Interactive Design of Animated Plushies

JAMES M. BERN, Carnegie Mellon University
KAI-HUNG CHANG, Carnegie Mellon University
STELIAN COROS, Carnegie Mellon University

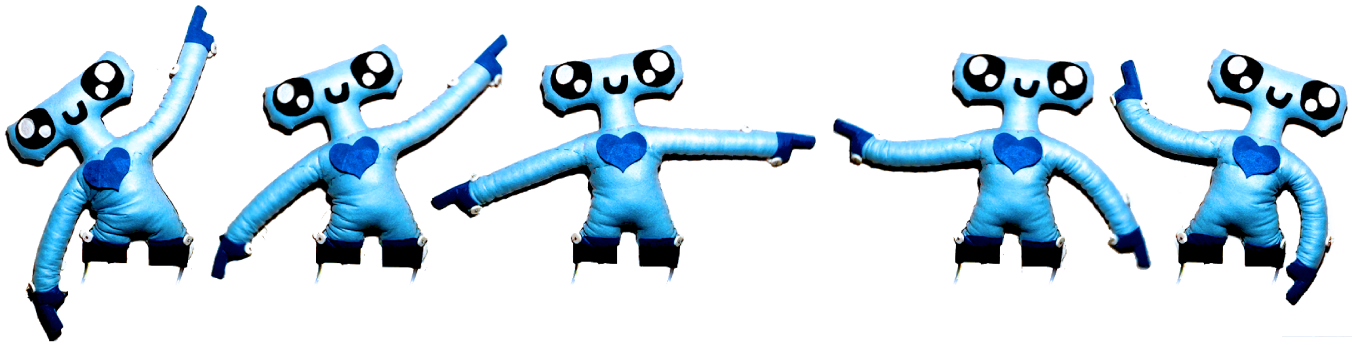


Fig. 1. An animated plushie designed and fabricated using our computational approach. Tendons run through the plushie's skin, and are contracted by motorized winches inside of the plushie's body.

We present a computational approach to creating animated plushies, soft robotic plush toys specifically-designed to reenact user-authored motions. Our design process is inspired by muscular hydrostat structures, which drive highly versatile motions in many biological systems. We begin by instrumenting simulated plush toys with a large number of small, independently-actuated, virtual muscle-fibers. Through an intuitive posing interface, users then begin animating their plushie. A novel numerical solver, reminiscent of inverse-kinematics, computes optimal contractions for each muscle-fiber such that the soft body of the plushie deforms to best match user input. By analyzing the co-activation patterns of the fibers that contribute most to the plushie's motions, our design system generates physically-realizable winch-tendon networks. Winch-tendon networks model the motorized cable-driven actuation mechanisms that drive the motions of our real-life plush toy prototypes. We demonstrate the effectiveness of our computational approach by co-designing motions and actuation systems for a variety of physically-simulated and fabricated plushies.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; • **Computer systems organization** → *Robotic control*;

Additional Key Words and Phrases: animation, plushies, computational design, soft robotics

ACM Reference format:
James M. Bern, Kai-Hung Chang, and Stelian Coros. 2017. Interactive Design of Animated Plushies. *ACM Trans. Graph.* 36, 4, Article 80 (July 2017), 11 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073700

0730-0301/2017/7-ART80 $15.00
DOI: http://dx.doi.org/10.1145/3072959.3073700

1 INTRODUCTION

Endeared companions to children and adults alike, plush toys have enjoyed widespread popularity since their commercial debut in the late 1800's. Thanks to the relative ease with which they are made, a Do-It-Yourself movement for plush toys emerged shortly thereafter and continues to flourish today. Recognizing an interesting technical challenge, the research community has taken an interest in formalizing computational design methods that facilitate the creation of plush toys [Igarashi and Igarashi 2008; Mori and Igarashi 2007] and other related structures made by adjoining flat patterns [Furuta et al. 2010; Skouras et al. 2014]. The solutions proposed to date help non-experts create designs that depict *static* shapes. In contrast, our goal is to make equally accessible the creation of plush toys that are specifically-designed to produce compelling motions.

To create animated plush toys, it is common practice to enclose traditional robotic devices in fabric exteriors. While great from the point of view of motion capabilities, this approach has two important drawbacks. First, the rigid nature of the embedded electromechanical assemblies hinders the large, organic deformations that are expected during typical interactions. Second, designing and fabricating the internal mechanisms can be very difficult, and therefore largely inaccessible to casual users. An alternative strategy, which we adopt for our work, is to employ cable-driven setups. In this setting, motions are created by activating, through pulling or contractions, cables that are laid out within the soft body of the plush toys. Ensuring that the resulting motions are desirable, however, requires solving a challenging design problem: the number of cables to be used, their routing, the patterns of co-activation, and the physical interactions between the tensioned cables and the plush toy are all coupled and must be considered concurrently.

Overview and contributions. We propose a computational approach to interactively designing cable-actuated robotic plush toys.

Our design methodology is characterized by two underlying principles. First, we do not assume as input a target motion that needs to be reproduced (cf. [Skouras et al. 2013]). This decision removes the need for hard-to-quantify perceptual similarity measures or unintuitive Cartesian or strain-space error metrics. Instead, users are presented with a familiar posing and keyframing system that allows plush toy motions to be interactively created. The posing system employs a novel inverse kinematics-like solver that operates in the space of deformations induced by activations of cable-based *contractile elements*. Consequently, all motions authored by the user are guaranteed to be realizable using the type of actuation that our plush robots will employ.

As a second defining characteristic of our work, a plush toy's actuation system is designed *while* the user is creating its motions. Our approach is inspired by muscular hydrostats – biological structures composed of muscles but lacking any rigid skeletal support. In nature, these systems have proven to be extremely useful and versatile. Elephant trunks, for example, are wonderfully dexterous; their amazing motion repertoire is due to hundreds of thousands of muscle fiber bundles. Unfortunately, this actuation complexity is clearly well beyond current fabrication capabilities. Simulations, however, are not bound to fabrication constraints. A key insight is to therefore begin by instrumenting the physically-simulated body of a plush toy with a large number of virtual muscles, which we call *muscle-fibers*. Mimicking biological systems, each muscle-fiber is small and capable of contracting independently. While not physically realizable, this dense arrangement of muscle-fibers gives rise to the richest space of motions possible. During the motion authoring process, our computational method analyzes the muscle-fibers' co-activation patterns; those with large activation levels contribute most to the user-specified motions are merged into ready-to-fabricate *winch-tendon networks*. Winch-tendon networks are used to model the motorized devices and cables which, when embedded in the soft bodies of physical plush robots, will be driving their motions.

We demonstrate the versatility of our computational approach by creating a variety of animated plush toys. Since the authoring process is interactive, a key benefit of our method is that it enables an intuitive exploration of the design space. Consequently, users can find a balance between the intricacies of their plush toy's motions and the complexity of its design. To validate our results, we employ simulations and fabricate several physical prototypes.

2 RELATED WORK

Our work is inspired by the considerable attention the research community has given to the development of computational tools for designing physical surfaces. Recent investigations, for example, represent surfaces using paper-craft models [Kilian et al. 2008; Mitani and Suzuki 2004] and paper pop-ups [Li et al. 2011, 2010], inflatable structures [Skouras et al. 2012, 2014], wire-mesh sculptures [Garg et al. 2014], flexible rod and curve networks [Pérez et al. 2015; Zehnder et al. 2016], procedurally-generated filigrees [Chen et al. 2016], tensegrities [Gauge et al. 2014], inextensible sheets with cut patterns that lead to auxetic behavior [Konaković et al. 2016], modular interlocking [Skouras et al. 2015] or self-supporting [Song et al. 2013; Vouga et al. 2012] primitives, and, closest to our work, plush toys [Igarashi and Igarashi 2008; Mori and Igarashi 2007]. While these efforts are aimed at creating static depictions of shape, the goal of our work is to create physical models that are capable of producing purposeful animations.

