Automated Routing of Muscle Fibers for Soft Robots

Guirec Maloisel[1,2], Espen Knoop[1,2], Christian Schumacher[1], Moritz Bächer[1]

***Abstract*—This paper introduces a computational approach for routing thin artificial muscle actuators through hyperelastic soft robots, in order to achieve a desired deformation behavior. Provided with a robot design, and a set of example deformations, we continuously co-optimize the routing of actuators, and their actuation, to approximate example deformations as closely as possible.**

We introduce a data-driven model for McKibben muscles, modeling their contraction behavior when embedded in a silicone elastomer matrix. To enable the automated routing, a differentiable hyperelastic material simulation is presented. Because standard finite elements are not differentiable at element boundaries, we implement a Moving Least Squares formulation, making the deformation gradient twice-differentiable.

Our robots are fabricated in a two-step molding process, with the complex mold design steps automated. While most soft robotic designs utilize bending, we study the use of our technique in approximating twisting deformations on a bar example. To demonstrate the efficacy of our technique in soft robotic design, we show a continuum robot, a tentacle, and a 4-legged walking robot.

***Index Terms*—Soft Robotics, Artificial Muscles, MLS-based Simulation, Equilibrium-Constrained Design Optimization**

I. Introduction

THE OCTOPUS tentacle and the elephant trunk are comprised of soft tissue only, and their deformation behavior is entirely governed by the actuation of complex internal muscle networks. The routing patterns of these muscles have evolved over millions of years, and enable extraordinarily dexterous movement, manipulation, and locomotion.

Taking this as inspiration, we here study the problem of designing and controlling muscle-like embedded actuation systems for soft-bodied robots, in particular the routing and the actuation of individual muscle fibers. We consider the general problem of placing soft one-dimensional actuators inside an *arbitrary* hyperelastic soft body, to obtain a desired deformation behavior.

Due to the non-linear coupling between the soft continuum body and the actuators, the complex interaction between control and routing parameters, and the large deformations, a manual trial-and-error exploration of the design space would rapidly become intractable for all but the simplest cases. A simulation-based design optimization approach is therefore required.

To this end, we formulate an optimization problem where we solve for the optimal routing and actuation parameters of artificial muscle actuators, such that we get as close as possible to user-specified target deformations.

We represent fiber-like actuators with cubic Hermite splines. Embedding them in a Finite Element (FE) representation of the soft body, we model their contraction behavior along these curves with a non-linear spring, estimating pressure-dependent spring parameters from characterization experiments. During optimizations, we then seek to find the optimal spline and actuation values, such that the robot design matches a set of targets in a least-squares sense. We explore several distance metrics to measure differences between simulated and target deformations, observing that the minimization of deformations along user-selected curves leads to desirable results.

Standard FE formulations interpolate FE degrees of freedom with Lagrange shape functions. While sufficient for simulation, the resulting deformation gradient is *discontinuous* at element boundaries. However, to freely move actuation fibers through elements using *continuous* Newton-type optimization, the deformation gradient has to be at least twice-differentiable. To make our simulations differentiable, we replace Lagrange with Moving Least Squares (MLS) shape functions. Constraining simulated deformations to be in equilibrium when comparing them to targets, we then co-optimize the routing of actuators, and actuation variables.

To showcase our formulation, we use thin McKibben artificial muscle actuators and route them through a soft silicone body. These actuators are fully soft and compliant, they are readily available and easily controllable, and they lend themselves well towards integration into robots of arbitrary shape as they do not impose constraints on the geometry of the robot. We would expect our approach and model to also be applicable to other embedded fiber-like actuators such as shape memory alloys or twisted Nylon actuators [1].

We demonstrate the efficacy of our method on a soft continuum robot, undergoing a complex spatial deformation, a soft robotic octopus tentacle, and a four-legged robot which twists and bends to locomote.

Succinctly, our core contributions are:

- A technique for co-optimizing the design and control of fiber-like actuators embedded in hyperelastic soft bodies, targeting the approximation of a desired deformation behavior spanned by user-provided samples.
- A differentiable simulation with a MLS-based deformation field, enabling the continuous optimization of fiber-like entities in soft continua.
- A data-driven model for McKibben actuators embedded in silicone, well-suited for simulation and optimization.

This work was funded by Disney Research.

[1] All authors are with Disney Research, Stampfenbachstrasse 48, 8006 Zurich, Switzerland. [2] The first two authors contributed equally. *first.lastname*@disneyresearch.com

II. Related Work

Early work in computational design of actuated soft robots includes research by Hiller et al. [2], where a complete automated pipeline for designing and fabricating soft robots is presented. However, their method is specific to actuation by volume-changing voxels and is not suited for more common actuation mechanisms.

More recently, Connolly et al. [3, 4] considered the simulation-driven design of fiber-wrapped finger-like actuators. However, their technique is tailored to fiber-wrapped actuators and therefore not readily transferrable to robots of arbitrary geometry. Similarly, Deimel et al. [5] considered the simulation-aided optimization of a soft robot hand for a grasping task, varying the actuator design parameters. Other computational methods have also been developed which are specific to finger-like actuators [6, 7, 8]. In this work, we are interested in the more general problem of actuating a soft-bodied robot of arbitrary shape with fiber-like actuators.

The actuation of arbitrary shapes has also received some attention. Ma et al. [9] developed a method for designing pneumatic chambers in soft-bodied objects so that they deform in a specified manner, but their approach is tailored to soft-material 3D printing. Bern et al. [10] considered the design of cable networks for actuating soft plush robots.

A central aspect of our work is that the artificial muscle routing is posed as a continuous optimization problem, where the muscle is free to move smoothly through the underlying simulation mesh. This is in contrast to approaches where only discrete and discontinuous sets of routings are considered [10, 11].

Design using Differentiable Simulation: With regards to simulation-based design, Morzadec et al. [12] considered the shape optimization of a soft robotic leg, driven by conventional (rigid) servomotors. Related design problems were tackled in computer graphics, including the design of cable-driven deformable [13] and compliant [14, 15] structures, rubber balloons [16] and inflatable structures [17], and compliant mechanisms [18]. Contrary to this approaches, we describe a differentiable simulation that enables the continuous optimization of fiber-like entities embedded in soft bodies.

Our approach is also similar in nature to [19] and [11]: although they considered the design of sensing networks rather than actuation networks, the underlying principles are similar. In [11], a sub-selection strategy was considered, where a subset of fiber-like sensors was chosen from a larger, redundant set. However, the routing of sensing elements was not refined. In [19], the routing of sensors was optimized. However, they restricted refinements to planes, and smoothed the deformation field, introducing significant error.

Use of MLS in Simulation: Instead of standard Lagrange shape functions, we rely on MLS shape functions in our simulation. Various MLS-based mesh-free simulation techniques have been proposed, and we point the reader to the survey by Fries and Matthies [20] for a detailed review. Like Breitkopf et al. [21] and Martin et al. [22], we rely on an explicit formulation of MLS shape functions, and like [23, 24], we use linear and quadratic moving least squares. However, in contrast to previous work, we discuss the use of MLS in the context of equilibrium-constrained optimization, providing sufficient smoothness to enable the continuous optimization of the routing of fiber-like entities embedded in soft robots.

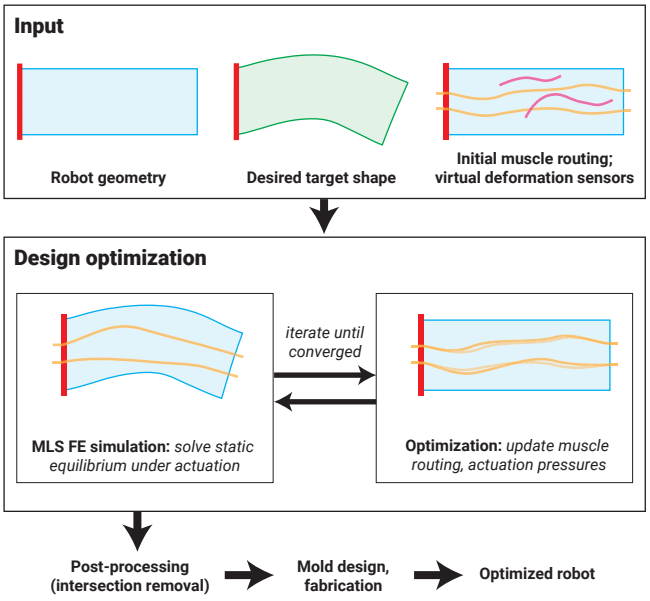


Fig. 1. **Pipeline Overview** As *input*, the user provides a soft robot design, a set of desired target deformations, initial routings for the artificial muscle fibers (yellow), and a set of "tracking curves" (pink) which specify the locations on the robot where the target-matching error should be minimized. In the *design optimization* stage, a Finite Element simulation predicts the robot deformation for a given muscle routing and actuation pressures, and this is used in conjunction with a 2nd order optimization scheme that updates the muscle routings and actuation pressures to improve target-matching performance. By using a Moving Least Squares (MLS) Finite Element model, the muscle fibers can move smoothly through the underlying simulation mesh. This is iterated until convergence. After an optional post-processing step of removing collisions by slightly perturbing colliding fibers, molds are designed, and the robot is fabricated using silicone injection molding.

Artificial Muscle Modeling: While analytical artificial muscle models exist [25, 26], they idealize the behavior of these actuators, and cannot directly be used for optimization. Phenomena such as friction between fibers, or the resistance to stretching beyond the intended maximum length of an actuator, are difficult to model analytically. Most importantly, previous models do not account for the embedding of muscles in a soft elastomer matrix. We instead propose a data-driven model that explains measured data, and is well-suited for automated routing using optimization.

