



Neural reservoir control of a bio-hybrid soft arm

Noel Naughton^{a,1} , Arman Tekinalp^{b,c,2} , Keshav Shivam^{c,d,3}, Seung Hyun Kim^{b,c} , Apoorva Khairnar^a , Volodymyr Kindratenko^{c,d} , and Mattia Gazzola^{b,c,d,e,1}

Affiliations are included on p. 11.

Edited by K. W. Wang, University of Michigan, Ann Arbor, MI; received September 15, 2025; accepted March 25, 2026, by Editorial Board Member John A. Rogers

A long-standing engineering problem, the control of soft robots is difficult because of their highly nonlinear, heterogeneous, anisotropic, and distributed nature. Here, bridging engineering and biology, neural reservoirs are employed for the dynamic control of a bio-hybrid model arm made of multiple muscle-tendon groups enveloping an elastic spine. We show how the use of reservoirs facilitates simultaneous control and self-modeling across challenging tasks, outperforming classic neural network approaches. Further, through the use of spiking reservoirs on neuromorphic hardware, energy efficiency gains of up to 75 and 45 times are obtained relative to standard and high-efficiency CPUs, with implications for the on-board control of untethered, small-scale systems.

soft robotics | bio-inspiration | neuromorphic computing | reservoir computing

Hyperredundancy, underactuation, distributedness, and continuum mechanics are defining features of soft robots [artificial or biological (1–8)], intrinsic to their compliant, elastic constitutive materials. These traits are attractive in the pursuit of extreme reconfigurability, morphological adaptivity, delicacy, and dexterity, for applications in medicine, defense, or agriculture (9–12). However, the same advantageous traits pose fundamental challenges in control, due to the associated vast space of degrees of freedom and the highly nonlinear dynamics involved.

Recognizing the importance of the problem, significant effort has been devoted to the development of control approaches suitable for continuum and elastic structures (13–15). Model-based controllers have proven effective in quasi-static settings, but lack accuracy when inertial effects become significant and typically rely on simplifying assumptions that may overlook environmental interactions, anisotropy, or material nonlinearities (14). Data-driven approaches bypass such modeling difficulties by directly learning associated dynamics via artificial neural networks (16, 17). These techniques are generally resource-intensive and may employ neural architectures not necessarily well-suited to capture the long-range spatial and temporal dependencies that define soft systems (9, 18), hindering robustness.

An alternative, bio-inspired route to soft robotic control hinges on the concept of “mechanical intelligence” (19–22), whereby nonlinear structural and environmental effects are purposefully exploited to passively achieve adaptivity, thus simplifying and robustifying control. Complementarily, rhythm-generation techniques inspired by the nervous system (e.g., central pattern generators) have also been shown effective, particularly for locomotion (23, 24), spurring bio-hybrid attempts where neural-tissue is directly integrated with soft scaffolds, muscles, and electronics to drive simple actuation dynamics (7, 8, 25–27).

Here, with the goal of intimately meshing compliant mechanics and neural dynamics for enhanced adaptivity and control, we numerically explore the integration of a soft bio-hybrid arm with reservoir computing (RC). Partially inspired by the mammalian neocortex (28), RC was developed (29) for processing real-time, analog, spatiotemporally correlated inputs, by means of a dynamically rich substrate (30–32), the reservoir. The reservoir, which in principle can be any dynamical system (33), integrates and projects input data streams into a separable, high-dimensional latent space that decomposes nonlinear correlations. Reservoir dynamics are then sampled and recombined via linear maps into desired computations. While a variety of algorithmic and physical implementations have been proposed (31, 33), in general the RC paradigm offers several appealing features (29, 34): it does not require expensive backpropagation (only output linear maps are learned), it intrinsically accommodates nonlinear, spatiotemporally correlated dynamics, it is natively parallel as multiple maps can be learned separately

Significance

Controlling soft robots is challenging due to their complex, nonlinear dynamics. We show that a bio-inspired reservoir computing approach enables the control of a simulated musculoskeletal bio-hybrid arm beyond traditional neural network performance. The use of neural reservoirs naturally accommodates the arm's elastic dynamics and allows simultaneous muscle coordination and self-modeling. When implemented on neuromorphic hardware, this approach delivers significant computational energy savings, a key requirement for untethered small-scale robots.

Author contributions: N.N., A.T., K.S., S.H.K., V.K., and M.G. designed research; N.N., A.T., K.S., S.H.K., A.K., and M.G. performed research; V.K. contributed new reagents/analytic tools; N.N., A.T., K.S., S.H.K., A.K., and M.G. analyzed data; and N.N., A.T., K.S., S.H.K., and M.G. wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission. K.W.W. is a guest editor invited by the Editorial Board.

Copyright © 2026 the Author(s). Published by PNAS. This article is distributed under [Creative Commons Attribution-NonCommercial-NoDerivatives License 4.0 \(CC BY-NC-ND\)](#).

¹To whom correspondence may be addressed. Email: nnaughton@vt.edu or mgazzola@illinois.edu.

