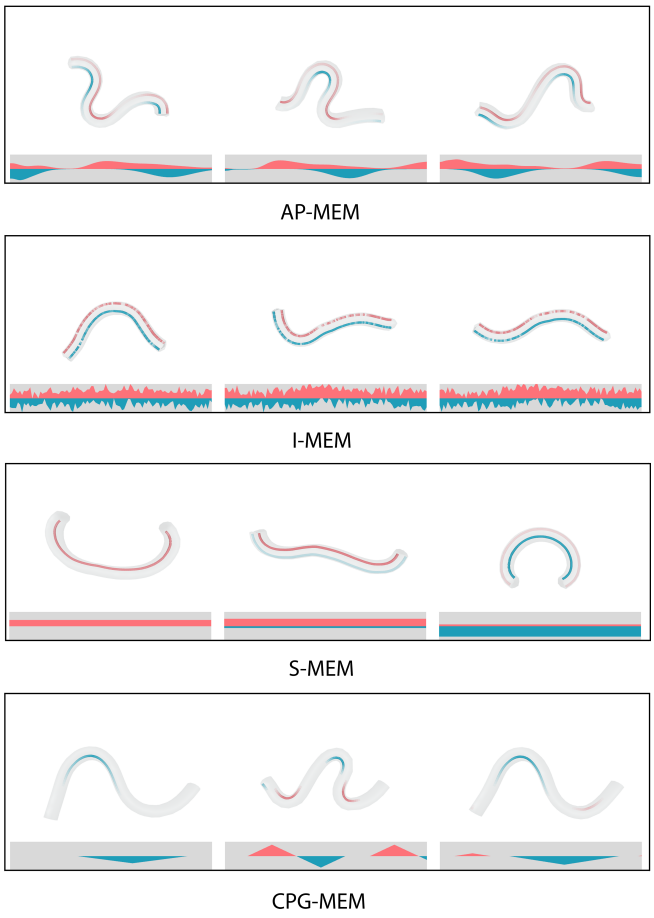
Fig. 9. The octopus turning. The longitudinal muscle inside the arm shown in green plays a role in making forward thrusts, while another longitudinal muscle outside the arm shown in pink plays an opposite role in recovering the arm position for the next thrust and making backward thrusts to brake. The octopus makes strong thrusts with the arms outside of turning and brakes with the arms inside to change its moving direction rapidly.

anatomy of stingrays [Park et al. 2016]. Propagating muscle excitation signals through the zigzag muscles produce wavy, fluttering movements of soft fins. We also created a variety of imaginary, thin-shell animals actuated by various muscles and their embeddings.

The *Cuttlefish* has a volumetric body with ten short arms and thin fins on both sides of the body. It has two longitudinal contractile muscles in each arm and a bending muscle in each fin. It can move forward slowly by fluttering the fins. The arms can make additional thrusts to move faster.

5.2 Comparison of Muscle Excitation Models

We compared the effectiveness of our AP-MEM with alternative models including I-MEM, S-MEM and CPG-MEM (See figure 10). The free-space swimming policies for the lamprey, the starfish and the octopus are learned with different MEMs and we compared their average returns and learning time. The I-MEM policies achieved high average returns for simple animals (the lamprey and the starfish),



	AP-MEM	I-MEM	S-MEM	CPG-MEM
Lamprey	798.98	3655.67	937.25	29.39
(time)	19	350	45	47.85
Starfish	1684.5	1809.34	146.43	40.22
(time)	6	202	22	47.85
Octopus	1503.55	8.56	15.05	N/A
(time)	47	156	57	N/A

Fig. 10. Comparison of MEMs. The average return of the free-space swimming task and the learning time (hours) are compared.

implying that the I-MEM policies perform well in respect to achieving a task-driven goal. However, the I-MEM policies produce noisy, unstructured movements that do not appear biological. The average return is probably good at measuring the performance of a policy, but certainly poor at evaluating the quality of movements. Learning an I-MEM policy is slow because of its excessive degrees of freedom in control. The CPG-MEM policies produce more structured, better-looking movements than the I-MEM policies. Even with fewer degrees of freedom, learning a CPG-MEM policy for the octopus is much slower than learning an I-MEM policy. The

CPG-MEM policy for the octopus is not robust even after two weeks of learning in our experiments. The computational cost for policy learning is not proportional to the degrees of freedom, the number of FEM elements, and the number of muscle actuators. There are qualitative factors, such as the structural complexity of the body and muscles, that affect more than the numbers. Our AP-MEM policies out-perform all alternatives by a large margin in regards to both computational efficiency and the quality of movements. The advantages of AP-MEM is more prominent when the model is more complex.