III. Overview

An overview of the steps in our pipeline is presented in Fig. 1, with a simplified example of a bar. As input, the user specifies an initial robot design, a set of target shapes to match, initial routings for the artificial muscle fibers, and a set of "tracking curves" that specify the locations on the robot where the target-matching error should be minimized. The core of our pipeline is an automated design optimization, which combines a differentiable FE simulation of the soft robot with embedded muscles, together with a 2nd order optimization scheme that updates the muscle routings and actuation pressures in order to

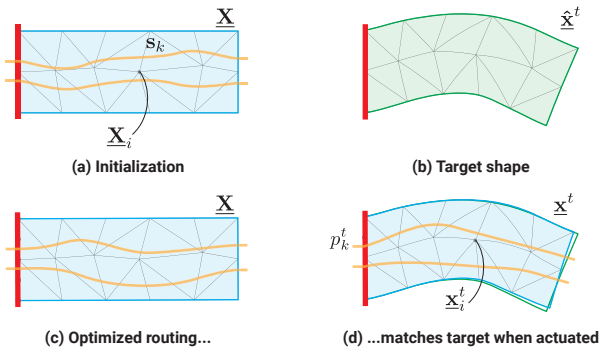


Fig. 2. **Design Optimization** We represent the individual artificial muscle fibers in a robot design at rest with spline curves $\mathbf{s}_k$, and ask the user to specify their initial routing with a spline tool (a), together with a set of target shapes t (b). We then co-optimize their routing (c, yellow) and actuation pressures p_k to minimize differences between simulated and desired target shapes (d). We use n FE degrees of freedom $\mathbf{X}_i$ and $\mathbf{x}_i$, stacked in $3n$-vectors $\underline{\mathbf{X}}$ and $\underline{\mathbf{x}}$, to represent the undeformed and deformed configuration of the robot. Because we optimize a different set of pressure values per target, we add superscripts t to the quantities describing the deformed configuration.

improve target matching performance. After an optional post-processing step to remove collisions between muscle fibers, the optimized robot is fabricated using a two-step silicone injection molding process.

In the design optimization, illustrated in Fig. 2, we co-optimize the muscle fiber routing and actuation to approximate the user-specified shapes as closely as possible. The initial routing defines the in- and outlets of the actuators, whose positions we keep fixed throughout optimizations.

We represent an unpressurized artificial muscle k, embedded in a soft robot design, with spline parameters $\mathbf{s}_k$. When pressurized to pressure p_k, the robot deforms.

As strain fields are difficult to predict, the routing of actuators, and their optimal actuation, is a highly non-trivial and counterintuitive task, and a manual routing leads to suboptimal results at best. To navigate this complex design space, we propose to utilize computation.

To pose our optimization problem, we collect the individual spline control points in a larger column vector $\mathbf{s}$, and corresponding pressure values in a vector $\mathbf{p}$. Our ultimate goal is then to find an optimal routing of the actuators, approximating a set of target shapes t as closely as possible, with a set of co-optimized pressure values $\mathbf{p}^t$ that vary per target.

A. Simulating Artificial Muscle Actuation

To measure differences between current and desired deformations, we simulate the response of an actuated robot using finite elements e. For FE simulation, we use an energy-based formulation [27] where we integrate the strain energy density Ψ of a standard hyperelastic model, over the rest volume V_e of each element

$$E_{\text{elast}}(\underline{\mathbf{x}}) = \sum_e \int_{V_e} \Psi(\mathbf{F}(\underline{\mathbf{X}}, \underline{\mathbf{x}}, \mathbf{X}))\, \mathrm{d}V. \tag{1}$$

We evaluate the strain energy density at points $\mathbf{X} \in \mathbb{R}^3$ in the rest volume by computing the deformation gradient $\mathbf{F}$ at

TABLE I
NOTATION: UNDEFORMED AND DEFORMED CONFIGURATIONS, AND TARGETS.

	dim.	undef.	def.	def. (tar.)	tar.
point in domain	$\mathbb{R}^3$	$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$	$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\mathbf{x}^t$	-
FE DoF	$\mathbb{R}^3$	$\underline{\mathbf{X}}_i = \begin{bmatrix} \underline{X}_i \\ \underline{Y}_i \\ \underline{Z}_i \end{bmatrix}$	$\underline{\mathbf{x}}_i = \begin{bmatrix} \underline{x}_i \\ \underline{y}_i \\ \underline{z}_i \end{bmatrix}$	$\underline{\mathbf{x}}_i^t$	-
FE DoFs	$\mathbb{R}^{3n}$	$\underline{\mathbf{X}}$	$\underline{\mathbf{x}}$	$\underline{\mathbf{x}}^t$	$\hat{\underline{\mathbf{x}}}^t$

dim.: dimensions; *undef.*: undeformed configuration; *def.*: deformed configuration; *def. (tar.)*: deformed configuration for target t; *tar.*: user-specified target deformation; *FE DoF*: single finite element degree of freedom; *FE DoFs*: all finite element degrees of freedom stacked in a single vector.

$\mathbf{X}$ that depends on the n undeformed and deformed three-dimensional FE degrees of freedom, $\underline{\mathbf{X}}_i$ and $\underline{\mathbf{x}}_i$, collected in $3n$-vectors $\underline{\mathbf{X}}$ and $\underline{\mathbf{x}}$ (compare with Fig. 2 and Tab. I).

To model the energy induced by the actuators, we propose a data-driven artificial muscle model that acts like a non-linear spring, contracting the fiber along the spline curves $\mathcal{C}$ when pressurized. To this end, we formulate a custom strain energy density, and make this energy density, as well as the strain ε along the spline curve, dependent on the pressure

$$E_{\text{act}}(\mathbf{s}, \mathbf{p}, \underline{\mathbf{x}}) = \sum_k \int_{\mathcal{C}(\mathbf{s}_k)} \Psi_{\text{act}}(p_k, \varepsilon(p_k, \underline{\mathbf{X}}, \underline{\mathbf{x}}, \mathbf{X}))\, \mathrm{d}\mathcal{C}, \tag{2}$$

with the differential $\mathrm{d}\mathcal{C}$ set to an infinitesimal circle-swept curve element, and hence a volume. Note that this formulation has a resemblance to common rod models. We will discuss our custom muscle fiber model in Sec. VI.

While we describe a specific model for McKibben-type muscles, we would like to stress that our simulation and optimization would interface with other fiber-like actuators.

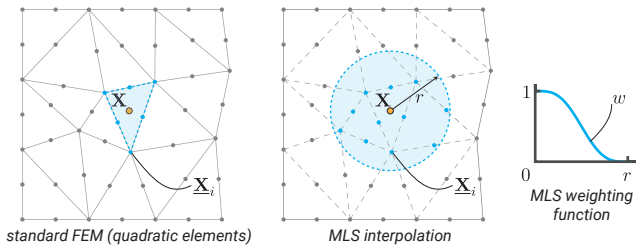
To simulate the actuated robot , i.e. to compute the deformed configuration $\underline{\mathbf{x}}(\mathbf{s}, \mathbf{p})$, we minimize the total potential energy $E = E_{\text{elast}} + E_{\text{act}}$ to first-order optimality

$$\mathbf{f}(\mathbf{s}, \mathbf{p}, \underline{\mathbf{x}}) = \frac{\partial E_{\text{elast}}}{\partial \underline{\mathbf{x}}} + \frac{\partial E_{\text{act}}}{\partial \underline{\mathbf{x}}} = \mathbf{0}, \tag{3}$$

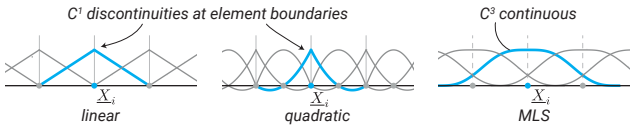
where the elastic and actuation forces are in equilibrium. For robots made of a softer silicone, gravity has a non-negligible effect, and we formulate an additional energy that accounts for the work done by the gravitational force.

Note that this equilibrium changes whenever we move embedded fibers, or pressurize them with a different set of pressures. Hence, to compare simulated to user-specified targets during optimizations, it is important to enforce this equilibrium condition as a constraint.

When optimizing a robot design, we seek to move muscle fibers across finite element boundaries. Because standard Lagrange shape functions lead to a deformation field that is continuous but *not* differentiable at these boundaries, we replace this basis with sufficiently smooth Moving Least Squares (MLS) shape functions. We discuss the differentiability of our simulation in Sec. IV.



(a) 2D comparison of standard FEM and MLS interpolation



(b) 1D illustrations of shape functions

Fig. 3. **Smoothness of Deformation Field.** If we interpolate deformed degrees of freedom of elements with standard Lagrange shape functions (a, standard FEM), the deformation gradient field is discontinuous at element boundaries independent of the order of the shape functions as illustrated in 1D (b, linear quadratic). To enable the automated routing of fibers across boundaries, we rely on MLS shape functions (b, MLS), weighing the influence of nodal degrees of freedom (a, MLS interpolation) with a thrice-differentiable weighting function (a, MLS weighting function).

B. Automated Routing of Actuators

To co-optimize the routing and actuation of muscle fibers, we minimize the distance between simulated deformations, $\underline{\mathbf{x}}^t \in \mathbb{R}^{3n}$, and user-specified target deformations $\hat{\underline{\mathbf{x}}}^t \in \mathbb{R}^{3n}$. Because user-specified targets result in strain distributions that are, in general, difficult to achieve with embedded muscles, we cannot hope to match targets exactly. In Sec. V, we will introduce a distance metric f_{dist} that relies on user-specified tracking curves to achieve a desired target matching quality.

In summary, we solve the problem

$$\min_{\mathbf{s},\mathbf{p}} \sum_t f_{\text{dist}}(\underline{\mathbf{x}}^t(\mathbf{s},\mathbf{p}^t), \hat{\underline{\mathbf{x}}}^t) + \mathcal{R}(\mathbf{s}) \tag{4}$$
$$\text{s.t.} \quad \mathbf{f}(\mathbf{s},\mathbf{p}^t,\underline{\mathbf{x}}^t(\mathbf{s},\mathbf{p}^t)) = \mathbf{0}, \quad \mathbf{p}_{\text{lo}} \le \mathbf{p}^t \le \mathbf{p}_{\text{up}}, \quad \forall t,$$

bounding the allowable pressure values from above and below. The term $\mathcal{R}(\mathbf{s})$ penalizes high-curvature in routed fibers, and prevents them from moving outside of the design.