²Present address: Department of Mechanical Engineering, University of Maryland, College Park, MD 20742

³Present address: Google, Mountain View, CA 94043.

This article contains supporting information online at <https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2522094123/-/DCSupplemental>.

Published April 24, 2026.

to achieve different tasks while running on the same reservoir, and can be matched with specialized hardware [for example energy efficient neuromorphic systems (35, 36)] or “wetware” [neural tissue used as living reservoir (37–40)].

In this work, we consider a neural reservoir (modeled as a recurrently connected or spiking network) able to sense the shape (proprioception) of a simulated bio-inspired arm made of muscles and tendons enveloping an elastic spine. Neural dynamics are coupled to muscle actuations through reinforcement learning (RL), to learn to control the system’s musculature in an unsupervised manner. We demonstrate that our approach produces control policies that outperform traditional deep-learning methodologies, with advantages seen to widen as the arm’s compliance increases. RC’s high-dimensional latent space is also leveraged, through parallel output maps, for concurrent self-modeling to improve control robustness during conditions of disturbance and sensing failure. Further, motivated by their potential relevance in bio-hybrid contexts (7, 8, 37, 41, 42), spiking neural reservoirs are considered, and subsequently mapped onto neuromorphic hardware, attaining up to ~75- and ~45-fold reduction in energy usage, depending on the reference CPU model. Neuromorphic RC is finally employed to drive the arm through a set of unstructured obstacles, learning to exploit solid objects to reshape and facilitate the reaching of a target—a hallmark of mechanical intelligence that is recovered here.

Overall, this work not only advances soft robotic control, but also furthers neuro-mechano integration, a characteristic trait of organic systems. This, in turn, may lead to designs that incorporate artificial and living elements (7, 8, 26, 37, 40, 43), while providing insights into biological processing.

Modeling of a Musculoskeletal Bio-Hybrid Soft Arm. While no soft, synthetic actuator has yet reached biologically comparable levels of performance, improved artificial (44, 45) or tissue-engineered (8, 46) prototypes are continually being developed. Cognizant of this rapid development, we consider the automated, dynamic control of a computational, bio-hybrid neuromuscular arm, characterized by multiple, antagonistic and overlapping muscle-tendon pairs arranged around and along an elastic supporting spine (Fig. 1A).

Such a design captures key organizational motifs commonly encountered in biology and, increasingly, in robotics (21, 24). Antagonistic muscles are indeed ubiquitous in both limbed (47, 48) and limbless animals (49, 50), and are gradually being integrated in robotic systems (2, 4, 8, 51). Their asymmetric contractions enable rotations [i.e., joints (47)] or continuum bending and twisting [e.g., octopus arms (49, 52)], while graded cocontractions can improve structural stability (53, 54) by actively increasing body stiffness. Similarly, overlapping muscle groups [of which the many staggered spinal muscles of snakes provide an extreme example (50)] enable precise, distributed, and efficient actuation (55). Our model arm then allows us to explore compliant actuation and control in a biologically and engineering relevant setting, while offering a generalizable blueprint for future experimental implementations. For example, the elastic spine may be realized at various scales via molding or 3D printing, and decorated with arrays of compliant actuators, from advanced supercoiled artificial muscles (56) to cultured skeletal muscles (8, 42).

To computationally instantiate our arm, we employ a modeling approach based on assemblies of Cosserat rods [Materials and Methods, (57, 58)]. These are slender, one-dimensional elements able to undergo all modes of deformation—bend, stretch, twist, and shear—to dynamically reconfigure in three-dimensional

space. Cosserat rods provide a convenient modeling representation for several reasons. They naturally map to elastic beams (spine), tendons, and muscles, they can actively contract and stretch along their length according to prescribed force–length relations, and they can be connected together (via appropriate boundary conditions) into arbitrary architectures. They are thus well-suited to capture the heterogeneous, anisotropic, and distributed nature of our bio-inspired arm. Assemblies of rods and their governing equations are numerically discretized and solved via the open-source software *Elastica* (59), whose quantitative utility has been demonstrated in a range of biophysical settings, including animal locomotion (55, 60, 61), invertebrate manipulation (52), plant dynamics (62), fibrous metamaterials (63–65), and soft robotic design and control (8, 42, 55, 66–71).

We begin with the soft elastic spine/backbone, represented as a single passive rod, onto which 16 muscle-tendon units are patterned (Fig. 1A and SI Appendix, Fig. S1). Three-dimensional arm deformations are achieved by organizing muscle-tendon units into four layers of orthogonal agonist-antagonistic pairs, with adjoining layers overlapping by 50% of their length and offset 45° to avoid intersection. This architecture enables bending in all directions, via the coordinated activation of orthogonal muscles combined with the omnidirectional bending/twisting capability of the elastic backbone. Further, muscle overlap allows for continuously graded contractions along the arm despite the limited number of individual muscles (55). Each muscle-tendon unit is modeled as an individual rod with an actively contractible muscle belly and tapered tendinous ends that insert into the backbone. The entire unit is glued lengthwise to the spine, to conform to it while bending, mimicking the presence of surrounding fascia tissue or encapsulating materials.