The challenge of creating physical objects that are specifically designed to produce interesting motions is also fueling an active area of research. Design methods for articulated characters [Bächer et al. 2012; Cali et al. 2012], mechanical automata [Ceylan et al. 2013; Coros et al. 2013; Thomaszewski et al. 2014; Zhu et al. 2012], and even walking [Megaro et al. 2015] and flying [Du et al. 2016] robots have been recently proposed. While these works study mechanical structures composed primarily of rigid body parts, the method we propose considers soft physical systems that undergo large deformations. Consequently, our work is most closely related to the methods introduced by Bickel et al. [2012] and Skouras et al. [2013]. The former employs shape optimization techniques to control the deformations of a silicone skin driven by an underlying animatronic system. The latter optimizes a distribution of soft and rigid materials, as well as external actuation forces, so that the deformations of elastic characters match a set of input poses. In contrast to these methods, our approach lets users author the motions of soft characters through an intuitive posing interface that automatically computes contractions for embedded muscle-like elements. While the animation is being authored, our design system also creates an internal actuation system that will drive the motions of real-life plush toy prototypes.

The formulation we propose is inspired by control methods developed to animate soft virtual characters [Coros et al. 2012; Tan et al. 2012]. These methods develop optimization solutions that couple implicit time stepping with computation of control inputs. However, solving for deformed configurations, control signals and contact forces at the same time leads to challenging constrained optimization programs. Our solution, in contrast, is specifically tailored for soft and light plush toys driven by muscle-like actuators. Under the assumption of quasi-static motions, an implicit relationship between deformed configurations and actuation inputs can be established. By exploiting this relationship, we develop an efficient numerical solver to compute muscle activations that lead to desirable plush toy motions. It is worth noting that while the method developed by Tan et al. [2012] also uses strand-based actuation, physical constraints require us to make different modeling choices. In particular, our actuating elements can be of arbitrary length, they generate pull-only forces, and slide freely through a series of waypoints in order to model the behavior of cables routed through the soft bodies of plush toys [Yamashita et al. 2012]. As another key difference, our approach is used to co-design motions along with the actuation systems necessary to recreate them in a physical prototype. The output of our computational design system consists of fabrication blueprints for customized plush robots.

3 DESIGN PROCESS OVERVIEW

In this section we describe the computational approach we use to create animated plush robots, and introduce the technical concepts that form the core of our design methodology.

Input. Our design system takes as input a geometric mesh depicting the shape of the desired plush robot. For this work, we restrict

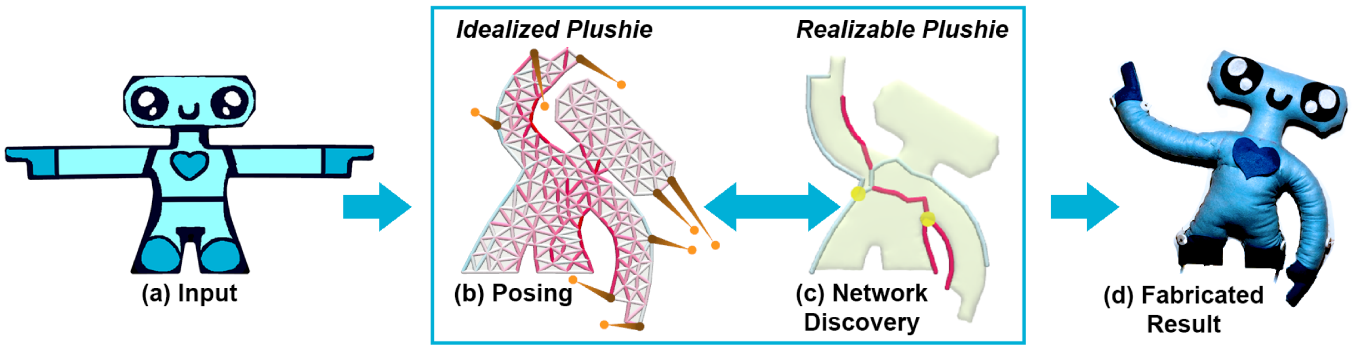


Fig. 2. Overview of our computational approach to creating animated plushies. A simulation model of an input plush robot (a) is animated using our Soft IK posing system. Based on target locations for user-selected nodes, the Soft IK solver computes muscle-fiber activations to optimally deform the idealized plushie (b). By analyzing the pattern of muscle-fiber co-activations, our design system generates physically-realizable winch-tendon networks (c). Winches are visualized as yellow circles. The output of our system directly informs the placement of actuators, the routing of cables throughout the body of the fabricated prototypes, and the control signals required to reproduce the user-created motions (d).

our attention to designing planar motions, but note that the concepts we develop are general. The plushie is visualized as a 2.5D, optionally textured model, as illustrated in Figure 2 (a).

Modeling plush toys. We model the elastic behavior of plush toys using a standard finite-element analysis approach. As detailed in Section 4.1, we expect plushies to undergo large deformations, which are assumed to be quasi-static, during the motion authoring process. We therefore employ a non-linear Neo Hookean material model, but assume constant strain over each element of the simulation mesh in favor of computational efficiency. We note that the predictive power of this elastic model depends on the method used to fabricate the plush toys. The type of stuffing that is used and how tightly packed it is, whether it clumps or not, how uniformly it is distributed throughout the plushie, and the material characteristics of the fabric layer all play a role in the plushie's deformation behavior. Our experiments show that the FEM model we use captures well the low-frequency, most salient features of the plushie's motions. It is also worth mentioning that our computational methodology is very general and would directly work with other material models as long as they provide analytic derivatives of their strain energy.

Graphical Design Interface. Users are presented with two views of their design, as shown in Figure 2 (b) and (c). For the view on the left, the plush toy is equipped with a large number of muscle-like virtual *muscle-fibers*. This model, which we call the *idealized plushie*, is capable of a rich space of motions that users can freely explore during the animation authoring process. The view on the right shows the plush robot instrumented with physically-realizable *winch-tendon networks*. Its motions are in close correspondence to those of fabricated plush robot prototypes. We therefore refer to this model as the *realizable plushie*. As the main means for authoring motions, we employ a *posing system* that is conceptually related to traditional character animation techniques based on inverse kinematics, as shown in Figure 2 (b).

Muscle-fibers and winch-tendon networks. Muscle-fibers and tendons are the two types of *contractile elements* used in our design system. They are modeled as unilateral springs that can change their rest length in order to actively modulate the tension forces they generate, as described in Section 4.2. The internal forces that their contractions give rise to deform the soft body of the plush toys. Inspired by biological muscular hydrostats, muscle-fibers are small, independently activated, and instantiated in large numbers. Although different alternatives can easily be considered, our design system creates a muscle-fiber for every edge of the simulation mesh that is used to model plush toys. Muscle-fibers are highly versatile and can produce intricate motions. Unfortunately, they lead to actuation systems that are far too complex to fabricate. Rather than relying on them to animate the final plush toy prototypes, therefore, we use them to guide the design of physically-realizable winch-tendon networks.

Winch-tendon networks (Figure 3, left) model the physical mechanisms we employ to animate our fabricated prototypes (Figure 3, right). Each winch is driven by a brushed DC motor and winds an arbitrary number of cables, which we call tendons, onto the same 3D-printed spool. As a winch is activated, it therefore shortens all tendons attached to it by the same amount. The motion capabilities of a physical plush robot are therefore governed by 1) the number of winch-tendon networks that are employed, 2) the number of tendons connected to each winch, and 3) the path that each tendon takes as it is routed through the soft body of the plushie. Our computational method provides a systematic approach to creating winch-tendon networks that are custom-designed based on the motions envisioned for the plush toy.