IV. Differentiable Simulation

To automate the routing of muscle fibers through a discretized deformable solid, the deformation field has to be at least thrice-differentiable to enable numerical optimization using a quasi-Newton method with an approximate Hessian [28]. Note that a twice-differentiable deformation field is insufficient, because we require a twice-differentiable *deformation gradient* for optimization with 2nd-order techniques.

In standard finite element simulations (Fig. 3 a, standard FEM), the deformed configuration is interpolated *separately* on each element, and hence the deformation field is only C^0-continuous at element boundaries, independent of the order of the shape functions used (b, linear, quadratic). If we were only interested in simulating soft robots with embedded artificial muscles, standard finite elements would suffice. However, for optimization, the deformation field has to be sufficiently smooth not only within, but also across, elements, and a smooth transition at element boundaries is only feasible if degrees of freedom of *neighboring* elements are taken into account (b, MLS).

The mapping of points $\mathbf{X}$ in the undeformed domain to points $\mathbf{x}$ in the deformed domain is commonly defined by interpolating the deformed nodes $\underline{\mathbf{x}}_i$ with standard Lagrange shape functions N_i

$$\mathbf{x}(\underline{\mathbf{X}},\underline{\mathbf{x}},\mathbf{X}) = \sum_i \underline{\mathbf{x}}_i N_i(\mathbf{X}) \quad \text{with} \quad \underline{\mathbf{x}}_i = \begin{bmatrix} \underline{x}_i \\ \underline{y}_i \\ \underline{z}_i \end{bmatrix}. \tag{5}$$

We instead rely on an approximating scheme with MLS shape functions that are common in meshless techniques [20]. Unlike for other schemes, the weighting function (Fig. 3 a, MLS weighting function) provides us with a mechanism to achieve a desired smoothness (b, MLS) while restricting the influence of each nodal degree of freedom to a local neighborhood (*local support property*). The latter is important for keeping the tangent stiffness matrix sparse, and simulations and analytical gradient computations efficient.

A. Constructing Thrice-Differentiable MLS Shape Functions

To construct Lagrange shape functions, we assume a polynomial basis $\mathbf{b}(\mathbf{X})$ (e.g., a linear basis $[1, X, Y, Z]^T$), and seek coefficients $\mathbf{c}_i$ such that shape function $N_i(\mathbf{X}) = \mathbf{b}(\mathbf{X})^T\mathbf{c}_i$ fulfills the Kronecker delta property $N_i(\underline{\mathbf{X}}_j) = \delta_{ij}$. This construction guarantees that nodal degrees of freedom are *interpolated*.

For an MLS-basis, a polynomial approximation is computed for *every* evaluation point $\mathbf{X}$, hence coefficients depend on $\mathbf{X}$. For example, to compute the x-component of point $\mathbf{x}$ in the deformed domain, the coefficients are the solution to the weighted least squares problem

$$\mathbf{c}(\mathbf{X}) = \arg\min_{\mathbf{c}} \sum_i w_i \left\| \mathbf{b}(\underline{\mathbf{X}}_i)^T\mathbf{c} - \underline{x}_i \right\|^2, \tag{6}$$

with weights $w_i = w(\|\mathbf{X} - \underline{\mathbf{X}}_i\|)$ decaying the further away the evaluation point is from an undeformed node $\underline{\mathbf{X}}_i$ (Fig. 3 b). Conveniently, the closed-form solution of this problem can readily be derived by setting the gradient of the least squares objective to zero

$$\mathbf{c}(\mathbf{X}) = \mathbf{G}^{-1}(\mathbf{X}) \sum_i w_i \mathbf{b}(\mathbf{X}) \underline{x}_i \quad \mathbf{G}(\mathbf{X}) = \sum_i w_i \mathbf{b}_i \mathbf{b}_i^T$$

where $\mathbf{b}_i$ is the basis evaluated at $\underline{\mathbf{X}}_i$. Hence, we compute the x-coordinate of the deformed configuration according to

$$x(\mathbf{X}) = \mathbf{b}(\mathbf{X})^T\mathbf{c}(\mathbf{X}) = \sum_i \underline{x}_i \underbrace{w_i \mathbf{b}(\mathbf{X})^T \mathbf{G}^{-1}(\mathbf{X}) \mathbf{b}_i}_{N_i(\mathbf{X})}, \tag{7}$$

defining the MLS-shape functions $N_i(\mathbf{X})$. As long as a sufficient number of degrees of freedom is within the finite support of the weighting function (compare with Fig. 3 a, MLS interpolation), matrix $\mathbf{G}$ is invertible, and shape functions are well-defined. Note that the shape functions do not depend on the deformed configuration, so they can be precomputed for each quadrature point, and used as in Eq. 5.

In contrast to standard interpolation, the MLS-based deformation field is *approximating* (Kronecker delta property is not satisfied). Hence, we use a variant of Nitsche's method [29] to approximate Dirichlet conditions. Alternatively, regularized weighting functions could be used [30]. The shape functions fulfill the convergence requirements of FEM, as the continuity condition is met by construction, and the completeness condition (strain-free rigid body motion) is fulfilled as long as we include the constant and linear terms in our polynomial basis.

To guarantee sufficient smoothness for optimization with 2nd order techniques while keeping the finite support of radius r as small as possible, we custom-crafted a C^3 weighting function (see Fig. 3 a, MLS weighting function)

$$w(x) = \begin{cases} 20\left(\frac{x}{r}\right)^7 - 70\left(\frac{x}{r}\right)^6 + 84\left(\frac{x}{r}\right)^5 - 35\left(\frac{x}{r}\right)^4 + 1, & 0 \leq x \leq r \\ 0, & r \leq x. \end{cases}$$

B. Simulating Robot Designs

Other than exchanging Lagrange with C^3-MLS shape functions in the definition of the deformation gradient

$$\mathbf{F}(\underline{\mathbf{X}}, \underline{\mathbf{x}}, \mathbf{X}) = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \sum_i \underline{\mathbf{x}}_i \frac{\partial N_i(\mathbf{X})}{\partial \mathbf{X}}, \tag{8}$$

we proceed analogously to standard FEM [31]: We discretize the integral over tetrahedral elements with an 8-point Gauss quadrature, and form the gradient and Hessian of the total potential energy E for minimization with standard Newton. For MLS, we use a linear polynomial basis $[1, X, Y, Z]^T$, or a quadratic basis with 6 additional monomials of degree 2. While counterintuitive at first glance, a linear basis is often sufficient as MLS performs a weighted least squares approximation at *every* evaluation point, resulting in a deformation field that can be made arbitrarily smooth by choosing an appropriate weighting function. Interestingly, with the same number of degrees of freedom as for standard FEM, MLS achieves a significantly smoother deformation field at a small price in computational efficiency due to more non-zero entries in the energy Hessian (or tangent stiffness matrix).

C. Taking Derivatives of Equilibrium Constraints

To solve our automated routing problem (Eq. 4), we implicitly enforce equilibrium constraints by running simulations to first-order optimality before evaluating target matching objectives, or their gradients.

To compute the analytical gradient of the distance metric $f_{\text{dist}}(\underline{\mathbf{x}}^t, \underline{\hat{\mathbf{x}}}^t)$ for a target t, we take the derivative of the respective equilibrium constraint in Eq. 4, and apply the implicit function theorem

$$\frac{\mathrm{d} f_{\text{dist}}}{\mathrm{d}(\mathbf{s}, \mathbf{p}^t)} = -\frac{\partial f_{\text{dist}}}{\partial \underline{\mathbf{x}}} \left(\frac{\partial^2 E}{\partial \underline{\mathbf{x}}^2}\right)^{-1} \frac{\partial^2 E_{\text{act}}}{\partial \underline{\mathbf{x}}\, \partial(\mathbf{s}, \mathbf{p}^t)}. \tag{9}$$

The second derivative of the total potential energy E is the simulation Hessian. Because only the actuation energy depends on the spline parameters and pressure values, we restrict the derivative of the force $\frac{\partial E}{\partial \underline{\mathbf{x}}}$ with respect to $\mathbf{s}$ and $\mathbf{p}^t$ to the actuation energy. For evaluations of the latter Jacobian, second derivatives of shape functions are required. This is due to the dependence of the strain on the deformation gradient, as we will discuss in Sec. VI. With the exception of the derivatives of shape functions, all derivatives can be found with a symbolic differentiation package. To form derivatives of shape functions, the derivative of the inverse of matrix $\mathbf{G}$ is needed: $\frac{\partial \mathbf{G}^{-1}}{\partial X} = -\mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial X} \mathbf{G}$ for the X-coordinate of $\mathbf{X}$.

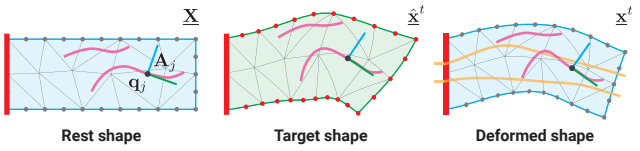


Fig. 4. **Measuring Distances to Targets**. To specify targets, we let the user specify deformations of the conformal surface mesh (middle), running a simulation with soft Dirichlet conditions to "fill in" the deformation for non-boundary nodes. To compare simulated to target deformations, we define *tracking curves* in form of spline curves (pink), measuring differences in sampled positions, $\mathbf{q}_j$, and corresponding frames $\mathbf{A}_j$.

V. Optimization

User-provided targets $\underline{\hat{\mathbf{x}}}^t$ are either specified by running forward simulations with user-specified external forces, or by specifying deformations of the conformal surface mesh with an external modeling tool, followed by a simulation where we approximate this deformation volumetrically with soft Dirichlet conditions [29] (see Fig. 4 middle). The former is the preferred option for simpler designs because a bound on the magnitude of the applied forces helps to keep strains within reasonable limits. For complex designs (e.g., the walking robot), it is tedious to achieve desired deformations with external forces, and we prefer to specify them with surface modeling tools. To keep strains within bounds for this second option, we optionally penalize large strains.