The active force–length and force–velocity relationships of the muscle belly and its passive hyperelastic material properties as well as the tendon’s material properties are based on reported biomechanical data (Materials and Methods). Noteworthy is the nonlinear passive stress–strain behavior of muscle and tendon tissues (72–74), which are characterized by a “J-curve” response that allows for compliant deformations at small strains, before stiffening at larger strains for structural stability (75). This is a key mechanical feature of biological musculoskeletal tissues, and significant engineering effort has been recently devoted to recapitulate similar properties (76–79). Finally, each muscle is independently controllable via a continuous tetanic activation ($a_i \in [0, 1]$, $i \in \{1, \dots, 16\}$). Full details are provided in Materials and Methods. We readily acknowledge the technological challenges associated with experimentally realizing such an arm, but emphasize the opportunity to computationally explore novel control approaches for next-generation implementations.

We challenge the arm to learn to coordinate its muscle activations (action space $\mathbf{a}_t(t)$) and dynamically track a target moving along a smoothly varying, random 3D trajectory (Fig. 1A). First, we outfit the arm with a compact set of sensory capabilities. Analogous to strain-sensing muscle spindles in vertebrates (80), proprioception is provided here through the arm’s curvature $\bar{\kappa}_i(t)$, estimated at four equally spaced locations (Materials and Methods). Environmental information is limited to the distance vector $\mathbf{x}_t(t)$ between the (fixed) base of the arm and the moving target, representing visual or acoustic tracking. Notably, this sensory arrangement is laboratory-frame invariant. Indeed, local curvature is sufficient to recover the arm shape, while the target is sensed relative to the arm base. Thus, overall, the arm state is constituted by $\bar{\kappa}_i(t)$ and $\mathbf{x}_t(t)$, with all higher-order information (velocities or accelerations) implicitly captured through the memory of the reservoir (31), described next.