Posing system. Our design system allows users to author motions for their plush robot through an intuitive posing system. As visualized in Figure 2 (b), the user can select any point on the body of the plushie and drag it to a desired location in the scene. This mode of operation is inspired by traditional inverse-kinematics techniques. However, rather than operating on joint angles and assuming an underlying rigidly articulated skeleton, our numerical solver computes activations for each available contractile element such that optimal quasi-static deformations of the soft robot are generated. In the resulting deformed configuration, the points selected on the body of the plushie are as close to their target location as the contractile

nature of its muscle-like actuators will allow. The technical details of this *Soft IK* posing system are presented in Section 5.

Design Loop. A key insight is that physically-realizable actuation systems can be incrementally designed while users are authoring the motions they desire for their plush toy. The iterative design process begins with an empty set $\mathcal{N}$ of winch-tendon networks. At every iteration, the idealized plushie is rigged with the full set of virtual muscle-fibers, as well as with the contractile elements stored in $\mathcal{N}$. The realizable plushie, in contrast, is only equipped with the current set of winch-tendon networks. The user selects a number of points on the body of the plush toy and drags them to different locations in the scene. Based on this input, our Soft IK posing system is applied in an interleaved fashion.

We begin by employing the Soft IK solver on the realizable plushie design. The resulting deformed configuration represents the best pose that can be achieved with the current set of physically realizable actuators. The activations of the contractile elements in $\mathcal{N}$ are then transferred over to the idealized plushie. Keeping these activations fixed, the Soft IK solver then computes contractions for each individual muscle-fiber in order to generate the best possible pose given the user input. Intuitively, the difference in the poses assumed by the two plushies provides a direct measure of how much more the final design could be improved. If the user is unhappy with the pose reached by the realizable plushie, they initiate the *network discovery* process, which adds a new winch-tendon network to $\mathcal{N}$.

As the posing and network discovery steps repeat, new winch-tendon networks are incrementally added to the design. This iterative design process terminates once the user finds the right balance between the ability of the plushie to produce nuanced motions and the complexity of its design. It is worth noting that our alternating application of the Soft IK solver ensures that the physically-realizable actuation system is used to its full potential before any muscle-fibers contribute to the resulting deformations. Muscle-fibers are therefore solely used to account for deficiencies in the plushie's current design. Importantly, we achieve this effect without needing to resort to sparse regularization terms or to potentially hard-to-tune objectives that bias the use of one type of contractile element over the other.

Network discovery. Throughout the posing process, our design system analyzes patterns of co-activation for the muscle-fibers that contribute to the deformations of the idealized plushie. In particular, active muscle-fibers with similar contraction patterns are merged into candidate winch-tendon networks, as outlined in section 6. These networks are presented to the user, who chooses whether to integrate them into the realizable plushie in order to enrich its motion repertoire.

4 SYSTEM MODEL

To capture the deformation behavior of soft plush toys, we represent them using a finite element model. The contractile elements that animate plushies are modeled as stiff unilateral springs with variable rest length. Adopting standard notation, we refer to the deformed, statically stable (e.g. minimum energy state) configuration of the plush model as $\boldsymbol{x}$, which is a vector storing the coordinates of all

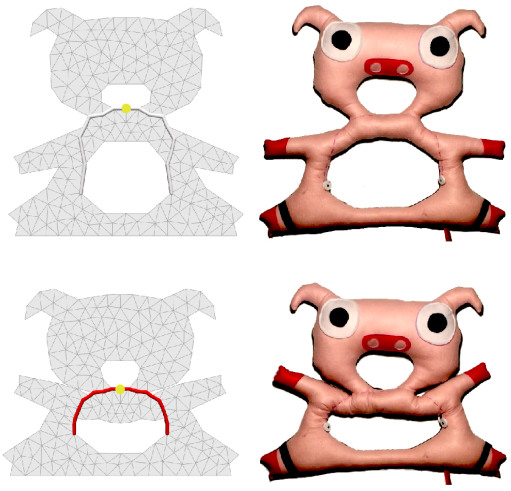


Fig. 3. The "pig" example shown in simulation and fabricated.

nodes in the simulation mesh. Similarly, we let $\boldsymbol{X}$ denote the rest, or undeformed configuration of the simulation mesh.

The total deformation energy of the system is defined as:

$$E = E_{\text{plush}} + E_{\text{contractile}} + E_{\text{pins}} \tag{1}$$

where E_{plush} is the energy due to deformations of the simulation mesh, $E_{\text{contractile}}$ is the strain energy stored by the contractile elements, and E_{pins} models the behavior of stiff springs that connect a small number of simulation nodes to world anchors in order to eliminate rigid body modes.

The gradient of this energy with respect to the nodal degrees of freedom gives us the forces acting on each node of the simulation mesh:

$$\boldsymbol{F} = -\frac{\partial E}{\partial \boldsymbol{x}} \tag{2}$$

and the Hessian of this energy outputs the force Jacobian

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}} = -\frac{\partial^2 E}{\partial^2 \boldsymbol{x}} \tag{3}$$

Likewise, differentiating the individual terms of the system energy with respect to the nodal degrees of freedom allows us to separate the forces generated by mesh deformations from those due to the contractile elements.

$$\boldsymbol{F} = \boldsymbol{F}_{\text{plush}} + \boldsymbol{F}_{\text{contractile}} + \boldsymbol{F}_{\text{pins}} \tag{4}$$

4.1 Simulation Model for Plushies

We model the elastic behavior of plush toys using linear finite elements with a non-linear material model. For each element e of the simulation mesh, we first compute the deformation gradient as $\mathcal{F} = \partial \boldsymbol{x}^e / \partial \boldsymbol{X}^e = \boldsymbol{d}\boldsymbol{D}^{-1}$, where $\boldsymbol{d}$ is a matrix whose columns store edge vectors: $\boldsymbol{d}_i = \boldsymbol{x}_i^e - \boldsymbol{x}_0^e$ with $\boldsymbol{x}_j^e$ denoting the world coordinates of the j-th node of element e. The matrix $\boldsymbol{D}$ is similarly defined using

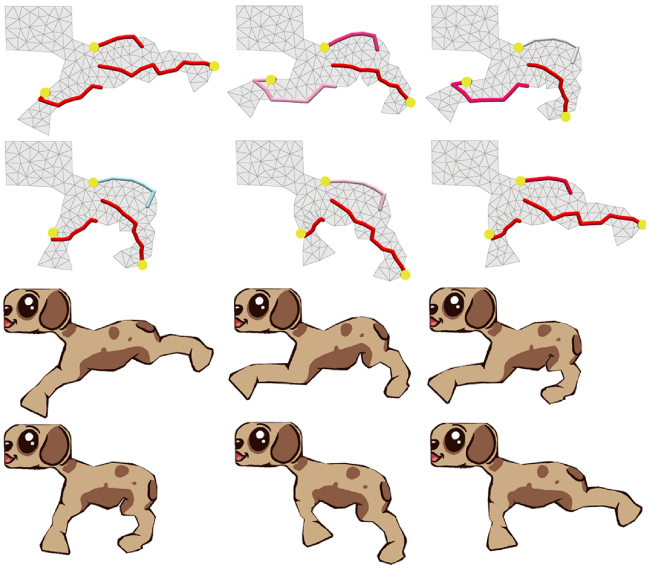
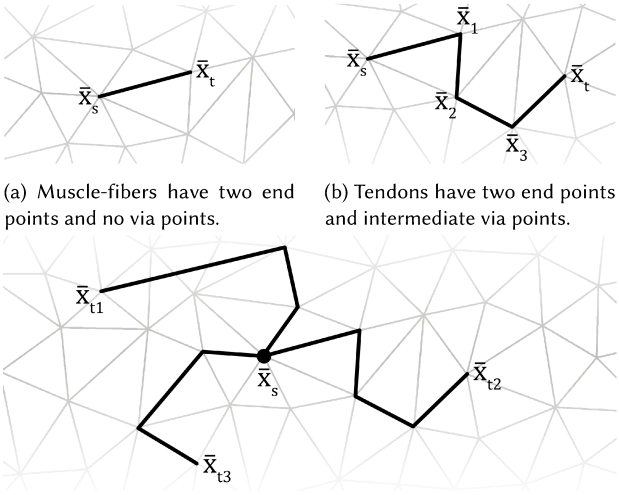


Fig. 4. The "puppy" example in simulation, performing a running motion.