5.3 Interactive User Interface

We implemented a user interface system that allows the user to design his/her own creatures easily (see Figure 11). Our user interface system currently supports the creation of an animal with its thin-plate body. The interactive design of volumetric bodies is a subject for future research.

The user interface system has two stages: modeling stage and simulation stage. In the modeling stage, the user can draw the contour of the animal and the routes of muscles. The system generates a triangle FEM mesh via Delaunay triangulization of the contour and embeds the nerve cords into the triangle mesh by computing the intersection between nerve cords and triangles. In the simulation stage, the user specifies the initial values of central nerve signals $(\theta_1, \theta_2, \theta_3)$ and propagation parameters $(\alpha_0, \beta_0, \kappa_0)$ for each muscle, while visualizing the body deformation in simulation. At this stage, we disable hydrodynamic forces to focus on visualizing the effect of central nerve signals and their propagation. Once the parameter setting is finished, the new animal model goes through the learning phase to learn its control policy under the influence of hydrodynamics.

6 DISCUSSION

We presented a framework for modeling, simulation, and control of soft-bodied animals with biomimetic actuators. The key to the success of our approach is our muscle excitation model that makes a good balance between simplicity vs. generalization capability, structure vs. flexibility, and low control dimensionality vs. high modeling dimensionality. The DRL algorithm equipped with our AP-MEM can cope with the complexity of deformable soft bodies, a continuum of muscle fibers, and impressive numbers of DoFs and discretized muscle actuators. The increased dimensionality and model complexity have been successfully translated into smooth, flexible movements that appear biological. We envision the framework in which accurate biomechanical models may be applied to truly accurate motions for animals alive, extinct, or imaginary.

The interactive performance of our system is largely attributed to projective dynamics and deep reinforcement learning. The runtime simulation on a typical desktop PC is 3 to 5 times slower than real-time if self-collision is not considered in the simulation. Collision detection and response are notorious computational bottlenecks of deformable body simulation. Although our method is much faster than previous systems for soft-bodied animal simulation, there is room for performance improvements. One possibility is the implicit handling of hydrodynamic forces, which will lead to a full implicit integration method that allows for larger time steps and unconditional stability.

Even though successful applications of AP-MEMs have been demonstrated so far, our framework also has numerous limitations. Underwater soft-bodied animals in nature have an alternative mechanism to move, which we are currently unable to reproduce with the simplified hydrodynamics model. For example, the octopus draws water into its body cavity and spurts the water out to generate thrust. A more sophisticated hydrodynamics model, which can simulate the incompressibility of fluids, is crucial to reproduce such behavior. Our animal models lack a lot of important anatomical features. The octopus has membranes between legs to push the water off and uses suckers for grabbing and holding prey. We have not implemented such features in our simulation model yet.

The design of body shapes and the embedding of nerve cords have significant impacts on the motor ability of the learned control policy. We designed our animal models based on similar creatures in nature and we often found that small tweaking of the body shapes and muscle embedding substantially affect the level of motor ability and the type of motor skills the control policy can achieve. An intriguing direction for future research is to learn the most efficient body shape, muscle embedding, and motion for an animal simultaneously [Geijtenbeek et al. 2013; Ha et al. 2017].

ACKNOWLEDGMENTS

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the SW Starlab support program(IITP-2017-0-00878) supervised by the IITP(Institute for Information & communications Technology Promotion).

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. 43–54.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable Object Animation Using Reduced Optimal Control. *ACM Trans. Graph.* 28, 3, Article 53 (2009).

Jernej Barbič and Jovan Popović. 2008. Real-time Control of Physically Based Simulations Using Gentle Forces. *ACM Trans. Graph.* 27, 5, Article 163 (2008).

James M. Bern, Kai-Hung Chang, and Stelian Coros. 2017. Interactive Design of Animated Plushies. *ACM Trans. Graph.* 36, 4, Article 80 (2017).

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 32, 6, Article 154 (2014).

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced Projective Dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (2018).

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Trans. Graph.* 29, 4, Article 130 (2010).