A. Measuring Distances To Targets

In most soft robotic tasks, there will be some part of the robot body where the deformation is more important than in other parts. For example, for a soft robotic gripper, one would typically place more importance to the deformation in areas of contact, whereas for a swimming continuum robot, one would want the overall shape of the robot to be matched. In many cases, it is still convenient for the user to specify a target deformation with a mesh, as this can be readily manipulated using standard software such as Autodesk Maya, however, the optimization should be guided towards where on the robot to match the target deformation.

To this end, we introduce the notion of *tracking curves*, which are spline curves going through the object (in pink in Fig. 4), and along which the deformation is measured and compared with the target. Specifically, we sample this sensor curves at a discrete set of locations $\mathbf{q}_j$, defining local frames whose orthonormal axes point in tangent, normal, and binormal directions, are collected as columns in an 3×3 orientation matrix $\mathbf{A}_j$ (see frames in Fig. 4). We then transform these sample points to their simulated and target locations

using Eq. 5, and corresponding orientations with the respective deformation gradient (Eq. 8), defining the distance metric

$$f_{\text{dist}}(\underline{\mathbf{x}}^t, \underline{\hat{\mathbf{x}}}^t) = w_{\text{pos}} \sum_j \frac{1}{2} \|\mathbf{x}(\underline{\mathbf{X}}, \underline{\mathbf{x}}^t, \mathbf{q}_j) - \mathbf{x}(\underline{\mathbf{X}}, \underline{\hat{\mathbf{x}}}^t, \mathbf{q}_j)\|^2$$
$$+ w_{\text{ori}} \sum_j \frac{1}{2} \|\mathbf{F}(\underline{\mathbf{X}}, \underline{\mathbf{x}}^t, \mathbf{q}_j) - \mathbf{F}(\underline{\mathbf{X}}, \underline{\hat{\mathbf{x}}}^t, \mathbf{q}_j)\|_F^2.$$

The weights w_{pos} and w_{ori} provide the user with a means to emphasize position, or orientation matching. Note that we dropped the local frame $\mathbf{A}_j$ in the second term, because it cancels out: if $\mathbf{D}$ is set to the difference between the two deformation gradients, we get $\|\mathbf{D}\mathbf{A}_j\|_F^2 = \text{tr}(\mathbf{D}^T\mathbf{D})$ for the squared Frobenius norm, because $\mathbf{A}_j$ is an orthonormal matrix.

Penalizing differences in deformation gradients provides an effective means to circumnavigate undesired minima. As we demonstrate with a validation example (Sec. VIII; Fig. 8), our optimization is largely insensitive to the user-specified initial routing of fibers.

To represent fibers in optimizations, we use cubic Hermite spline curves. We recall that the entry and exit points of the muscle are provided by the user, and remain fixed throughout optimizations. This means that other considerations can be accounted for (for example, in the case of the Tentacle, it is desirable to have the supply tube at the fixed end). While our pneumatic muscles do not require an exit point but could be terminated inside the body, it is advantageous to have an exit point for thermal actuators such as shape memory alloys or twisted Nylon actuators.

B. Regularizing Shape and Length of Actuators

Some regularization is required to ensure a meaningful solution. A first term of the regularizer $\mathcal{R}(\mathbf{s})$ in our routing problem (Eq. 4) penalizes high actuator curvature. This prevents designs in which muscles could malfunction, and simplifies fabrication. To prevent the muscle from getting too close to the outer surface of the robot, we add a term that penalizes, with a barrier function, small distances between actuator sample points and the outside surface of the undeformed design. This is implemented with an Axis-Aligned Bounding Box (AABB) tree, for speed. While it would be straightforward to add a penalty on the length, this is, in general, not necessary: because we keep the number of control points fixed, the length is naturally limited if the other two regularization terms are active.

For multi-fiber designs, we either optimize fibers simultaneously, or sequentially. The benefit of the first strategy is that simultaneous actuation can approximate a richer deformation space with less actuators. With the second strategy, we can keep actuators orthogonal, meaning that only one actuator is pressurized at a time.

Independent of the chosen strategy, muscles may overlap after optimizations. While it is important to let individual fibers "unfold" (freely cross) during optimizations, for optimal target matching quality, fabrication requires non-overlapping actuators. We manually move actuators apart if they cross after optimization. While these adjustments are relatively small due to the small cross-section of the actuators, a second optimization with a non-overlapping penalty between pairs of actuators could be performed.

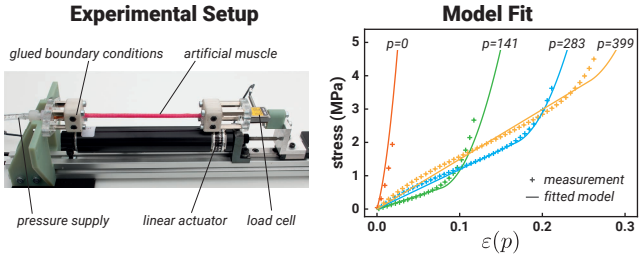


Fig. 5. **Muscle characterization.** Experimental setup, where we control the pressure and length of an actuator, and measure the resulting contractile force (left). Plot showing measured data points as well as the curves from our fitted model (right). As described in the text, the muscle strain ε is a function of p. Pressure values are in kPa. Note that only some pressure values are shown, for clarity.

To keep actuation pressure values in a permissible range $[0, p_{\text{max}}]$, we again use a barrier function, implementing the bound constraints in Eq. 4 with an objective term. Implicitly enforcing equilibrium constraints, and enforcing bound constraints with a penalty, we pave the way for unconstrained optimization with a quasi-Newton with a standard BFGS approximation of the Hessian [28].

VI. Modeling Artificial Muscle Actuators

We model artificial muscles as fibers routed through the body, coupled to the surrounding body. This is in contrast to cable-driven systems [10], where there is sliding between the cable and body. We assume that the actuators have a constant size in the radial direction, and we therefore do not account for the radial bulging of McKibben actuators.

Specifically, in this work, we consider thin (3 mm diameter) McKibben pneumatic artificial muscles (*S-muscle*, Japan). However, unlike typical applications, our muscles are embedded in silicone, meaning that fibers are not free to slide against each other. Thus, although there exist models for (unembedded) McKibben actuators [25, 26], these are not readily applicable here.

As introduced previously, we choose to model actuators as nonlinear rods or generalized springs, where their parameters as well as their rest length are functions of the muscle activation variable (pressure, in the case of pneumatic muscles).

Assuming that the actuator is homogeneous along its length, we perform characterization experiments on a finite-length sample, and then use this to fit length-invariant parameters suitable for actuator elements of arbitrary length. We assume the effect of actuator curvature to be negligible.

We perform a characterization experiment where we measure the blocking force of the actuator at different pressures, and for a set of lengths ranging between the length with unconstrained deformation at maximum pressure and the maximum obtainable length. The actuator is coated in a thin layer of silicone, to emulate its embedding. The experimental setup can be seen in Fig. 5 (left). We characterize the actuator behavior up to a pressure of 425 kPa, as increasing the pressure beyond this caused a significant increase in the actuator failure rate.

To fit our data-driven model to this behavior, we consider the rest length at zero pressure, and the rest length at non-zero pressure. Let $L(p)$ be the free deformation (zero-force) length of the actuator at pressure p, giving the strain $\varepsilon(p) = ({}^{l}/_{L(p)}) - 1$ when the actuator length is l. We also take $\varepsilon_L(p) = ({}^{L(p)}/_{L(0)}) - 1$ to be the strain relative to the free deformation state of the actuator at *zero* pressure (i.e. the "true" rest state of the actuator). In addition, we define the maximum strain $\varepsilon_{\max}(p) = ({}^{L(0)}/_{L(p)}) - 1$ to differentiate between cases where l is smaller or larger than the maximum length $L(0)$ at zero force. Recall that the artificial muscle contracts, hence $L(p) < L(0)$.

Assuming the cross-sectional area A_{act} of the actuator to remain constant, we divide measured forces by this area, then fit a pressure-dependent stress-strain relationship to the characterization data. At a given pressure p, we observe the stress to vary linearly with the strain for strains $\varepsilon \leq \varepsilon_{\max}$ (or, equivalently, for $l < L(0)$). Straining the actuator beyond its rest length at zero pressure, i.e. for $\varepsilon > \varepsilon_{\max}$, leads to a sharp rise in the stress (as the fibers become taut), which we model as quadratic polynomial. Our model for the muscle's stress can thus be written as

$$\sigma(p, \varepsilon(p)) = \begin{cases} E(p)\,\varepsilon, & \varepsilon(p) \leq \varepsilon_{\max}(p) \\ A(p)\,\varepsilon^2 + B(p)\,\varepsilon + C(p), & \varepsilon(p) \leq \varepsilon_{\max}(p) \end{cases}$$

where "Young's modulus" E and coefficient A are two-piece piece-wise linear functions of p, and B and C are set such that the curve is C^1-continuous at $\varepsilon_{\max}(p)$: $B = E - 2A\varepsilon_{\max}$ and $C = A\varepsilon_{\max}^2$. We find the appropriate coefficients for expressing E and B in terms of p using least-squares fitting. The rest strain ε_L is approximated with a linear function of p.

The overall fit of the model can be seen in Fig. 5 (right). The proposed model captures the muscle behavior well, while requiring only a small number of fitted parameters.

It is interesting to note that in our system, the quadratic part of the model is only required for cases where there are multiple embedded actuators, or where other external forces are acting; under their own actuation, the muscles can only contract in length.