(a) Muscle-fibers have two end points and no via points.

(b) Tendons have two end points and intermediate via points.

(c) A winch-tendon network contains an arbitrary number of tendons, sharing a common attachment point.

Fig. 5. Examples of the two contractile elements used in this work–muscle-fibers and tendons–as well as an example of a winch-tendon network.

rest configuration quantities. The energy density of the element is then defined using a compressible Neo-Hookean material model:

$$\Psi(\boldsymbol{x}, \boldsymbol{X}) = \frac{\mu}{2}\mathrm{tr}(\mathcal{F}^T\mathcal{F} - I) - \mu \ln J + \frac{\kappa}{2}(\ln J)^2 \tag{5}$$

where μ and κ are material parameters, I is the identity matrix and $J = \det(\mathcal{F})$. The elastic energy stored by the element is obtained by integrating (5) over its domain, which is trivial given the assumption that deformation gradients are constant across each element. The energy term E_{plush} is obtained by summing up the individual contributions of all elements in the simulation mesh.

We note that the computational design method we propose is material model-agnostic. As long as analytic formulations for the deformation energy and its derivatives exist, any other simulation model can be employed in a plug-and-play fashion.

4.2 Contractile elements

We model muscle-fibers and tendons using an abstraction we call a *contractile element*. This abstraction is built around the concept that we can model the contraction of a muscle-fiber or tendon by changing the rest length of its underlying unilateral spring model.

We define a contractile element as a piecewise linear curve with two endpoints $\bar{x}_s, \bar{x}_t$, which we call *attachment points* and n intermediate vertices $(\bar{x}_1, ..., \bar{x}_n)$, which we call *via points*. A muscle-fiber has no via points, whereas a tendon may have many, shown in Figure 5 (a) and (b). We use the word *routing* to refer to the choice (and order) of attachment points and via points. In this work we assume that attachment points and via points are bound to nodes of the mesh.

In order to model cables free to slide through (frictionless) via points, we consider the length $\ell = \ell(\boldsymbol{x})$ of the contractile element as the sum of distances between consecutive vertices.

$$\ell(\boldsymbol{x}) = \|\bar{x}_s - \bar{x}_1\| + \sum_{i=1}^{n-1}\|\bar{x}_i - \bar{x}_{i+1}\| + \|\bar{x}_n - \bar{x}_t\| \tag{6}$$

We model a *winch-tendon network* as a set of contractile elements with a common endpoint $\bar{x}_s$, shown in Figure 5 (c). This endpoint specifies where we position the winch in a physical assembly.

Formally, a contractile element consists of the following components.

(1) Attachment points $\bar{\boldsymbol{x}}_s, \bar{\boldsymbol{x}}_t$.
(2) Via points $(\bar{\boldsymbol{x}}_1, ..., \bar{\boldsymbol{x}}_n)$.
(3) An *initial rest length* α^0.
(4) A *contraction* α^c.
(5) A *rest length* $\alpha := \alpha^0 - \alpha^c$, defined the difference of the previous two quantities.
(6) A *deformation* $\Gamma = \Gamma(\boldsymbol{x})$, where $\Gamma(\boldsymbol{x}) := \ell(\boldsymbol{x}) - \alpha$.
(7) A *strain energy* $U = U(\Gamma)$, defined in the following section.

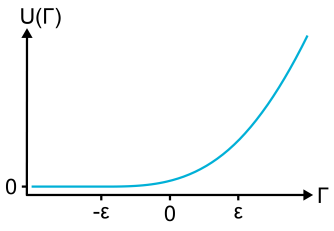
The derivative of strain energy with respect to deformation is the element's *tension*.

$$\tau = \frac{\partial U}{\partial \Gamma} \tag{7}$$

4.3 Unilateral strain energy

Both muscle-fibers and tendons make use of a unilateral strain energy defined to have the following properties.

(1) At least C^2 continuous so that second-order numerical solvers are well-behaved.
(2) Zero for $\Gamma < -\varepsilon$ to model the behavior we would expect from a physical muscle, cable, or tendon, which do not resist compression.
(3) A monotonically increasing quadratic for $\Gamma > \varepsilon$, to model resisting stretch according to Hooke's law.



The specific function we make use of is a piece-wise C^2 polynomial, which is zero below a negative threshold, quadratic above a positive threshold, and cubic in between. The parameter K determines the steepness of the force response, and parameter ε defines the domain of the interpolating cubic piece of the function. The coefficients of the cubic were chosen such that the function value and derivatives at match at $\Gamma = \pm\varepsilon$.

$$U(\Gamma) := \begin{cases} 0 & \Gamma \leq -\varepsilon \\ \frac{K}{6\varepsilon}\Gamma^3 + \frac{K}{2}\Gamma^2 + \frac{K\varepsilon}{2}\Gamma^x + \frac{K\varepsilon^2}{6} & -\varepsilon \leq \Gamma < \varepsilon \\ K\Gamma^2 + \frac{K\varepsilon^2}{3} & \text{otherwise} \end{cases} \tag{8}$$

4.4 Pins

To anchor the plushie in space, we allow users to attach *pins* to nodes, which we model as zero-length springs. The pins specified by the user can correspond in a physical prototype to e.g. hook and loop attachments to a base, or the users hand gripping the plushie. Each pin contributes $\frac{1}{2}K_{\text{pin}}(\boldsymbol{x}_{\text{pin}} - \boldsymbol{x}_i)^T(\boldsymbol{x}_{\text{pin}} - \boldsymbol{x}_i)$ to the pin energy term E_{pin}, where $\boldsymbol{x}_{\text{pin}}$ is the position of the pin in world coordinates, $\boldsymbol{x}_i$ is the position of the pinned vertex the deformed configuration, and K_{pin} is a large spring constant.

5 SOFT IK

The goal of the Soft IK procedure is to find a set of activations for each contractile element that deform the finite element mesh as close as possible to a target position $\boldsymbol{x}'$. The target position $\boldsymbol{x}'_i$ of any node in the mesh can be specified by the user via drag-and-drop. If no position is specified, the target position defaults to the position of node in the undeformed mesh, and that node's contribution to the objective greatly reduced.

5.1 Preliminaries

We stack the tensions of all contractile elements into a vector $\boldsymbol{\tau}$, and define the matrix $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{x})$ to satisfy the equation

$$\boldsymbol{F}_{\text{contractile}} = \boldsymbol{A}\boldsymbol{\tau} \tag{9}$$

as in [Tan et al. 2012]. The matrix $\boldsymbol{A}$ encodes the routing of each contractile element. This lets us write the partial derivative

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\tau}} = \boldsymbol{A} \tag{10}$$

since no other force terms depend explicitly on $\boldsymbol{\tau}$.

5.2 Objective function

Our main objective function is a weighted 2-norm penalizing distance from the target position as

$$O = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T\boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{x}') \tag{11}$$

where the diagonal matrix $\boldsymbol{Q}$ masks out the contributions of nodes for which the user has not specified a target position.