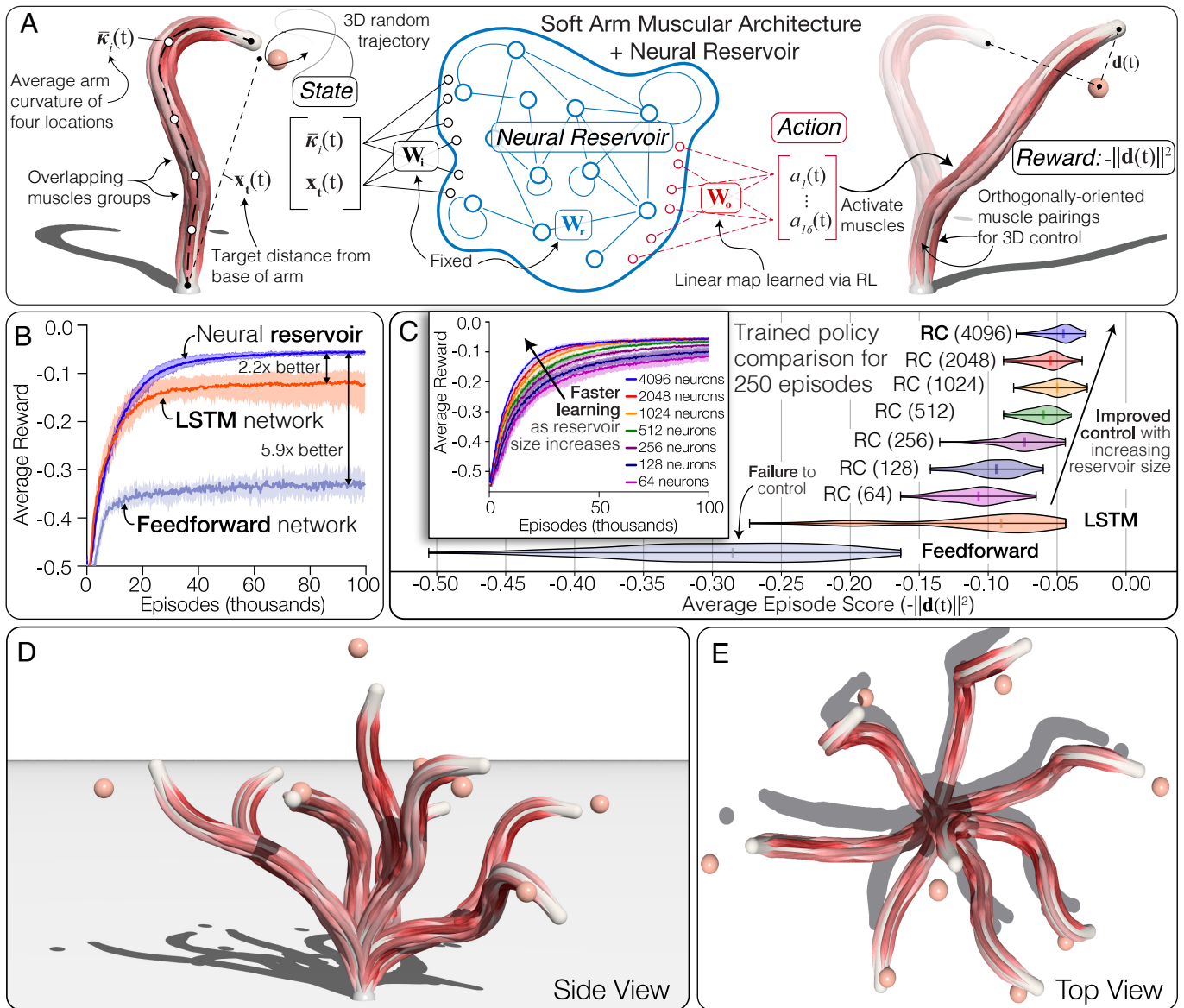


Fig. 1. Neural reservoir control of a musculoskeletal soft arm. (A) Schematic depicting the coupling of a neural reservoir with a soft bio-inspired arm made of sixteen muscle-tendon units enveloping an elastic spine to achieve autonomous control. (B) Learning performance for the control of a soft bio-inspired arm (backbone stiffness: 125 kPa) tracking a 3D moving target: neural reservoir and traditional feedforward and LSTM network architectures. The solid lines depict the average learning performance (average episode return), each obtained by considering five neural architectures instances initialized using different random number generator seeds. The shaded regions depict the spread in learning performance of the five architectures with different initial seeds. (C) Violin plots of average performance of trained policies evaluated over 250 episodes, for different reservoir sizes and reference architectures (FF and LSTM). *Inset* shows learning performance as reservoir size increases, illustrating how both more rapid and better overall learning performance is achieved as the reservoir size increases. For FF and LSTM, different numbers of neurons and layers/stacks were considered (FF: [64×64], [128×128], [64×64×64], [64×64×64×64]; LSTM: [64×1 stack], [128×1 stack], [256×1 stack], [64×2 stacks], [128×2 stacks], [256×2 stacks]), with no significant differences observed. Here we report data for the best performing networks. (SI Appendix for full details. A comparison with PID control is also reported for completeness, finding it inadequate for controlling dynamic soft arms.) Overlaid snapshots of (D) side view and (E) top view of the bio-inspired arm successfully tracking a 3D target (see also Movie S1) using a trained neural reservoir control policy. Intensity of muscle color (pale to dark red) denotes a muscle's activation level.

Neural Reservoir Control. Controlling this structure is challenging because of the arm's continuum elastic nature, characterized by highly nonlinear and long-range stress propagation effects. Indeed, distributed and localized loads (musculature/environment) are communicated throughout the entire system, potentially leading to mechanical instabilities and global morphological reconfigurations. Compounding this complexity are the nonlinear contractile properties of the muscle-tendon units, the anisotropic and heterogeneous quality of the arm, as well as its inertial dynamics.

To establish control (Fig. 1A), we seek to learn, in an unsupervised, model-free fashion, muscle activations (action:

$a = \{a_j(t)\}$, $j \in \{1, \dots, 16\}$) that minimize at all times the distance $\mathbf{d}(t)$ between the arm tip and the target (reward: $r = -\|\mathbf{d}(t)\|^2$), based on available sensory information (state: $s = \{\mathbf{x}_i(t), \bar{\kappa}_i(t)\}$, $i \in \{1, \dots, 4\}$). The mapping (control policy) between state and action space may be represented by any neural network topology (81, 82), and it is here that we insert our neural reservoir (Fig. 1A). This is initially realized as a recurrent artificial neural network (Materials and Methods) paired with a single, external linear recombination layer (map) whose function is to transform the reservoir's internal dynamics (sampled at each neuron) into muscle contractions $a = \{a_j(t)\}$. To learn this linear map, we employ the reinforcement learning (on-policy)

proximal policy optimization (PPO) strategy (83). We emphasize here that since RC requires learning the output map weights only, all connections from the input state to the reservoir and within the reservoir itself remain fixed, removing the need for backpropagation.

To contextualize the reservoir performance, we additionally consider feedforward (FF) and long short-term memory (LSTM) networks, reference architectures commonly employed in learning-based control (84, 85). Training is again performed using PPO, where this time the network weights are all learned (as opposed to RC) through backpropagation. To facilitate a meaningful comparison, all network topologies are evaluated against the same control problem, and set up/trained according to the same protocol (*Materials and Methods*).

As illustrated in Fig. 1 B–E and Movie S1, upon training, the neural reservoir successfully learns to track the target, substantially outperforming the FF and LSTM networks. Indeed, the FF network is found to quickly reach a (poor) performance limit beyond which it is unable to improve (Fig. 1B). The LSTM network converges to a better performance, although the ceiling remains significantly below the reservoir's (which is $2.2\times$ better). The neural reservoir's performance is found to correlate with reservoir size, whereby learning speed and tracking accuracy increase as the number of neurons grows (Fig. 1C). This is in keeping with the known ability of larger reservoirs to better capture complex dynamics (31, 86) and supports the hypothesis that the control of elastic systems may benefit from a specific focus on highly nonlinear, spatiotemporally correlated dynamics. The neural reservoir's performance is also found to be robust to perturbations in target trajectory as well as arm's material and actuation properties, as extensively explored in SI Appendix and further dissected in the following sections.