To use this model in simulations and optimizations, we integrate to form the strain energy density

$$\Psi_{\text{act}}(p, \varepsilon(p)) = \begin{cases} \frac{1}{2}E(p)\varepsilon^2, & \varepsilon(p) \leq \varepsilon_{\max}(p) \\ \frac{1}{3}A(p)\varepsilon^3 + \frac{1}{2}B(p)\varepsilon^2 + C(p)\varepsilon + D(p), & \varepsilon(p) \leq \varepsilon_{\max}(p). \end{cases}$$

To preserve C^1-continuity, we set the integration constant D to $-\left(\frac{1}{3}A\varepsilon_{\max}^3 + \frac{1}{2}(B-E)\varepsilon_{\max}^2 + C\varepsilon_{\max}\right)$.

For numerical optimization, we require derivatives of this data-driven muscle model with respect to the pressure p. Because of the two-piece piece-wise linear approximation of E and A, the model is not sufficiently smooth. Thus, to make the strain energy differentiable, we turn these two pressure-dependent coefficients into C^∞ functions featuring the appropriate oblique asymptotes, as we describe in our Appendix.

A. Discretizing Muscle Fibers

To discretize the integral in the actuator energy E_{act} (Eq. 2), we partition spline curves into line segments (compare with Fig. 6), connecting segment end points with vectors $\mathbf{d}_j$, and averaging them to form their centers $\mathbf{c}_j$. We then compute the rest and deformed length of segments,

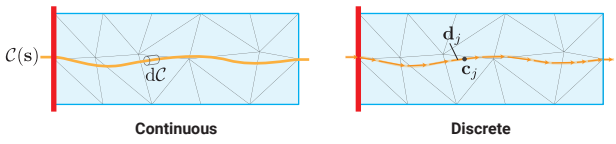


Fig. 6. **Discretizing Artificial Muscles**. To discretize actuation fibers (left), we split them into straight line segments, representing segments with vectors $\mathbf{d}_j$ and center points $\mathbf{c}_j$ (right).

$$L_j(\mathbf{s}, p) = (\varepsilon_L(p) + 1)\ \|\mathbf{d}_j(\mathbf{s})\| \ \text{ and}$$
$$l_j(\mathbf{s}, \underline{\mathbf{X}}, \underline{\mathbf{x}}) = \|\mathbf{F}(\underline{\mathbf{X}}, \underline{\mathbf{x}}, \mathbf{c}_j(\mathbf{s}))\ \mathbf{d}_j(\mathbf{s})\|,$$

to compute their strain ε_j, and use the midpoint rule to approximate the integrated actuator energy (Eq. 2). Because we assume the actuator to be bonded to the surrounding soft body, the deformed length of an actuator element is the product of the deformation gradient, evaluated at the segment center point, and the linear approximation of the segment. Note that the rest length of segments depends on the pressure. Hence, we multiply the linear segments with the pressure-dependent "rest" strain, $\varepsilon_L(p)$, plus one, resulting in the discretized integral

$$\sum_j \Psi_{\text{act}}(p, \varepsilon_j) A_{\text{act}} L_j \quad \text{with} \quad \varepsilon_j(\mathbf{s}, p, \underline{\mathbf{X}}, \underline{\mathbf{x}}) = \frac{l_j}{L_j} - 1. \quad (10)$$

VII. Fabrication

Robots are fabricated from silicone elastomer (EcoFlex 00-30, Smooth-On) using a two-step injection molding process, similarly to [11]. The injection is done with a pneumatic dispenser gun; compared to gravity-assisted pouring, this greatly reduces problems of air entrapment, in particular for complex geometries. The fabrication steps are shown in Fig. 7. The groove geometry is generated automatically with a script, given a muscle routing, so the mold design complexity is independent of the routing complexity. This also means that little manual effort is required to fabricate molds for alternative muscle routings (e.g. the two variants of the Tentacle). Molds are printed on a Stratasys Objet Connex 350. After molding, the artificial muscles are terminated by attaching a Luer Lock fitting to one end and sealing the other. We observe good bonding between the silicone from the two molding steps, and also between the silicone and the muscle.

We note that the particular artificial muscles used in this work have, in the rest configuration, a gap of ~0.5 mm between the inner silicone tube and the outer braid. While the gap is filled during the molding process, this introduces some uncontrollable variation in the muscle position, and therefore also in the deformation behavior.

VIII. Results

To generate our results, we model the silicone material with a Neo-Hookean material (Young's modulus set to 86.4×10^3

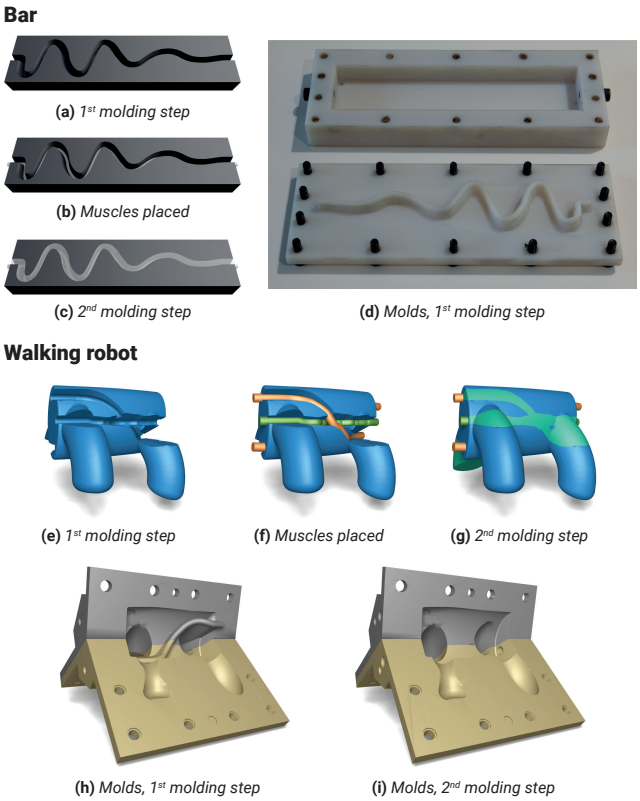


Fig. 7. **Fabrication process.** Steps and mold designs for Bar (Fig. 10) and Walking robot (Fig. 16). First, the robot is fabricated with grooves for the artificial muscles (a, e). The muscles are then placed in the grooves (b, f), and overmolded (c, g). Note that for (h) and (i), one mold part is not shown.

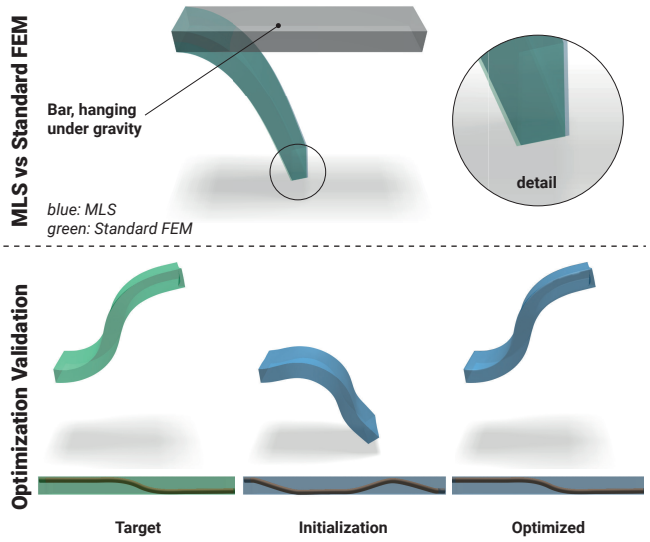


Fig. 8. **Simulation validation (top).** We compare the prediction of our MLS-based formulation and a standard FEM model on a cantilever bar (length: 100 mm), hanging under gravity; we observe excellent agreement (RMSE: 1.01 mm; max error: 1.82 mm). **Optimization validation (bottom).** We manually place a muscle inside a bar (length: 100 mm) and actuate it to generate the target deformation. We then perturb the muscle, leading to a significantly different deformation, and use this as an initialization for our optimization. The optimization recovers the muscle shape near-perfectly (RMSE: 0.10 mm; max error: 0.20 mm).

Pa; Poisson's ratio to 0.47), and set the MLS radius r to a globally constant value of 1.7 times the average edge length of the volumetric mesh. We use a linear MLS basis for our bar, Tentacle and Walking robot examples, and a quadratic basis for our Continuum robot.

For the data-driven actuator model, we use two line segments with slope-intercept pairs $(-549.92,\ 73.068 \times 10^6\,\text{Pa})$ and $(26.36,\ 4.40 \times 10^6\,\text{Pa})$ to represent $E(p)$, and the two pairs $(-23.91 \times 10^3,\ 4067.66 \times 10^6\,\text{Pa})$ and $(-986.60,\ 833.51 \times 10^6\,\text{Pa})$ to represent $A(p)$ (assuming SI units for all quantities). $\varepsilon_L(p)$ is represented with the line $(-0.512 \times 10^{-6},\ 0)$. To smooth functions $E(p)$ and $A(p)$, we use $\rho = 10^4\,\text{Pa}$ (see formula in our Appendix).

A. Prediction Quality of MLS-Formulation

To quantify the prediction accuracy of our technique, we consider a cantilever bar, hanging under gravity. We run the same simulation using a standard FEM model and our MLS-based formulation. As can be seen in Fig. 8 (top), we observe excellent agreement of predictions at nodes of standard quadratic elements.

B. Navigating the Design Space

As a validation of our optimization system, and to investigate the sensitivity of our technique w.r.t. initializations that are far away from the optimal solution, we consider a case where we know that a perfect solution exists; see Fig. 8 (bottom). We manually place an S-shaped actuator in a bar, resulting in an S-shaped deformation when actuated. We take the resulting deformation as our target, and then perturb the actuator such that a significantly different deformation mode is seen under actuation. This perturbed muscle shape is then given as an initial guess to the optimization. As shown in the figure, the optimization is able to recover the muscle shape and resulting deformation near-perfectly. Our system therefore exhibits little sensitivity to initializations, even when the initialization leads to significantly different deformations. For this example, we place a tracking curve along the centerline of the bar.