5.3 Gradient

Our goal is to minimize the objective O, a function of the force equilibrium configuration $\boldsymbol{x}$ that is affected by contractions $\boldsymbol{\alpha^c}$. In this section we will establish the relationship between $\boldsymbol{x}$ and $\boldsymbol{\alpha^c}$, and compute the gradient $\frac{\partial O}{\partial \boldsymbol{\alpha^c}}$. To begin with, we expand $\frac{\partial O}{\partial \boldsymbol{\alpha^c}}$ by considering how O changes with x.

$$\frac{\partial O}{\partial \boldsymbol{\alpha^c}} = \frac{\partial O}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha^c}} \tag{12}$$

The Jacobian $\frac{\partial O}{\partial \boldsymbol{x}}$ is readily computed.

$$\frac{\partial O}{\partial \boldsymbol{x}} = \boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{x}') \tag{13}$$

The Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha^c}}$ captures changes in the force equilibrium configuration induced by changes in the contractions of actuating elements. Since these contractile elements contribute to the net nodal forces $\boldsymbol{F}(\boldsymbol{x})$ through the tensions $\boldsymbol{\tau}$ they generate, we expand the Jacobian above by considering how $\boldsymbol{x}$ changes with $\boldsymbol{\tau}$:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha^c}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}}\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\alpha^c}} \tag{14}$$

The Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}}$ relates changes in the plushie's shape to changes in the tensions generated by the contractile elements. Because an analytic formula for the relationship $\boldsymbol{x}(\boldsymbol{\tau})$ does not exist, the Jacobian we seek cannot be computed directly. However, due to the quasi-static assumption we make, we know that a change in $\boldsymbol{\tau}$ must induce a change in $\boldsymbol{x}$ such that force equilibrium is maintained. Mathematically, this means that the total derivative of the net nodal forces with respect to tensions vanishes:

$$\frac{\mathrm{d}\boldsymbol{F}}{\mathrm{d}\boldsymbol{\tau}} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\tau}} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}} = 0 \tag{15}$$

We have already derived analytic expressions for the Jacobians $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\tau}}$ and $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}$ in eqs. (3) and (10). Substituting those into eq. (15) gives:

$$\boldsymbol{A} = \frac{\partial^2 E}{\partial^2 \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}} \tag{16}$$

which we can solve for $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}}$. In our implementation this is accomplished with a sparse Cholesky solver.

The Jacobian $\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\alpha^c}}$ can be expanded by recalling our expression for tension in eq. (7) is written purely in terms of the deformation $\boldsymbol{\Gamma}$.

$$\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\alpha^c}} = \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\Gamma}}\frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{\alpha^c}} \tag{17}$$

The Jacobian $\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\Gamma}}$ is a diagonal matrix storing the second derivative of strain energy of each contractile element.

$$\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\Gamma}} = \frac{\partial^2 \boldsymbol{U}}{\partial^2 \boldsymbol{\Gamma}} \tag{18}$$

The Jacobian $\frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{\alpha^c}}$ relates changes in deformation to changes in contraction. For any contractile element we can expand the definition of contraction as $\Gamma = \ell - (\alpha^0 - \alpha^c)$ implying that this term is just the identity.

$$\frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{\alpha^c}} = \boldsymbol{I} \tag{19}$$

Putting everything together, we have the gradient expanded below for reference.

$$\frac{\partial O}{\partial \boldsymbol{\alpha^c}} = \frac{\partial O}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\tau}} \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\Gamma}} \frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{\alpha^c}} \tag{20}$$

5.4 Incorporating actuation constraints

As written, the gradient in eq. (20) could be used as part of a constrained optimization, with two sets of constraints, once to enforce an upper bound on contraction $\boldsymbol{\alpha^c} \leq \boldsymbol{\alpha}^{\max}$, and the other to model *winch-tendon networks*.

A winch-tendon network is considered to be a set of tendons with a common endpoint. In fabrication, the *winch* is positioned at the common endpoint. The activation of the winch contracts all the tendons in a winch-tendon network, and tendons provide passive coupling to the body of the plushie.

In order to say that tendon T_i is a member of winch W_j we write $T_i \in W_j$. The *winch constraint* ensures that all tendons in a given winch-tendon network share the same contraction.

$$\alpha_h^c = \alpha_i^c \text{ for all } T_h, T_i \in W_j \tag{21}$$

In the following two sections we provide reparametrizations that will take the place of these two sets of constraints, and enable the use of an unconstrained optimization method (presented in section 5.7).

5.5 Winch synergy

In order to avoid having to explicitly enforce the winch constraint from section 5.4, we introduce a new variable β_j to be the *activation* of each winch-tendon network, and reparametrize tendon contractions in terms of activations.

We stack activations into a vector $\boldsymbol{\beta}$, and define a matrix $\boldsymbol{Z}$

$$Z_{ij} = \begin{cases} 1 & \text{if } T_i \in W_j \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

to map from winch activations to tendon contractions.

$$\boldsymbol{\alpha^c} = \boldsymbol{Z\beta} \tag{23}$$

With this definition for $\boldsymbol{Z}$, it follows that the contraction of every tendon in a given winch-tendon network is equal to the activation of the winch in that network. This models the action of a physical winch, which takes in or lets out the same amount of cable for each tendon connected to it.

We then have the Jacobian

$$\frac{\partial \boldsymbol{\alpha^c}}{\partial \boldsymbol{\beta}} = \boldsymbol{Z} \tag{24}$$

and the overall gradient with respect to $\boldsymbol{\beta}$.

$$\frac{\partial O}{\partial \boldsymbol{\beta}} = \frac{\partial O}{\partial \boldsymbol{\alpha^c}} \frac{\partial \boldsymbol{\alpha^c}}{\partial \boldsymbol{\beta}} \tag{25}$$

To have a unified treatment of both winch-tendon networks and muscle-fibers, we consider each muscle-fiber as having its own activation $\beta := \alpha^c$, in which case $\boldsymbol{Z}$ is just the identity. In this way, a muscle-fiber can be seen as a winch-tendon network with exactly one tendon.

5.6 Actuation limits

To avoid having to explicitly use constraints to enforce the upper bound on contractions in section 5.4, we define the *maximum activation* $\beta^{\max}$ of each muscle-fiber or winch-tendon network, and reparametrize activations using new variables $\boldsymbol{\xi}$.

The maximum activation is the largest activation that does not violate the constraint $\boldsymbol{\alpha^c} \leq \boldsymbol{\alpha}^{\max}$. The maximum activation of a muscle-fiber is equal to that fiber's maximum contraction (recall in the previous section we defined the activation of a muscle fiber equal to its contraction).

$$\beta_i^{\max} = \alpha_i^{\max} \tag{26}$$

The maximum activation of a winch-tendon network W_j, is the smallest maximum contraction of the network's tendons.

$$\beta_j^{\max} = \min_{T_i \in W_j} \alpha_i^{\max} \tag{27}$$

For each contractile element we can define a function f_i that maps from the optimization domain (the real line) to the allowed activation range.

$$f_i : \mathbb{R} \to (-\infty, \beta_i^{\max}) \tag{28}$$

We define a corresponding vector function $\boldsymbol{f}$ such that $\boldsymbol{\beta}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi})$.

$$\boldsymbol{f}(\boldsymbol{\xi}) := (f_1(\xi_1), \ldots)^T \tag{29}$$

We note that in our implementation, each f_i is actually the same function we use for unilateral strain energy in section 4.3, just reflected over the x-axis and translated up by $\beta^{\max}$. While $\boldsymbol{\xi}$ is unconstrained, $\boldsymbol{f}(\boldsymbol{\xi})$ is bounded above by $\boldsymbol{\beta}^{\max}$.

We then immediately have the Jacobian

$$\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\xi}} = \boldsymbol{f}'(\boldsymbol{\xi}) \tag{30}$$

and gradient of the objective with respect to $\boldsymbol{\xi}$.

$$\frac{\partial O}{\partial \boldsymbol{\xi}} = \frac{\partial O}{\partial \boldsymbol{\beta}} \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\xi}} \tag{31}$$

It is worth noting that now that we have established a relationship between deformed configuration and the driving forces, it is trivial to add a regularizer on tension to prevent slack from building up in the system.

5.7 The Soft IK Solver

Recall that we began with our objective O which is defined in terms of the deformed mesh position $\boldsymbol{x}$. We introduced optimization variables $\boldsymbol{\xi}$, any choice of which automatically adheres to the actuation limits of all contractile elements, as well as the winch constraint. Finally we established the gradient $\frac{\partial O}{\partial \boldsymbol{\xi}}$, which describes how our objective changes as we vary $\boldsymbol{\xi}$. We are now ready to present the details of our optimization procedure.