To further test these aspects, we investigate the effect of the arm's elastic spine stiffness by progressively decreasing its elastic modulus (while holding mass and geometry constant). This softening enables larger modes of deformations, further exacerbating the system's nonlinear response, rendering control harder and allowing us to tease out differences between RC, FF, and LSTM approaches. We begin with a relatively stiff backbone (1 MPa, equivalent to rubber (87) and $\sim 10\times$ stiffer than Fig. 1B) for which all three network topologies successfully learn to track the moving target (Fig. 2A and Movie S2). However, as soon as the spine softens, differences start to manifest. The learned FF controllers are indeed observed to rapidly degrade, practically failing to coordinate the arm already at 500 kPa (Movie S2). The LSTM network instead initially maintains control, although below 250 kPa its performance too rapidly degrades (Movie S2). In contrast, the neural reservoir exhibits a compact performance envelope and maintains sustained control down to a 62.5 kPa stiffness (comparable to mammalian (88) or octopus (89) muscle tissue (88), for reference).

An energetic analysis reveals that the neural reservoir minimizes (relative to FF and LSTM, Fig. 2 B and C) both the kinetic and bending energy of the spine. For each network architecture, kinetic energy (Fig. 2B) is found to remain approximately consistent across Young's moduli, albeit settling at different levels, with LSTM and FF entailing $\sim 3\times$ and $\sim 10\times$ larger energies than RC, respectively. Phenomenologically, higher kinetic energies manifest as faster swings or pronounced structural vibrations (Movie S2), implying that the RC-based controller (which produces reduced kinetic energies) exhibits more precise and smoother actuation, without extensively relying on corrective muscle contractions. The neural reservoir also demonstrates clear superiority in minimizing the spine's bending energy (Fig. 2C).

It is worth noting that bending stiffness scales linearly with the elastic modulus. We would thus expect that a controller able to effectively coordinate the arm's musculature would produce bending deformations (and therefore energies) congruent with this linear scaling as the spine softens. This trend is indeed met by RC, while LSTM and FF networks are found to exhibit a sublinear decrease. This indicates that LSTM and FF networks resort to deformations that are excessive in relation to the spine's stiffness, eventually leading to failure (Movie S2).

Recorded muscle-activation patterns indicate that RC's superior control arises from its extensive use of antagonistic cocontractions across all four muscle layers of the arm. These co-contractions locally stiffen and stabilize the arm as it tracks the targets. As detailed in SI Appendix, Figs. S6 and S7, the neural reservoir discovers a strategy in which muscles at the base of the arm employ moderate, asymmetric cocontractions that both stiffen the base and allow the arm to coarsely orient itself toward the target. The intermediate muscle layers instead rely on strong, symmetric cocontractions to maximize stability in the arm's midsection. Finally, the upper muscle layers use strongly asymmetric contractions to finely orient the arm's tip toward the target. In contrast, the LSTM and FF controllers fail to discover this multi-layered, coactivation strategy. Indeed, they largely avoid co-contractions, relying instead on unilateral muscle activations that lead to poor control and unstable arm dynamics.

Therefore, RC-based controllers are shown to achieve higher tracking accuracies and, concurrently, avoid unnecessary bending, twisting, or corrective accelerations, reflecting their ability to capture (and predict) the arm's dynamics, as quantified next.

Self-Modeling for Robust Control. Reservoir dynamics can also be used for self-modeling (90, 91), that is, for the explicit estimation or prediction of the system's configuration, which in turn can be used to robustify the controller. Self-modeling is achieved here through three dedicated maps (Fig. 3A), to extract the arm's full pose, future target location, and future tip location. Maps are obtained in a supervised fashion via Ridge regression based on data from the 250 kPa case of Fig. 2A (*Materials and Methods*). The maps are then directly appended to the reservoir, exploiting RC's intrinsic parallelism whereby several computations can be carried out concurrently using the same underlying dynamics.

We show how the reservoir, using target location and curvature proprioception history, accurately synthesizes future target locations (Fig. 3B), current arm pose (Fig. 3D and Movie S3), and future tip positions (by anticipating forthcoming muscle activations selected by the control policy based on the predicted target trajectory, Fig. 3C). We note that in all cases target trajectories are randomly generated so that the reservoir must constantly produce new predictions. These capabilities are then directly used to improve control in a scenario where the arm intermittently loses sight of the target, mimicking a faulty sensor or environmental disturbances. For the arm to keep "seeing," the target's next location is predicted and fed back into the reservoir in lieu of the missing data. This in turn enables the forecasting of muscle actuations and future tip positions, allowing the blinded arm to continue tracking (Fig. 3 E and F). Using this strategy, inference periods of less than 1 s (interval during which the arm is "blind," approximately equivalent to the time need by the target to travel a distance of 50% the arm's length) result in no performance degradation relative to perfect knowledge of the target location. If the disturbance lasts beyond 1 s, then