C. Twisting bar

As a first example of our full pipeline, including physical fabrication, we consider a bar (length: 129 mm) which is fixed at one end and undergoing a twisting deformation. Again, we place a tracking curve along the centerline of the bar, and we also initialize the actuator by routing it along this centerline. We then optimize for the actuator routing and the actuation pressure. The target deformation, resultant optimized actuator routing, and resultant optimized deformation can be seen in Fig. 9. For the target, the angle of twist at the distal end of the bar is 39.3°. It is important to note that as target deformations were not generated by embedded artificial muscle actuators, we can in general not expect targets to be matched perfectly.

We then fabricate the bar, and compare the simulation prediction to the physical results (see Fig. 10). To compare the two, we 3D-print the simulation result, capture images of the two, and overlay them. There is good correspondence between

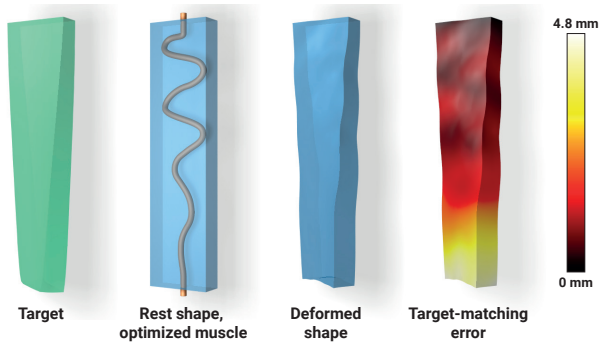


Fig. 9. **Twisting bar, simulation.** The optimization captures the overall deformation with relatively low error (RMSE: 1.9 mm; max error: 4.8 mm). The optimization was initialized with a straight muscle.

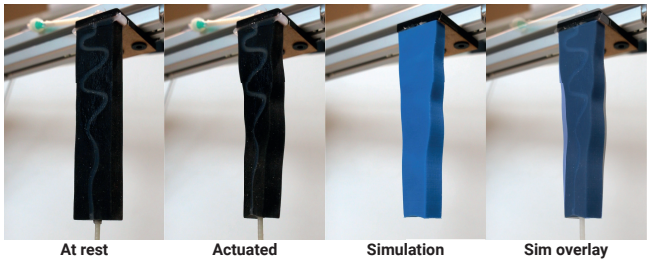


Fig. 10. **Twisting bar, experiment.** We 3D-print the simulation result, so that the results can be overlaid. The simulation matches the physical result closely, including local undulations along the length of the bar and also the concavity at the end of the bar.

the two. We highlight in particular the undulations along the side of the bar resulting from the helical actuator shape, and also the concavity at the end of the bar, both of which the simulation captures well. As an additional comparison, we measure the angle of twist at the distal end of the bar, in simulation and on the physical bar, and we observe excellent agreement: both have a twist angle of 28.3°.

See also the supporting video, where we show the bar being actuated.

D. Continuum robot

In this example we consider a soft underwater continuum robot with a cylindrical shape (16 mm diameter, 300 mm length) that is fixed at one end. Such a robot could be used to inspect underwater structures. The robot is neutrally buoyant (water is used as a working fluid in the actuators), so the effect of gravity is negligible. We initialize the robot with a single actuator along its medial axis, and also place a tracking curve at the medial axis, along the full length of the robot. As a deformation target to match, we specify a spatial target shape which involves bending about multiple axes and continuously-varying bend curvature. As the continuum robot is rotationally symmetric, we in this example only track *positions* along the tracking curve, and not rotations (in the other examples, we track positions and rotations along the tracking curve, as the robots are not rotationally symmetric and the robot twist matters).

Fig. 11 shows the input, as described above, along with the result of the optimization. Overall, it can be seen that

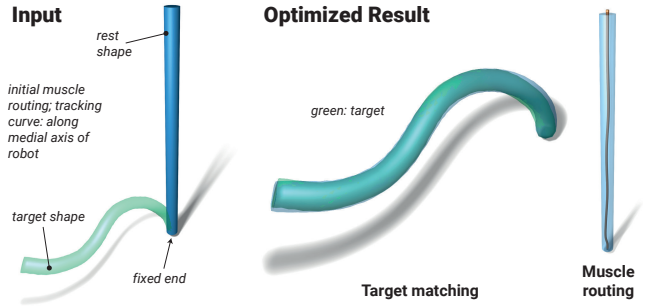


Fig. 11. **Continuum robot, simulation** The target shape is spatial and requires bending about multiple axes. The optimized result matches the specified target well: RMSE 1.7 mm, max error 3.2 mm. Note that for this example, as it is rotationally symmetric, we report the target matching error along the tracking curve.

the optimization result matches the desired target shape well along the full length of the robot. Interestingly, the maximum target matching error as computed between the target mesh and optimization result (which is not twist-invariant) is significantly larger at 12.1 mm, due to differences in twist. In informal experiments, we observed that if we asked the optimization to also match the twist values for this robot, we obtained significantly worse overall performance. This illustrates the importance of specifying targets that pertain to the robot's function: in this example, twist is not important to the function of the robot, and leaving it unconstrained allowed the optimization to find a better solution overall.

The robot is fabricated, and experimental results can be seen in Fig. 12. As the experiment is conducted under water, a Mocap system cannot readily be used for tracking. Instead, we 3d print the simulation result and mount that at the same location, and then overlay this onto the robot—see rightmost panels in figure.

It can be seen that overall, there is good agreement between the fabricated robot and the simulation result. The maximum target-matching error, estimated using image analysis tools, is 9.0 mm. See also the supporting video.

E. Tentacle

The octopus tentacle is a well-known example from nature of exploiting softness, and also a well-studied example in Soft Robotics (see, e.g., [32, 33]). We start with a straight tentacle (length: 298 mm), and generate a target deformation which combines bends about multiple axes to produce a complex spatial shape.

To illustrate how tracking curves allow the user to tune the optimization result, we show results for two different cases. For the *full-length* condition, we place a tracking curve along the full length of the tentacle, which causes the optimization to try to match the position and orientation of the tentacle along its entire length. For the *tip only* condition, we place a tracking curve at the tip of the tentacle only, so that the optimization will only try to match the position and orientation of the tip.

Putting this into context; for a robot executing a task such as grasping or locomotion where only the tip interacts with the environment, it is more important to control the end

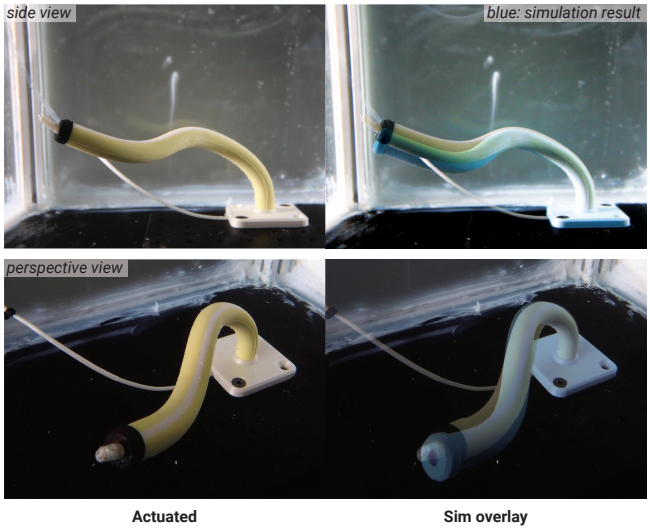


Fig. 12. **Continuum robot, experiment** The robot is submerged in water. The right-hand panels show the 3d-printed simulation result, overlaid the fabricated robot.

effector position. However, for full-bodied locomotion such as swimming, it will be important to control the displacement everywhere.

The optimization is initialized with a single muscle along the centerline of the tentacle, and the results of the optimization for the two conditions are shown in Fig. 13. Again, the target deformation was created manually by a user, and can therefore not be expected to be matched perfectly. In particular, the McKibben artificial muscle actuators unavoidably induce length contraction under actuation. With this in mind, the performance of the optimization is good for both conditions. The results match expectation, with the *tip only* condition exhibiting smaller error at the tip while the *full-length* distributes the error more evenly.

We fabricate both versions of the tentacle, and submerge them in water. We use water as a working fluid in the actuators, instead of air, to prevent unwanted buoyancy. The results are shown in Fig. 14, and we also refer to the supporting video. As for the twisting bar, we 3D-print the simulation result and overlay images of the two results. In general, we observe reasonable agreement between the simulated and physical results. In particular, it is worth noting the local undulations along the length for the *tip-only* condition which the simulation predicts very well. Using image processing software, we find the maximum target matching error to be 20.5 mm for the *full-length* case and 40.1 mm for the *tip-only* case.

It can be seen that in both conditions, the error is largest at the distal end of the tentacle. This can be explained in part by the accumulation of error along the tentacle, but other likely sources of error are the modeling assumption of the negligible size of the actuator in the radial direction, and the MLS-shape functions that are known to not work well for rods [22] (co-linearity of FE DoFs can lead to rank-deficiencies in matrices $\mathbf{G}$). Towards the distal end, where the tentacle is thinnest, it is likely that these assumptions introduce additional error. More specifically, the previous result demonstrated good target

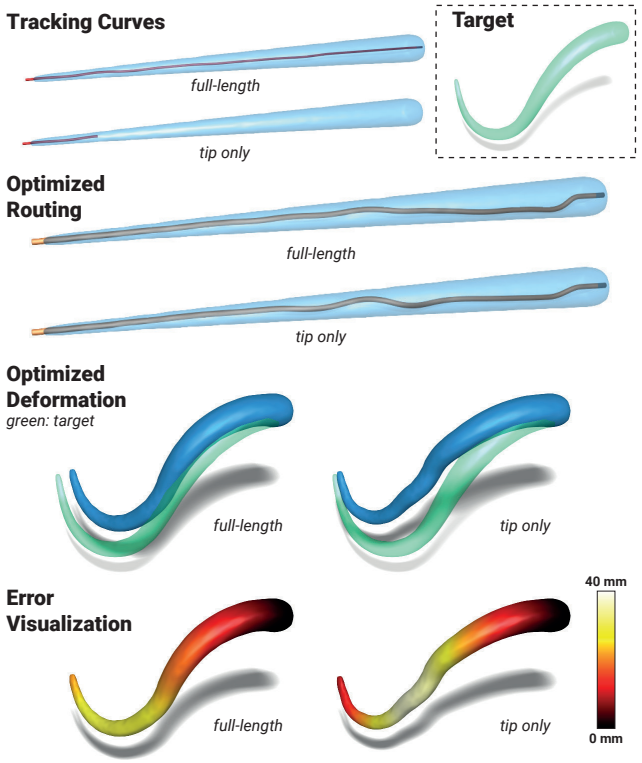


Fig. 13. **Tentacle, simulation.** We study two conditions: for *full-length* we place a deformation sensor along the length of the tentacle, and for *tip only* we only try to match the target near the tip. The results match expectation, with *tip only* having smaller error at the tip but larger error elsewhere. *Full-length:* RMSE: 20.3 mm; max error: 29.5 mm. *Tip only:* RMSE: 24.0 mm; max error: 39.6 mm. Note that this error is reported for the full mesh, not just at the tracking curve.

matching performance for a robot of diameter 16 mm (5.3× actuator diameter), but we observe a larger error at the tip of the tentacle, which has diameter 6 mm (2× actuator diameter).