We can write a statically-stable configuration of a plushie, which we call a *pose*, as $P = (\boldsymbol{x}, \boldsymbol{\xi})$. By this we mean that if we find contractions $\boldsymbol{\alpha^c}(\boldsymbol{\xi})$, and propagate them into the rest lengths of the corresponding contractile elements, we will find that the statically stable position of the mesh is $\boldsymbol{x}$.

Our central algorithm, which we call *Soft IK* is based on gradient descent. Pseudocode is provided in Algorithm 1. One iteration of

ALGORITHM 1: Soft IK

Input: Statically stable P_a
Output: Statically stable P_b
compute $\frac{\partial O}{\partial \xi}_a$ according to eq. (31)
compute γ via efficient line search
$\xi_b \leftarrow \xi_a + \gamma \frac{\partial O}{\partial \xi}_a$
$\boldsymbol{\beta}_b \leftarrow f(\xi_b)$
$\boldsymbol{\alpha}_b^c \leftarrow Z\,\boldsymbol{\beta}_b$
$\boldsymbol{\alpha}_b \leftarrow \boldsymbol{\alpha}_0 - \boldsymbol{\alpha}_b^c$
solve statics for $\boldsymbol{x}_b$

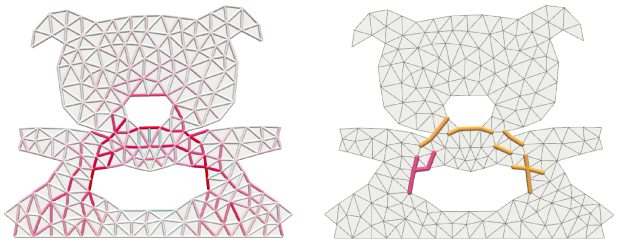
Soft IK takes us from a statically stable pose P_a to another statically stable pose P_b, which is closer to the user-specified target position.

We note that a naive line search would require a statics solve for $\boldsymbol{x}$ every time we wanted to evaluate the objective function, which would be computationally expensive. To avoid this we make use of the fact that we can easily compute $\frac{\partial \boldsymbol{x}}{\partial \xi}$ using the last four terms of the gradient of our objective function, and linearly approximate $\boldsymbol{x}$.

$$\boldsymbol{x}(\xi + \Delta\xi) \approx \boldsymbol{x}(\xi) + \frac{\partial \boldsymbol{x}}{\partial \xi}\Delta\xi \tag{32}$$

This yields a computationally efficient line search criterion to estimate the step size γ.

One final detail is that we have yet to define the initial rest lengths of any contractile elements. In our implementation we make the convenient choice to set the initial rest lengths of each contractile element equal to its length measured in the mesh's *undeformed pose*, i.e. $\boldsymbol{\alpha}^{\boldsymbol{c}} := \boldsymbol{\ell}^{\boldsymbol{X}}$. This means that in our physical prototypes all tendons will be assembled with zero slack (though with extra cable wound around the spool). Other choices for $\boldsymbol{\alpha}^{\boldsymbol{c}}$ are certainly possible, but we leave a full exploration of this topic for future work.



(a) Activation graph, with active fibers highlighted in red.

(b) Thresholded fibers, with starting component in red.

(c) Winch tree, with joining paths in blue.

(d) Output network, with winch in yellow and tendons in green.

Fig. 6. Overview of network discovery.

6 NETWORK DISCOVERY

Posing the idealized plushie using Soft IK yields activations for its many virtual muscle-fibers. The job of network discovery is to find winch-tendon networks that enable the realizable plushie to approximate the poses specified by the user. We explain the algorithm here, and provide pseudocode with full details in Algorithm 2.

To formalize network discovery, it is convenient to consider the muscle-fibers of the idealized plushie as an edge-weighted undirected graph G, which we call the *activation graph*, illustrated in Figure 6 (a). The vertex set $V(G)$ has one vertex corresponding to each node in the plushie's underlying finite element mesh, and the edge set E has one edge corresponding to each muscle-fiber (recall that muscle-fibers connect exactly two nodes in the finite element mesh). For each edge $e \in E(G)$ we associate a weight $w(e)$ equal to the contraction of the corresponding muscle fiber. These weights will form the basis of the subsequent thresholding procedure, winch placement, and tendon selection.

The thresholding procedure simply selects the most active muscle-fibers in the idealized plushie. The number of fibers to threshold is set by a parameter SEED_NUM. The result of this thresholding is a set of connected components in the winch graph, illustrated in Figure 6 (b). We will designate the connected component with largest total tension as the *starting component*, or S in the pseudocode.

We then join the starting component to other, nearby connected components in the winch graph. This will form a structure we call the *winch tree*, illustrated in Figure 6 (c). The joining procedure is done iteratively. Connected components sufficiently close to the winch tree are added to it (along with a connecting path for each component), and then the procedure is repeated. The maximum allowed distance for a component to be added is set by a parameter JOIN_LEN, and the number of times the overall procedure is repeated is set by a parameter JOIN_DEPTH.

Finally, network discovery chooses both which vertex in the winch tree will correspond to the position of the winch, as well as which set of candidate tendons to include in the final network, as shown in Figure 6 (d). The number of tendons in the final network is set by a parameter TENDON_NUM. These selections are made according to a simple heuristic. This heuristic makes use of a quantity which we call the *net contraction*. For a candidate tendon, the net contraction is defined to be the sum of contractions over all muscle-fibers the tendon passes through. The net contraction estimates the candidate tendon's contraction if it were to be incorporated into the realizable plushie. Intuitively, the heuristic we developed rewards large net contractions for each tendon. However, it also penalizes large variance in net contraction between candidate tendons, thus discouraging the emergence of winch-tendon networks that poorly capture co-activation patterns (recall that contraction is shared for all tendons in a given winch-tendon network). The trade-off between these two sub-goals is determined through a user-tunable parameter $\phi \in [0, 1]$.

7 FABRICATION

It is worth noting that the physical prototypes are surprisingly easy to create. Each prototype can be completely fabricated and assembled in the span of a few hours, not counting the 3D printing of the

ALGORITHM 2: Network discovery

Input: Activation graph G
Output: Winch position $r*$ and tendon paths $Q*$
Parameters: SEED_NUM, JOIN_DEPTH, JOIN_LEN, TENDON_NUM, ϕ
`/* 0: Preliminary computation. */`
$E' \leftarrow$ SEED_NUM heaviest edges in E
identify connected components of E'
for each connected component, break cycles by deleting lightest edges
$S \leftarrow$ connected component with largest total contraction
$\mathcal{T} = \{T_1, \ldots\} \leftarrow$ other connected components
`/* 1: Build the winch tree` G_W`. */`
$G_W \leftarrow S$
for $d = 1, \ldots,$ *JOIN_DEPTH* **do**
  $G_{\text{temp}} \leftarrow$ null graph
  for $T_i \in \mathcal{T}$ **do**
    find shortest path p_i from $V(G_W)$ to $V(T_i)$ in G
    if *number edges in* p_i *no more than JOIN_LEN* **then**
      add (graph union) T_i, p_i to G_{temp}
      remove T_i from $\mathcal{T}$
    end
  end
  add (graph union) G_{temp} to G_W
end
`/* 2: Select winch position` r^* `and tendon set` Q^*`. */`
initialize table rootScore[] `// Best score for each candidate root.`
initialize table Q'[] `// Tendon sets corresp. to that best score.`
for *each candidate root* $r \in V(G_W)$ **do**
  initialize table winchScore[]
  find candidate tendons by traversing back from leaves of G_W to r
  for *each candidate tendon* q **do**
    netContraction[q] $\leftarrow \sum_{e \in q} w(e)$
  end
  for *each set* Q *of TENDON_NUM non-overlapping candidate tendons* **do**
    $W \leftarrow \{\text{netContraction}[q] \mid q \in Q\}$
    winchScore[Q] = ϕMean$W$ − $(1-\phi)$Var$W$
  end
  $Q'[r] \leftarrow \arg\max_Q$ winchScore[Q]
  rootScore[r] $\leftarrow$ winchScore[$Q'[r]$]
end
$r^* \leftarrow \arg\max_r$ rootScore[r]
$Q^* \leftarrow Q'[r^*]$

winch. We laser cut low-cost acrylic felt sheets that are then sewn together. We use standard polyester to fill the resulting structures, and employ monofilament cables for actuation. To create the plush toy's motions, the cables can either be pulled by hand (for simple motions), or contracted by a motorized winch. We tape or hold down part of the plushie to reproduce the effect of the simulated pins.