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. 2011. Locomotion Skills for Simulated Quadrupeds. *ACM Trans. Graph.* 30, 4, Article 59 (2011).

Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable Objects Alive! *ACM Trans. Graph.* 31, 4, Article 69 (2012).

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Interactive Simulation of Stylized Human Locomotion. *ACM Trans. Graph.* 27, 3, Article 82 (2008).

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. OpenAI Baselines. https://github.com/openai/baselines. (2017).

Ye Fan, Joshua Litven, and Dinesh K. Pai. 2014. Active Volumetric Musculoskeletal Systems. *ACM Trans. Graph.* 33, 4, Article 152 (2014).

Jingyi Fang, Chenfanfu Jiang, and Demetri Terzopoulos. 2013. Modeling and Animating Myriapoda: A Real-time Kinematic/Dynamic Approach. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. 203–212.

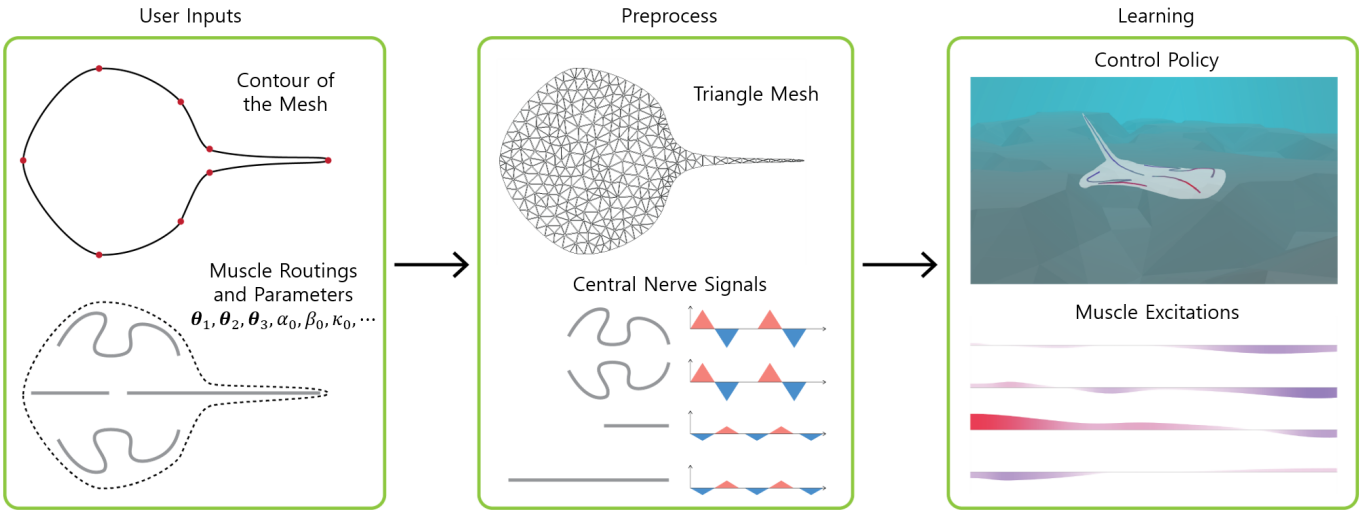


Fig. 11. The pipeline of animal model design.

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible Muscle-based Locomotion for Bipedal Creatures. *ACM Trans. Graph.* 32, 6, Article 206 (2013).

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. 1998. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-based Models. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. 9–20.

Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. 2017. Joint Optimization of Robot Design and Motion Parameters using the Implicit Function Theorem. In *Robotics: Science and Systems*.

Binyamin Hochner. 2012. An Embodied View of Octopus Neurobiology. *Current Biology* 22, 20 (2012), R887 – R892.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. 71–78.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (2018).

Takashi Ijiri, Kenshi Takayama, Hideo Yokota, and Takeo Igarashi. 2009. ProcDef: Local-to-global Deformation for Skeleton-free Character Animation. *Computer Graphics Forum (proceedings of Pacific Graphics)* 28, 7 (2009), 1821–1828.

Eunjung Ju, Jungdam Won, Jehee Lee, Byungkuk Choi, Junyong Noh, and Min Gyu Choi. 2013. Data-driven Control of Flapping Flight. *ACM Trans. Graph.* 32, 5, Article 151 (2013).