F. Walking robot

In this example, we consider an entirely-soft 4-legged walking robot (body length: 42 mm). We take as input the robot geometry, and to achieve a walking gait we define two targets of the body: *bend* and *twist*. As can be seen from Fig. 15, the *twist* deformation swaps which legs are touching the ground, and the *bend* deformation causes the robot to step forward. A gait cycle is then formed by {*rest*, *bend*, *bend+twist*, *twist*}. We make the somewhat crude assumption that we can achieve a *bend+twist* by actuating *bend* and *twist* simultaneously; the results in Fig. 15 show that, in this case, this assumption is reasonable. As the mass of the robot is small, we can consider the effect of gravity to be negligible. As we seek to route actuators through the body, this means that the legs will undergo negligible deformations, so for the optimization we consider the body in isolation. We then simulate the full robot, including the legs, with the optimized muscles, confirming that the deformations are still as expected. For the optimization of both the *bend* and *twist* poses we fix one end of the robot body and place a tracking curve at the other end of the body, as seen in the figure.

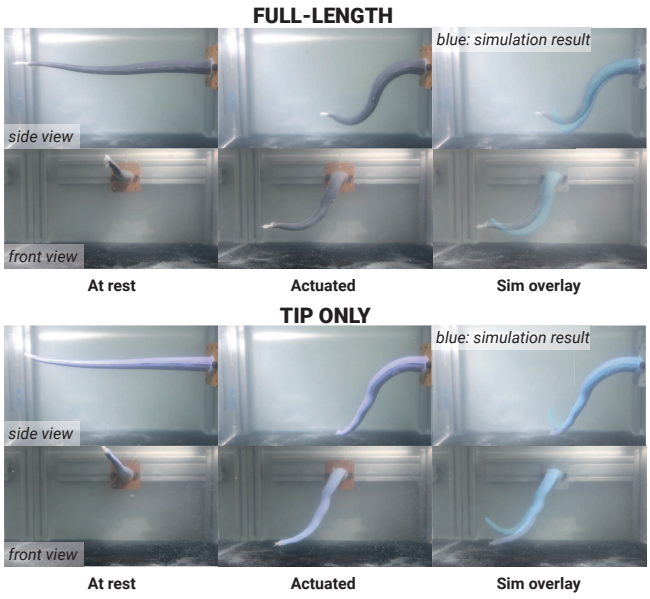


Fig. 14. **Tentacle, experiment.** See Fig. 13 for the two conditions; *full-length* and *tip only*. The tentacle is submerged in water, and the rightmost panels show the 3D-printed simulation result, overlaid the fabricated tentacles.

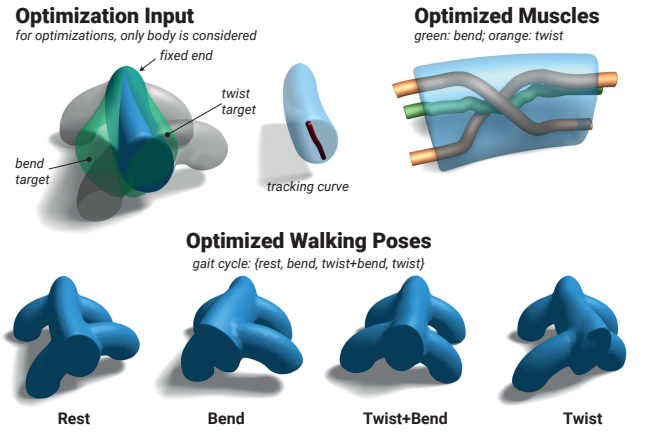


Fig. 15. **Walking robot, simulation**. For the optimization, we consider just the robot body. We define two target poses, *bend* and *twist*. A single deformation sensor is used. After the optimization, we add the optimized actuators to the full robot simulation and simulate into the two poses. We also run a forward simulation of the *bend+twist* pose, which is the superposition of *bend* and *twist* actuation. The result is a simple walking gait cycle.

For the *bend* deformation, we initialize the optimization with a single actuator along the medial axis of the body. For the *twist* deformation, we take inspiration from the Twisting Bar results (Fig. 9) and use a helical routing as the initialization. In order to enable a symmetric solution that does not introduce unwanted bending, we initialize a pair of actuators. This illustrates how our system supports co-optimization of multiple actuators.

We then combine the actuators resulting from the *bend* and *twist* optimizations. Our system does not check for collisions between actuators, so collisions are resolved manually in a post-processing step by offsetting one of the colliding actuators. Again, a forward simulation of the processed result is used to check that the function is preserved.

Fig. 16. **Walking robot, experiment.** The body length of the robot is 42 mm. The robot achieves a forward velocity of 7.9 body lengths/min. See also the supporting video. The two colors on the robot body come from the two-step molding process, see Fig. 7.

TABLE II
KEY STATISTICS.

	#tets	#DoFs	sim (s)	opt (min)	opt (iter)
Bar	2'442	702	63	169	381
Cont. Rob.	2'033	707	539	62	121
Tent. (full)	902	330	48	88	365
Tent. (tip)			47	58	463
Rob. (bend)	5'553	1'301	122	76	18
Rob. (twist)			97	85	31

#tets: number of tetrahedra; *#DoFs*: number of FE DoFs; *sim (s)*: simulation time in s, for optimized actuator and starting from rest pose; *opt (min)*: optimization time in min; *opt (iter)*: optimization iterations.
Bar: Twisting bar; *Cont. Rob.*: Continuum robot; *Tent.*: Tentacle; *Rob*: Walking robot (body only).

The four stages of the gait cycle for the simulated robot are shown in Fig. 15. The fabricated robot is presented in Fig. 16. See also the supporting video. The physical robot is driven using a valve array (Festo VTUG series) controlled by an Arduino, and achieves a forward velocity of 7.9 body lengths/min.

G. Performance

All simulations and optimizations were performed on a machine with an Intel Core i7-7700 processor (4 cores, 4.2 GHz) and 32 GB of RAM. We summarize key statistics for the different demonstrators in Tab. II. The walking robot requires fewer optimization iterations than the other demonstrators due to the relative simplicity of the bend deformation, and the informed initialization with a helical routing for the twist deformation.

IX. Conclusion

This paper has introduced an automated system for optimally routing one-dimensional artificial muscle actuators through arbitrary hyperelastic robots, so that a desired deformation behaviour is matched as closely as possible. The optimization is built on a differentiable finite element model, which uses MLS to achieve a C^3-continuous deformation field, allowing artificial muscles to move freely through element boundaries during the optimization. Furthermore, we have presented a simple data-driven model which captures the

behavior of embedded artificial muscles well, and can readily be integrated with standard finite element models.

As the system uses a finite element model, it generalizes well to arbitrary robot geometries. We expect the data-driven actuator model to generalize well to other types of one-dimensional artificial muscles, for example shape memory alloys or twisted Nylon actuators [1].

More generally, our work illustrates the potential of computational design tools for tackling soft robotic design problems; enabling designs and deformation behaviors that would be very difficult to achieve otherwise. To date, the field of Soft Robotics has been driven primarily by experimental research and prototype-driven development [34, 35, 36]. However, as the field advances, and the complexity of the tackled problems grows, we envisage that simulation and optimization will become increasingly important tools in the robot design process. The generality of these tools to handle arbitrary robots and geometries will be an important factor for their uptake.

A. Limitations and Future work

As our method integrates well with standard finite element simulations, it would be straightforward to combine this with multi-material robots, for more extreme deformations. In particular, there are recent results in multi-material, or meta-material, optimization [37, 9] that it would be exciting to combine with our approach in order to co-optimize the material distribution along with the actuation. This would also require adaptation of the fabrication pipeline.

Computational design tools are particularly well suited for emerging fabrication technologies, such as 3D printing or direct ink writing, which show great potential for fabricating soft robots [35]. This could enable large and complex actuator networks to be produced at the click of a button, and one can also envisage a combination of actuation and sensing [11] networks.

In our presented pipeline, the user prescribes the number of actuators along with their entry and exit locations of the actuators on the robot body. This is convenient for integrating other considerations, e.g. number of available pneumatic channels, keeping entry/exit points away from functionally or visually important areas, and accounting for supply tube routing. However, in particular for larger and more complex systems, an interesting avenue for future work would be to also optimize the *number* of actuators, and to automate the initial actuator routing. The work in [11] could provide a starting point for how to tackle the discrete problem of adding/removing actuators, and also describes a method for automated initialization. Further to this, we have not fully explored the space of optimizing for multiple deformation targets and multiple muscles—in particular, it is interesting to consider if N targets can be matched with $< N$ muscles.