7.1 Materials

Skin. We generate cut pattens which include small holes for tendon attachment points and via points. Using a later cutter, we cut out these patterns from acrylic craft felt. The resulting pieces of felt are sewn together on a standard sewing machine to serve as the skin of our plushie.

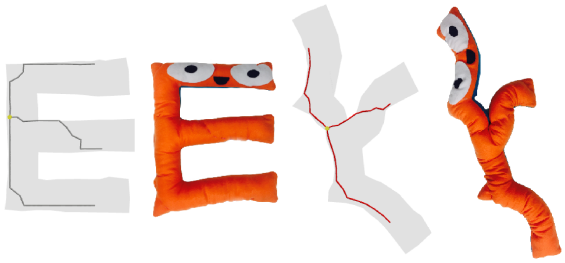


Fig. 7. The "E" example shown in simulation and fabricated.

Tendons. For tendons we use 0.50mm braided or monofilmanet fishing line. Either choice will work, and braided line was found to be easier to route. To route the tendon through the skin we use of large-eyed blunt needles.

Stuffing. We invert the sewn skin to ensure a smooth visual appearance, and stuff it with standard polyester fiber filling.

7.2 Winch design

Each winch is actuated by a FAULHABER® brushed DC gearmotor, driven by a corresponding motion controller. The motion controllers and power supply are external to the animated plushie.

We design a custom 3D-printed spool and housing that can handle a large number of cables. For the 2D plushies we fabricate, we rig two copies of the tendon routing discovered by our system, one on the front face of the plushie, and one on the back.

8 RESULTS

We used our design system to create six animated plushies, shown in Figures 1, 3, 4, 7, 10 and 11. We validated four of these designs through physically fabricated prototypes. Table 1 summarizes statistics for our designs, and the animated plushies are presented in the accompanying video. For all the examples presented in this paper, the Soft IK and network discovery algorithms ran in realtime, enabling an intuitive interactive design process.

Table 1. Examples, with statistics on the number of muscle-fibers in the idealized plushie, the number of target nodes used in the posing session (presented as a range for examples with multiple poses), the total number of winch-tendon networks and tendons in the realizable plushie, and whether or not it was fabricated.

	fibers	targets	winches	tendons	fabricated
robot	414	8-11	2	6	✓
pig	571	2	1	2	✓
E	442	6	1	3	✓
S	276	4	1	2	✓
puppy	307	3-6	3	3	
squid	429	4-8	8	11	

8.1 Comparing fabricated results against simulation

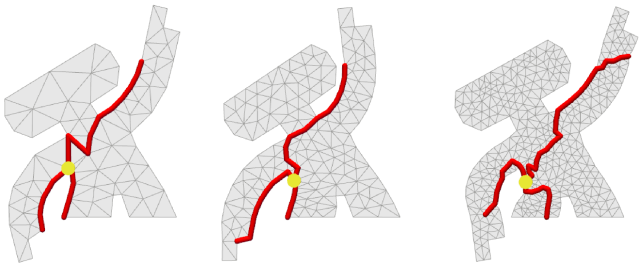
The large-scale deformations of our fabricated prototypes match those of the corresponding simulated plushies, as shown in Figures 1,

3, 7 and 10. Nevertheless, differences between our simulations and the fabricated results do exist. For example, the physical plushies do not always perfectly return to the rest pose when tension is released. Several factors contribute to this mismatch in deformation behavior. First, we do not model friction between cables and the skin of the plushies, nor friction between the plushies and the surfaces they lie on. Second, polyester filling can clump if compressed too much. These problems can be alleviated by having the design process be informed by higher fidelity simulation models. As another avenue for circumventing problems due to modeling approximations, we are interested in exploring different materials (low-friction cables and fabric skins, foam-based stuffing, etc.) for our physical prototypes.

8.2 Scalability

For all our designs, the idealized plushies have one muscle fiber for every edge in the simulation mesh. On a standard laptop, the SoftIK posing system runs in real time for plushies with up to 500 muscle fibers. As the number of muscle fibers nears 1000, the posing system is still usable, though it becomes less responsive. Nevertheless, we note that our model formulation supports high-resolution simulation meshes rigged with a coarser set of muscle fibers.

To analyze the effect of the input discretization on the resulting designs, we tested the ability of our system to reproduce the same pose starting from different resolutions of the simulation mesh. As shown in Figure 8, these designs are rigged with 177, 414, and 1055 muscle-fibers, respectively. We found that the output winch-tendon networks and poses are largely similar, though we observed some artifacts arising due to the tessellation of the simulation mesh. For the coarser discretization, the muscle-fibers to choose from are relatively long, and lead to a noticeable kink in the tendons of the output network. For the finer discretization, multiple, largely parallel groups of muscle fibers activate at similar levels, as illustrated in Figure 9. This behavior stresses our simple winch discovery algorithm, leading to non-smooth tendon routings that attempt to connect the muscle fibers that operate in parallel. To address these artifacts, we plan to develop regularizing terms that explicitly promote the emergence of smoother tendons.

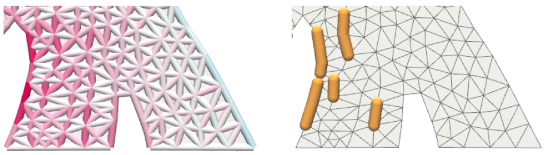


(a) 177 muscle-fibers (b) 414 muscle-fibers (c) 1055 muscle-fibers

Fig. 8. Comparison of winch-tendon varying fineness of triangulation for the "robot" example.

8.3 Advantages of networks

The use of winch-tendon networks (as opposed to winches driving just one tendon) enables the creation of complex motions with a low number of actuators. This is illustrated in Figure 7, where a single winch-tendon network successfully deforms each leg of our "E" example in a different way, as specified by the user. The top leg curves up, the bottom leg curves down, and the middle leg curves into an S-like shape.



(a) Activation graph. (b) Thresholded fibers.

Fig. 9. Detail of the legs of the finely-triangulated robot example. The overall motion of "bending to the left" is accomplished in parallel by multiple groups of active muscle-fibers. Note: SEED_NUM lowered for visualization.

9 LIMITATIONS AND FUTURE WORK

9.1 Network reusability

While complex networks can be very useful for creating single poses, in practice we find that the more complex a network is, the less likely it is be reused for other motions. Simpler winch-tendon networks tend to be more reusable in this sense, and so are excellent for creating animation sequences (which are just a series of poses). This is illustrated by Figure 4, in which a fairly complex animation sequence was created using a set of simple winch-tendon networks. An interesting avenue of future work would be to automatically decompose a complex network (created to match one particular pose) into simpler constituent networks, which could be more readily reused to create a richer space of poses.

9.2 Slack and exotic winches

In this work we assumed that there is no slack in any of the cables when the plushie is assembled. We also assumed a basic type of physical winch, one that has the same rate of contraction for all of its cables. These were convenient assumptions for fabrication and assembly, but removing them would increase design freedom.

If we had some cables slack at assembly, for example, we could control the relative timing of cable contractions. If we used a winch with multiple spools of different radii, we could control rates of cable contraction. Timing and rates of motion are important concepts in traditional animation, and their role in animating physical plushies would be interesting to explore further.