Junggon Kim and Nancy S. Pollard. 2011. Fast Simulation of Skeleton-driven Deformable Body Characters. *ACM Trans. Graph.* 30, 5, Article 121 (2011).

Cecilia Laschi, Matteo Cianchetti, Barbara Mazzolai, Laura Margheri, Maurizio Follador, and Paolo Dario. 2012. Soft Robot Arm Inspired by the Octopus. *Advanced Robotics* 26, 7 (2012), 709–727.

Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. 1996. Limit Cycle Control and Its Application to the Animation of Balancing and Walking. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. 155–162.

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018. Dexterous Manipulation and Control with Volumetric Muscles. *ACM Trans. Graph.* 37, 4, Article 57 (2018).

Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion Control for Many-muscle Humanoids. *ACM Trans. Graph.* 33, 6, Article 218 (2014).

Guy Levy and Binyamin Hochner. 2017. Embodied organization of Octopus vulgaris morphology, vision, and locomotion. *Frontiers in physiology* 8 (2017), 164.

Libin Liu and Jessica Hodgins. 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 142 (2018).

Libin Liu, Michiel Van De Panne, and Kangkang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Trans. Graph.* 35, 3, Article 29 (2016).

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-spring Systems. *ACM Trans. Graph.*, Article 214 (2013).

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based Elastic Materials. *ACM Trans. Graph.* 30, 4, Article 72 (2011).

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position Based Dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836.

Zherong Pan and Dinesh Manocha. 2018. Active Animations of Reduced Deformable Models with Environment Interactions. *ACM Trans. Graph.* 37, 3, Article 36 (2018).

Sung-Jin Park, Mattia Gazzola, Kyung Soo Park, Shirley Park, Valentina Di Santo, Erin L. Blevins, Johan U. Lind, Patrick H. Campbell, Stephanie Dauth, Andrew K. Capulli, Francesco S. Pasqualini, Seungkuk Ahn, Alexander Cho, Hongyan Yuan, Ben M. Maoz, Ragu Vijaykumar, Jeong-Woo Choi, Karl Deisseroth, George V. Lauder, L. Mahadevan, and Kevin Kit Parker. 2016. Phototactic guidance of a tissue-engineered soft-robotic ray. *Science* 353, 6295 (2016), 158–162.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (2018).

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2016. Terrain-adaptive Locomotion Skills Using Deep Reinforcement Learning. *ACM Trans. Graph.* 35, 4, Article 81 (2016).

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (2017).

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Trans. Graph.* 37, 6, Article 178 (2018).

Jonas N Richter, Binyamin Hochner, and Michael J Kuba. 2015. Octopus arm movements under constrained conditions: adaptation, modification and plasticity of motor primitives. *Journal of Experimental Biology* 218, 7 (2015), 1069–1076.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Christian Schulz, Christoph von Tycowicz, Hans-Peter Seidel, and Klaus Hildebrandt. 2014. Animating Deformable Objects Using Sparse Spacetime Constraints. *ACM Trans. Graph.* 33, 4, Article 109 (2014).

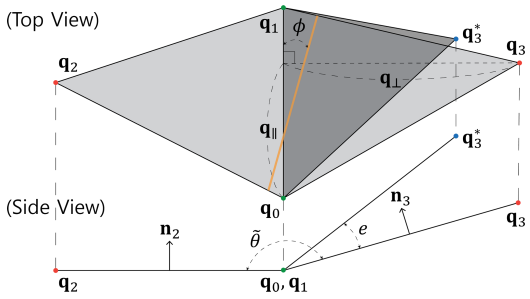
Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic Biomechanical Simulation and Control of Human Swimming. *ACM Trans. Graph.* 34, 1, Article 10 (2014).

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*. Article 20.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (2018).

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph.* 26, 3, Article 107 (2007).

Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu. 2011. Articulated swimming creatures. *ACM Trans. Graph.* 30, 4, Article 58 (2011).



Jie Tan, Greg Turk, and C. Karen Liu. 2012. Soft Body Locomotion. *ACM Trans. Graph.* 28, 3, Article 26 (2012).

TensorFlow. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

Juan Tian and Qiang Lu. 2015. Simulation of Octopus Arm Based on Coupled CPGs. *J. Robot.* 2015, Article 4 (2015).

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 (SIGGRAPH '94)*. 43–50.

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (2012).

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to Train Your Dragon: Example-guided Control of Flapping Flight. *ACM Trans. Graph.* 36, 6, Article 198 (2017).