For our tentacle example, we observe a sim-to-real gap at the thin tip. For rods and shells, FE degrees of freedom are close to co-linear or co-planar, resulting in rank-deficiencies in computations of MLS shape functions. The generalized moving least squares approach described by Martin et al. [22] could mitigate this problem. Moreover, for meshes with different-sized elements, an adaptively chosen MLS radius could improve performance, and simulation stability.

As illustrated by our results, the modeling assumption of the actuator having negligible size in the radial direction breaks down as the robot becomes too thin. Our results indicate that this assumption is reasonable for a ratio of robot diameter to actuator diameter of $\sim$5, but as this ratio becomes smaller one should expect sim-to-real mismatches to increase. A refined model, where the finite radial dimension of the actuator is accounted for by "cutting away" the surrounding matrix material, is non-trivial to implement as our optimization requires the actuator to be able to move smoothly through the underlying mesh. One approach could be to leverage eXtended Finite Element Modeling (XFEM) [38].

Furthermore, we can attribute some of the sim-to-real gap to tolerances in the fabrication process. As discussed previously, the most prominent source is likely to be the $\sim$0.5 mm gap between the inner silicone tube and the outer braid of the actuator. For context, in the Continuum robot, the muscle is a mean distance of 1.3 mm away from the medial axis of the robot. As we purchased the actuators off-the-shelf, this was not something we could improve on, however, there is not a fundamental reason why this could not be improved.

In this work, we have considered the deformation behavior to be the end goal. However, our system could also be used as a building block in a more task-driven optimization problem — for example the grasping problem considered by [5].

APPENDIX

To turn a two-piece continuous piece-wise linear function

$$f(x) = \begin{cases} ax + b, & x \leq x_{\text{intersec}} \\ cx + d, & x > x_{\text{intersec}} \end{cases} \tag{11}$$

with intersection point $x_{\text{intersec}} = -\frac{d-b}{c-a}$, into a C^∞ function, we use the smoothed version

$$\rho\tfrac{c-a}{2}\ln\left(\cosh\left(\tfrac{x-x_{\text{intersec}}}{\rho}\right)\right) + \tfrac{a+c}{2}x + \left(\tfrac{b+d}{2} + \rho\tfrac{c-a}{2}\ln(2)\right)$$

where ρ controls the width of the transition between the two pieces.

REFERENCES

[1] C. S. Haines, M. D. Lima, N. Li, G. M. Spinks, J. Foroughi, J. D. Madden, S. H. Kim, S. Fang, M. J. De Andrade, F. Göktepe, *et al.*, "Artificial muscles from fishing line and sewing thread," *Science*, vol. 343, no. 6173, pp. 868–872, 2014.

[2] J. Hiller and H. Lipson, "Automatic design and manufacture of soft robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 457–466, 2011.

[3] F. Connolly, P. Polygerinos, C. J. Walsh, and K. Bertoldi, "Mechanical programming of soft actuators by varying fiber angle," *Soft Robotics*, vol. 2, no. 1, pp. 26–32, 2015.

[4] F. Connolly, C. J. Walsh, and K. Bertoldi, "Automatic design of fiber-reinforced soft actuators for trajectory matching," *Proceedings of the National Academy of Sciences*, vol. 114, no. 1, pp. 51–56, 2017.

[5] R. Deimel, P. Irmisch, V. Wall, and O. Brock, "Automated co-design of soft hand morphology and control strategy for grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1213–1218.

[6] D. R. Ellis, M. P. Venter, and G. Venter, "Computational design for inflated shape of a modular soft robotic actuator," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 7–12.

[7] L. Ding, N. Dai, X. Mu, S. Xie, X. Fan, D. Li, and X. Cheng, "Design of soft multi-material pneumatic actuators based on principal strain field," *Materials & Design*, vol. 182, p. 108000, 2019.
[8] X. Peng, N. Zhang, L. Ge, and G. Gu, "Dimension optimization of pneumatically actuated soft continuum manipulators," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 13–18.
[9] L.-K. Ma, Y. Zhang, Y. Liu, K. Zhou, and X. Tong, "Computational design and fabrication of soft pneumatic objects with desired deformations," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 239, 2017.
[10] J. M. Bern, K.-H. Chang, and S. Coros, "Interactive design of animated plushies," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 80, 2017.
[11] J. Tapia, E. Knoop, M. Mutný, M. A. Otaduy, and M. Bächer, "Makesense: Automated sensor design for proprioceptive soft robots," *Soft Robotics*, vol. 0, no. 0, pp. 0–0, 2019.
[12] T. Morzadec, D. Marcha, and C. Duriez, "Toward shape optimization of soft robots," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 521–526.
[13] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross, "Computational design of actuated deformable characters," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 82:1–82:10, July 2013.
[14] V. Megaro, E. Knoop, A. Spielberg, D. I. Levin, W. Matusik, M. Gross, B. Thomaszewski, and M. Bächer, "Designing cable-driven actuation networks for kinematic chains and trees," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2017, p. 15.
[15] H. Xu, E. Knoop, S. Coros, and M. Bächer, "Bend-it: design and fabrication of kinetic wire characters," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 239.
[16] M. Skouras, B. Thomaszewski, B. Bickel, and M. Gross, "Computational design of rubber balloons," *Comput. Graph. Forum*, vol. 31, no. 2pt4, pp. 835–844, May 2012.
[17] M. Skouras, B. Thomaszewski, P. Kaufmann, A. Garg, B. Bickel, E. Grinspun, and M. Gross, "Designing inflatable structures," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 63:1–63:10, July 2014.
[18] V. Megaro, J. Zehnder, M. Bächer, S. Coros, M. Gross, and B. Thomaszewski, "A computational design tool for compliant mechanisms," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 82:1–82:12, July 2017.
[19] M. Bächer, B. Hepp, F. Pece, P. G. Kry, B. Bickel, B. Thomaszewski, and O. Hilliges, "Defsense: Computational design of customized deformable input devices," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 3806–3816.
[20] T.-P. Fries and H. G. Matthies, "Classification and overview of meshfree methods," *Informatik-Berichte der Technischen Universität Braunschweig*, vol. 2003-03, 2004.
[21] P. Breitkopf, A. Rassineux, G. Touzot, and P. Villon, "Explicit form and efficient computation of mls shape functions and their derivatives," *International Journal for Numerical Methods in Engineering*, vol. 48, no. 3, pp. 451–466, 2000.
[22] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross, "Unified simulation of elastic rods, shells, and solids," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 39:1–39:10, 2010.
[23] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, "Point based animation of elastic, plastic and melting objects," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '04, 2004, pp. 141–151.
[24] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas, "Meshless animation of fracturing solids," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 957–964, 2005.
[25] F. Daerden and D. Lefeber, "Pneumatic artificial muscles: actuators for robotics and automation," *European journal of Mechanical and Environmental Engineering*, vol. 47, pp. 10–21, 2000.
[26] S. Davis, N. Tsagarakis, J. Canderle, and D. G. Caldwell, "Enhanced modelling and performance in braided pneumatic muscle actuators," *The International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 213–227, 2003.
[27] E. Sifakis and J. Barbič, *Finite Element Method Simulation of 3D Deformable Solids*. Morgan & Claypool Publishers, 2015.
[28] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 1999.
[29] J. Nitsche, "Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind," *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, vol. 36, no. 1, pp. 9–15, Jul 1971.
[30] Y. Mori and T. Igarashi, "Plushie: An interactive design system for plush toys," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.
[31] J. Reddy, *An Introduction to Nonlinear Finite Element Analysis: with applications to heat transfer, fluid mechanics, and solid mechanics*. OUP Oxford, 2014.
[32] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced Robotics*, vol. 26, no. 7, pp. 709–727, 2012.
[33] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1109–1122, 2014.
[34] R. F Shepherd, F. Ilievski, W. Choi, S. A Morin, A. A Stokes, A. Mazzeo, X. Chen, M. Wang, and G. M Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, pp. 20 400–3, 11 2011.
[35] M. Wehner, R. L. Truby, D. J. Fitzgerald, B. Mosadegh, G. M. Whitesides, J. A. Lewis, and R. J. Wood, "An integrated design and fabrication strategy for entirely soft, autonomous robots," *Nature*, vol. 536, no. 7617, p. 451, 2016.
[36] T. Doi, S. Wakimoto, K. Suzumori, and K. Mori, "Proposal of flexible robotic arm with thin mckibben actuators mimicking octopus arm structure," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5503–5508.
[37] J. Zehnder, E. Knoop, M. Bächer, and B. Thomaszewski, "Metasilicone: design and fabrication of composite silicone with desired mechanical properties," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.
[38] P. Kaufmann, S. Martin, M. Botsch, E. Grinspun, and M. Gross, "Enrichment textures for detailed cutting of shells," *ACM Trans. Graph.*, pp. 1–10, 2009.

Guirec Maloisel is a Ph.D. student with the Computational Design and Manufacturing group at Disney Research and also the Computational Robotics Lab at ETH Zurich. He holds Masters degrees from EPFL and from École Polytechnique (Paris).

Espen Knoop is an Associate Research Scientist at Disney Research. He has worked on novel sensing and actuation methods for soft robots and novel interaction modalities offered by soft robotic technologies. He contributes to technologies that enables the creation of more expressive and dynamic robotic characters, by leveraging simulation and optimization tools. He holds Ph.D. and master's degrees from the University of Bristol.

Christian Schumacher is an Associate Research Scientist at Disney Research. He has been involved in research on computational design methods—proposing new approaches to integrate physics in the digital design process and enhance its capabilities through numerical optimization—and physically-based simulations for computer animation. He holds Ph.D. and master's degrees from ETH Zurich.

Moritz Bächer is a Research Scientist at Disney Research, where he leads the Computational Design and Manufacturing group. He is deeply passionate about solving real-world problems in computational robotics, fabrication, and architecture. His core expertise is the development of differentiable simulators to tackle complex design, control, and characterization problems in (soft) robotics, architecture, and computer graphics. Before joining Disney, he received a Ph.D. from the Harvard School of Engineering and Applied Sciences and graduated with a master's from ETH Zurich.