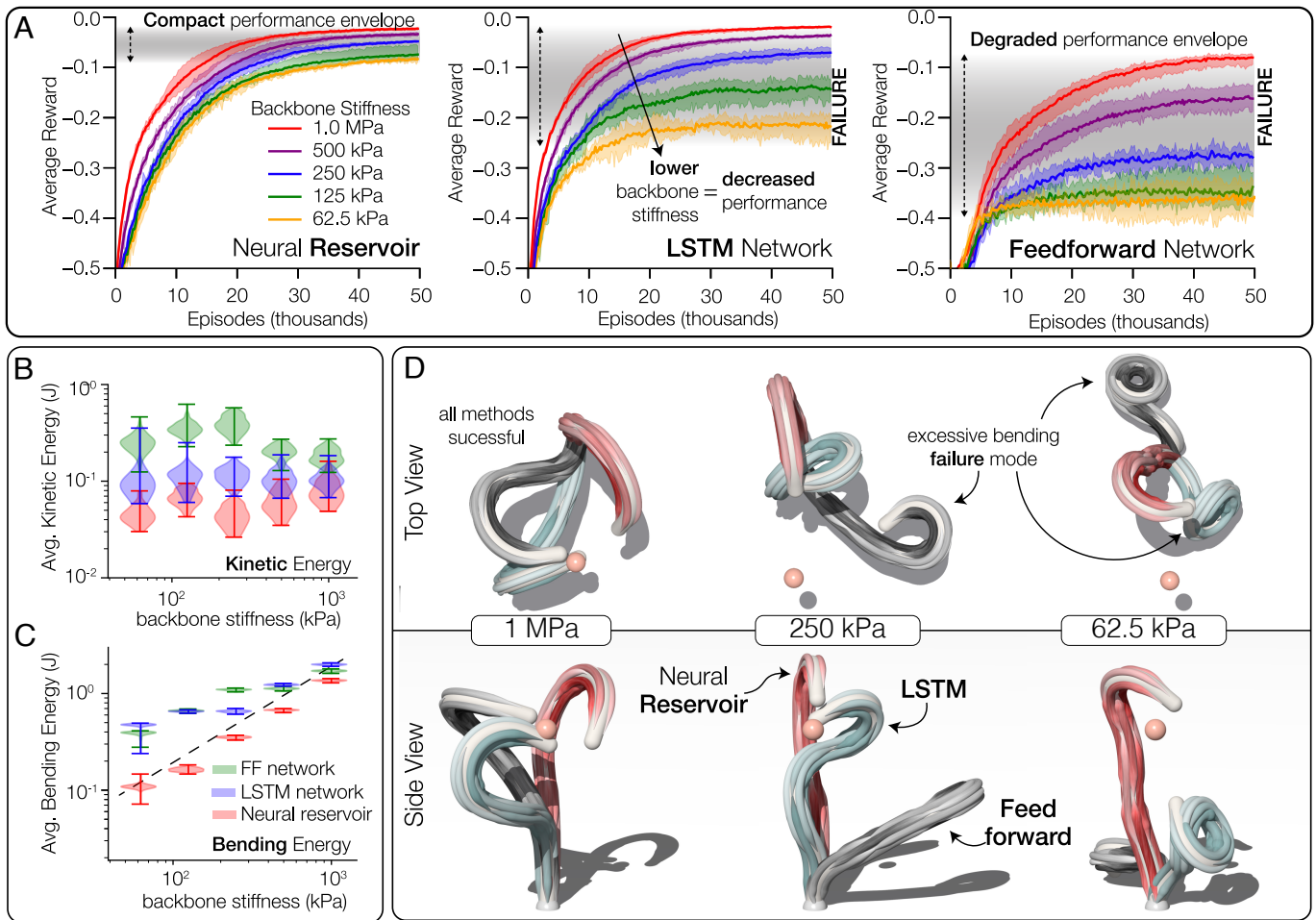


Fig. 2. Control of increasingly compliant arms. (A) Learning performance of a neural reservoir (4,096 neurons) compared to LSTM and FF networks for a soft bio-inspired arm with decreasing backbone elastic moduli. The neural reservoir exhibits a compact performance envelope while the LSTM and FF networks exhibit much wider performance envelopes with control performance drastically decreasing as the backbone softens. Solid and shaded regions denote the performance average and spread relative to five randomly initialized network instances (as in Fig. 1B). Violin plots of the average (B) kinetic and (C) bending energy of the arm's spine, when RC, LSTM, and FF network architectures are used for controlling systems of decreasing backbone stiffness. Values reported are total kinetic/bending energy integrated over the episode and normalized by episode length. The RC network exhibits lower energies in all cases. Bending energy is observed to decrease linearly in the case of RC, in keeping with the linear softening of the spine and in contrast with FF and LSTM, which exhibits sublinear scalings. (D) Snapshots of trained policy performance for different backbone stiffness levels illustrating control failure modes. While all three network architectures successfully control a soft arm with a backbone stiffness of 1 MPa, for a stiffness of 250 kPa the feedforward network fails to coordinate muscle contractions, demonstrating an excessive bending failure mode (with an associated increased bending energy compared to RC, see panel B). While the LSTM network exhibits better performance, it also produces excessive bending and fails as the backbone stiffness continues to decrease (62.5 kPa). Video comparison of tracking performances for all backbone stiffness cases is available in Movie S2. As in Fig. 1, for FF and LSTM, different numbers of neurons and layers/stacks were considered, with no significant differences observed across arm's stiffnesses. Here we report data for the best performing networks (SI Appendix for full details).

tracking quality begins to decrease, although modestly. Indeed, average performance remain comparable to an LSTM network with complete knowledge of the target location (SI Appendix). However, the spread of the arm performance (violin plot bars in Fig. 3E) increases for large disturbance periods, implying that the arm does occasionally lose track of the target while blinded.