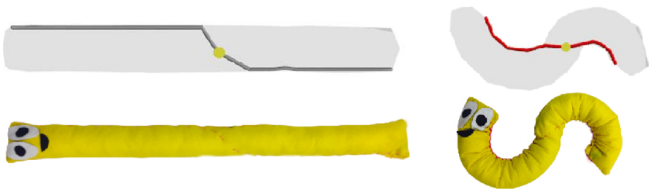


Fig. 10. The "S" example shown in simulation and fabricated.

9.3 Outlook

Although the design concepts we introduce are general, we have yet to apply our method to plushies that can create desired 3D motions in the presence of external forces due to gravity, user interactions, or manipulation of other objects. Furthermore, while this work focused on the design of animated plush toys, we believe it also represents a very promising step towards creating soft robots that are specifically designed to be safe, affordable, and personalized to specific needs and preferences.

ACKNOWLEDGEMENTS

We thank Timothy Mueller-Sim, Frances Tso, and Merritt Jenkins for helping out with physical prototypes, and we thank the anonymous reviewers for their insightful suggestions. This work was supported in part by NSF awards CHS-1566244 and NRI-1637853.

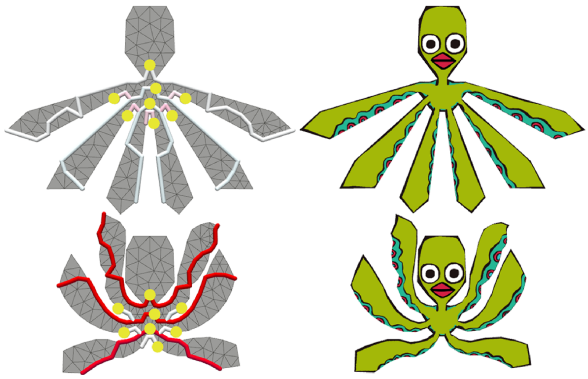


Fig. 11. The "squid" example in simulation, performing a swimming motion.

REFERENCES

Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating articulated characters from skinned meshes. *ACM Transactions on Graphics* 31, 4 (2012), 1–9. DOI: https://doi.org/10.1145/2185520.2335398

Bernd Bickel, Peter Kaufmann, Mélina Skouras, Bernhard Thomaszewski, Derek Bradley, Thabo Beeler, Phil Jackson, Steve Marschner, Wojciech Matusik, and Markus Gross. 2012. Physical Face Cloning. *ACM Trans. Graph.* 31, 4, Article 118 (July 2012), 10 pages. DOI: https://doi.org/10.1145/2185520.2185614

J Cali, D A Calian, C Amati, R Kleinberger, A Steed, J Kautz, and T Weyrich. 2012. 3D-Printing of Non-Assembly, Articulated Models. *Acm Transactions on Graphics* 31, 6 (2012). DOI: https://doi.org/Artn13010.1145/2366145.2366149

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and fabricating mechanical automata from mocap sequences. *ACM Transactions on Graphics* (2013). DOI: https://doi.org/10.1145/2508363.2508400

Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. 2016. Synthesis of Filigrees for Digital Fabrication. *ACM Trans. Graph.* 35, 4, Article 98 (July 2016), 13 pages. DOI: https://doi.org/10.1145/2897824.2925911

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

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics* 32, 4 (2013), 1. DOI: https://doi.org/10.1145/2461912.2461953

Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Transactions on Graphics* 35 (2016).

Yohsuke Furuta, Nobuyuki Umetani, Jun Mitani, Takeo Igarashi, and Yukio Fukui. 2010. A Film Balloon Design System Integrated with Shell Element Simulation.. In *Eurographics (Short Papers)*, Hendrik P. A. Lensch and Stefan Seipel (Eds.). Eurographics Association, 33–36. http://dblp.uni-trier.de/db/conf/eurographics/eg-short2010.html#FurutaUMIF10

Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire Mesh Design. *ACM Trans. Graph.* 33, 4, Article 66 (July 2014), 12 pages. DOI: https://doi.org/10.1145/2601097.2601106

Damien Gauge, Stelian Coros, Sandro Mani, and Bernhard Thomaszewski. 2014. Interactive Design of Modular Tensegrity Characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '14)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 131–138. http://dl.acm.org/citation.cfm?id=2849517.2849539

Yuki Igarashi and Takeo Igarashi. 2008. Pillow: Interactive Flattening of a 3D Model for Plush Toy Design. In *Proceedings of the 9th International Symposium on Smart Graphics (SG '08)*. Springer-Verlag, Berlin, Heidelberg, 1–7. DOI: https://doi.org/10.1007/978-3-540-85412-8_1

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved Folding. *ACM Trans. Graph.* 27, 3, Article 75 (Aug. 2008), 9 pages. DOI: https://doi.org/10.1145/1360612.1360674

Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. 2016. Beyond Developable: Computational Design and Fabrication with Auxetic Materials. *ACM Trans. Graph.* 35, 4, Article 89 (July 2016), 11 pages. DOI: https://doi.org/10.1145/2897824.2925944

Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. 2011. A Geometric Study of V-style Pop-ups: Theories and Algorithms. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 98, 10 pages. DOI: https://doi.org/10.1145/1964921.1964993

Xian-Ying Li, Chao-Hui Shen, Shi-Sheng Huang, Tao Ju, and Shi-Min Hu. 2010. Popup: Automatic Paper Architectures from 3D Models. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 111, 9 pages. DOI: https://doi.org/10.1145/1833349.1778848

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-Printable Robotic Creatures. *ACM Transactions on Graphics* (2015).

Jun Mitani and Hiromasa Suzuki. 2004. Making Papercraft Toys from Meshes Using Strip-based Approximate Unfolding. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. ACM, New York, NY, USA, 259–263. DOI: https://doi.org/10.1145/1186562.1015711

Yuki Mori and Takeo Igarashi. 2007. Plushie: An Interactive Design System for Plush Toys. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH '07)*. ACM, New York, NY, USA, Article 45. DOI: https://doi.org/10.1145/1275808.1276433

Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. Graph.* 34, 4, Article 138 (July 2015), 12 pages. DOI: https://doi.org/10.1145/2766998

Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. 2015. Interactive Surface Design with Interlocking Elements. *ACM Trans. Graph.* 34, 6, Article 224 (Oct. 2015), 7 pages. DOI: https://doi.org/10.1145/2816795.2818128

Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. Computational Design of Rubber Balloons. *Comput. Graphics Forum (Proc. Eurographics)* (2012).

Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational Design of Actuated Deformable Characters. *ACM Trans. Graph.* 32, 4, Article 82 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461979

Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. 2014. Designing Inflatable Structures. *ACM Trans. Graph.* 33, 4, Article 63 (July 2014), 10 pages. DOI: https://doi.org/10.1145/2601097.2601166

Peng Song, Chi-Wing Fu, Prashant Goswami, Jianmin Zheng, Niloy J. Mitra, and Daniel Cohen-Or. 2013. Reciprocal Frame Structures Made Easy. *ACM Trans. Graph.* 32, 4, Article 94 (July 2013), 13 pages. DOI: https://doi.org/10.1145/2461912.2461915

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

Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-Based Characters. *ACM Transactions on Graphics* 33 (2014).

Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of Self-supporting Surfaces. *ACM Trans. Graph.* 31, 4, Article 87 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185583

Yohei Yamashita, Tatsuya Ishikawa, Hironori Mitake, Yutaka Takase, Fumihiro Kato, Ikumi Susa, Shoichi Hasegawa, and Makoto Sato. 2012. Stuffed toys alive!: cuddly robots from fantasy world. In *ACM SIGGRAPH 2012 Emerging Technologies*. ACM, 20.

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing Structurally-sound Ornamental Curve Networks. *ACM Trans. Graph.* 35, 4, Article 99 (July 2016), 10 pages. DOI: https://doi.org/10.1145/2897824.2925888

Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided Mechanical Toy Modeling. *ACM Transactions on Graphics* 31, 6 (2012), 127:1–127:10. DOI: https://doi.org/10.1145/2366145.2366146