Jungdam Won, Jungnam Park, and Jehee Lee. 2018. Aerobatics Control of Flying Creatures via Self-regulated Learning. *ACM Trans. Graph.* 37, 6, Article 181 (2018).

Jia-chi Wu and Zoran Popović. 2003. Realistic modeling of bird flight animations. *ACM Trans. Graph.* 22, 3 (2003), 888–895.

Yuting Ye and C. Karen Liu. 2010. Optimal Feedback Control for Character Animation Using an Abstract Model. *ACM Trans. Graph.* 29, 4, Article 74 (2010).

Yoram Yekutieli, German Sumbre, Tamar Flash, and Binyamin Hochner. 2003. How to move with no rigid skeleton? The octopus has the answers. 49 (2003), 250–4.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Trans. Graph.* 26, 3, Article 105 (2007).

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-energy Locomotion. *ACM Trans. Graph.* 37, 4, Article 144 (2018).

Appendix A BENDING FIBERS

The bending muscle bends the thin surface according to muscle activation e, which changes the rest shape $\tilde{\mathbf{X}}(e)$ in the equation (10). Given an edge $\mathbf{q}_0\mathbf{q}_1$ and its adjacent vertices $\mathbf{q}_2$ and $\mathbf{q}_3$, the dihedral angle $\tilde{\theta}$ of the rest shape is

$$\tilde{\theta} = \cos^{-1}(\mathbf{n}_2 \cdot \mathbf{n}_3) \tag{18}$$

where $\mathbf{n}_2 = \frac{(\mathbf{q}_2-\mathbf{q}_1)\times(\mathbf{q}_1-\mathbf{q}_0)}{\|(\mathbf{q}_2-\mathbf{q}_1)\times(\mathbf{q}_1-\mathbf{q}_0)\|}$ and $\mathbf{n}_3 = \frac{(\mathbf{q}_1-\mathbf{q}_3)\times(\mathbf{q}_1-\mathbf{q}_0)}{\|(\mathbf{q}_1-\mathbf{q}_3)\times(\mathbf{q}_1-\mathbf{q}_0)\|}$ are the normal vectors of $\triangle\mathbf{q}_0\mathbf{q}_1\mathbf{q}_2$ and $\triangle\mathbf{q}_0\mathbf{q}_3\mathbf{q}_1$, respectively. The new activated shape can be computed by rotating $\triangle\mathbf{q}_0\mathbf{q}_3\mathbf{q}_1$ about axis $\mathbf{q}_0\mathbf{q}_1$ by angle $(e\cos\phi)$. The coordinate of $\mathbf{q}_3^*$ in the new rest shape is

$$\mathbf{q}_3^*(e) = \mathbf{q}_0 + \mathbf{q}_\parallel + \mathbf{R}(\hat{\omega}, e\cos\phi)\mathbf{q}_\perp, \tag{19}$$

where

$$\begin{aligned} \mathbf{q}_\parallel &= \frac{(\mathbf{q}_3-\mathbf{q}_0)\cdot(\mathbf{q}_1-\mathbf{q}_0)}{(\mathbf{q}_1-\mathbf{q}_0)\cdot(\mathbf{q}_1-\mathbf{q}_0)}(\mathbf{q}_1-\mathbf{q}_0), \\ \mathbf{q}_\perp &= (\mathbf{q}_3-\mathbf{q}_0)-\mathbf{q}_\parallel, \\ \hat{\omega} &= \frac{(\mathbf{q}_1-\mathbf{q}_0)}{\|\mathbf{q}_1-\mathbf{q}_0\|}. \end{aligned} \tag{20}$$

ϕ is angle between the fiber direction and the edge and $\mathbf{R}(\hat{\omega}, e\cos\phi)$ is a rotation matrix. We would like to note that the change of the rest shape does not affect the projection metric $\mathbf{A}_i$ in the equation (4) since the cotangent weights $\mathbf{c}$ remain intact. Given the new rest shape, the projection $\mathbf{p}_i \in SO(3) = \mathbf{U}\mathbf{V}^\top$ can be computed through the singular value decomposition of one-rank matrix $(\mathbf{X}\mathbf{c})(\tilde{\mathbf{X}}(e)\mathbf{c})^\top = \mathbf{U}\Sigma\mathbf{V}^\top$.