Thus, a mixed strategy where unsupervised learning is used for control and supervised learning is employed for self-modeling, with the second supporting the first, is found to be effective in robustifying performance (Movie S4). It is worth noting that the inclusion of self-modeling entails minimal computational overhead since all linear maps use the same reservoir.

Spiking Reservoirs on Neuromorphic Hardware. Here, we illustrate how our approach is also well-positioned to enhance computational energy efficiency, a concern in many robotic applications (92, 93). To this end, we consider a spiking

neural reservoir on neuromorphic hardware, demonstrating up to $\sim 75\times$ and $\sim 45\times$ improvements relative to standard and high-efficiency CPUs, while retaining control performance.

The use of spiking networks, inspired by cortical dynamics (31, 94), well-aligns with both RC (29) and specialized neuromorphic chips (35, 36) that physically implement energy efficient artificial spiking neurons (95, 96). It is thus only natural to explore the use of spiking networks within our framework, also envisioning potential applications where spiking neural tissue may replace synthetic reservoirs (37, 41).

To align algorithmic and hardware infrastructures, we implement a reservoir of leaky integrate-and-fire (LIF) spiking neurons on Intel's neuromorphic Loihi chip (35) (Fig. 4A, Materials and Methods). We evaluate the target-tracking performance achieved by spiking reservoirs and compare with the non-spiking controllers presented in Fig. 1. For completeness, we also consider spiking counterparts of the LSTM and FF architectures using

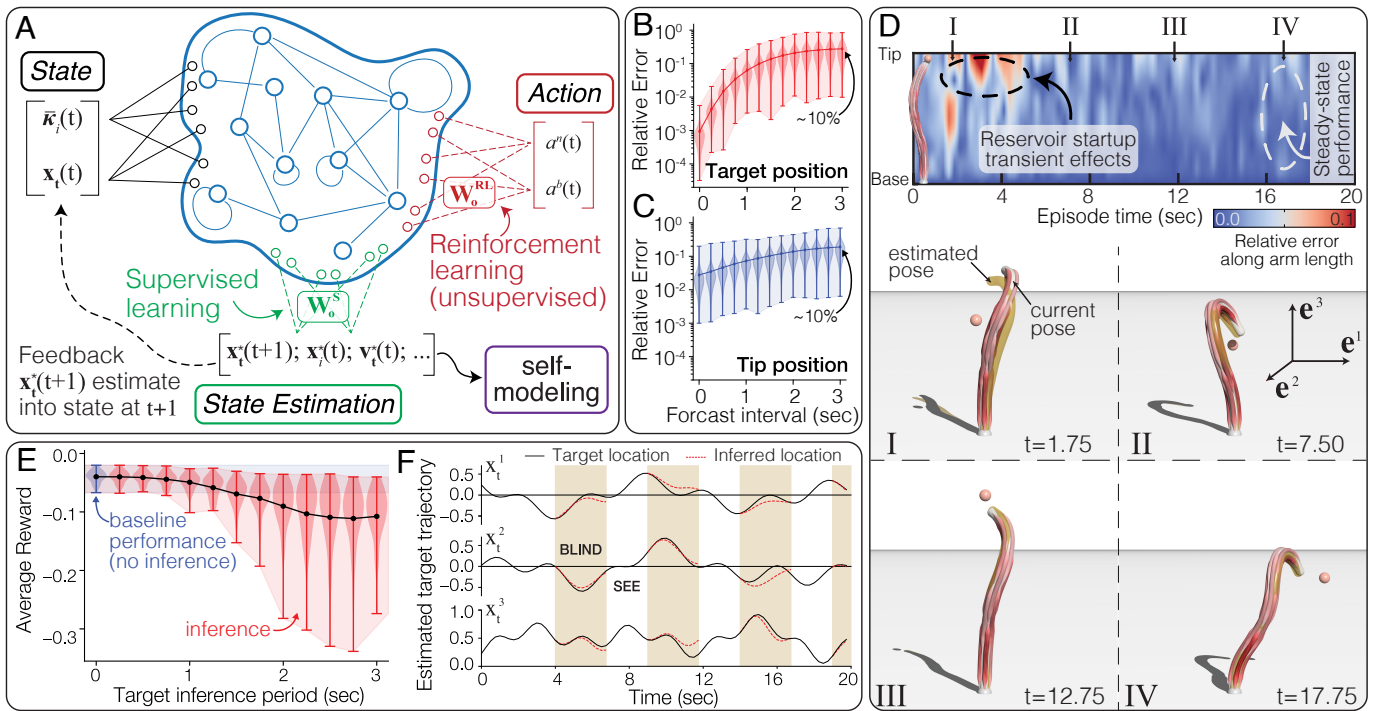


Fig. 3. Parallel maps for self-modeling and robust control. (A) Schematic of neural reservoir control equipped with additional, parallel maps to infer/predict state information. All reported results are for an arm with a backbone stiffness of 250 kPa controlled by an already trained (via RL) neural reservoir with 4,096 neurons, as described in Fig. 1. (B) Accuracy of parallel map estimates of future target positions, for increasing time-windows into the future. (C) Accuracy of parallel map estimates of future arm's tip position for increasing time-windows into the future. For both (B) and (C), the relative error is defined as $\|\hat{\mathbf{y}} - \mathbf{y}\|/\ell$ where $\|\cdot\|$ is the L2-vector norm, $\hat{\mathbf{y}}$ is the predicted position, $\mathbf{y}$ is the true position, and ℓ is the length of the soft arm. (D) Performance of reservoir self-modeling for estimation of current arm pose (top row). Heat map of accuracy of pose estimation along the length of the arm. Color denotes relative error of estimation of the position of a point $s \in [0, \ell]$ along the soft arm of length ℓ . Relative error is defined as $\|\hat{\mathbf{y}}(s) - \mathbf{y}(s)\|/s$ where $\hat{\mathbf{y}}(s)$ is the predicted location of point s and $\mathbf{y}(s)$ is the true location of that point. Accuracy is initially lower due to transient startup effects in the reservoir before reaching a consistent level of high accuracy as confirmed (mid/bottom row) by visualization of the estimated (gold) and true (full color) arm pose at selected time instances. (E) Tracking performance of the neural reservoir when the target position becomes unavailable for increasing lengths of time. The target oscillates between being measured ("seen") and inferred ("blind") for equal time intervals that range in length from 0.25 to 3 s. Violin plots show performance over 50 trials for increasing periods of time during which the arm is blinded. The blue shaded region shows the baseline performance of the reservoir when the target location is always known. (F) Comparison of inferred target's 3D position for a 3 s inference period. After approximately 1.5 s, the estimate of the target trajectory sometimes drifts from the true trajectory, leading to the drop in tracking performance seen in panel E.

surrogate-gradient training methods (*Materials and Methods*). After establishing the control capabilities of the spiking-reservoir approach, we quantify the associated energy savings when deployed on neuromorphic hardware, benchmarking it against non-spiking reservoirs executed on conventional and high-efficiency CPUs.

Consistent with the nonspiking reservoir results of Figs. 1C and 2A, the spiking reservoir exhibits a compact performance envelope as the arm softens (Fig. 4B), and larger reservoir sizes improve performance (Fig. 4C). Further, the spiking reservoir exhibits similar self-modeling capabilities as the nonspiking reservoir (SI Appendix, Fig. S9), and recovers (SI Appendix, Fig. S10) the linear scaling of Fig. 2C. When compared with the LSTM and FF architectures, both the spiking and non-spiking reservoirs achieve better control in the target-tracking task (Fig. 4D). The non-spiking reservoir attains the highest overall performance, with the spiking reservoir following closely. Examination of the trained policies (Movies S1 and S5) reveals that both reservoir-based approaches successfully control the soft arm, whereas all LSTM and FF implementations frequently lead to arm collapse. Overall, the spiking models perform similarly to their non-spiking counterparts (Fig. 4E), albeit with slightly lower aggregate scores—a trend that persists across backbone stiffness (SI Appendix, Fig. S11).

However, spiking reservoirs deployed on the neuromorphic Loihi chip provide a clear advantage when energy consumption

is considered (Fig. 4F). Indeed, power consumption is reduced by up to $\sim 75\times$ relative to non-spiking reservoirs on standard x86-64 CPUs (Intel Xeon W-2665), and by up to $\sim 45\times$ compared with low-power, ARM-based CPUs (ARM Cortex-A53 and ARM Cortex-A76). Critically, energy consumption of spiking reservoirs on Loihi scales linearly with reservoir size, in contrast to the quadratic scaling observed for nonspiking reservoirs executed on conventional silicon hardware. This favorable scaling implies that significantly larger—and therefore more capable—spiking reservoirs can be employed in energy-constrained settings.

Environmental Interactions. A key advantage of soft robots is their ability to comply with the environment without damaging themselves or their surroundings. We explore these boundary interactions, and the possibility of taking advantage of them, by considering the reaching of a target through a set of unstructured obstacles (Fig. 4G–I). We underscore the steep control challenge associated with this test, exacerbated by a complex environment, geometric constraints, and contact dynamics.

In Fig. 4G and H, we show results obtained using a spiking neural reservoir for arms characterized by spines of decreasing stiffness. As can be seen, all arms successfully learn to reach through the obstacles with softer ones exhibiting slightly faster initial learning, ostensibly due to their ability to better conform to obstacles and push through gaps without precise actuation. However, the more precise control associated with stiffer arms

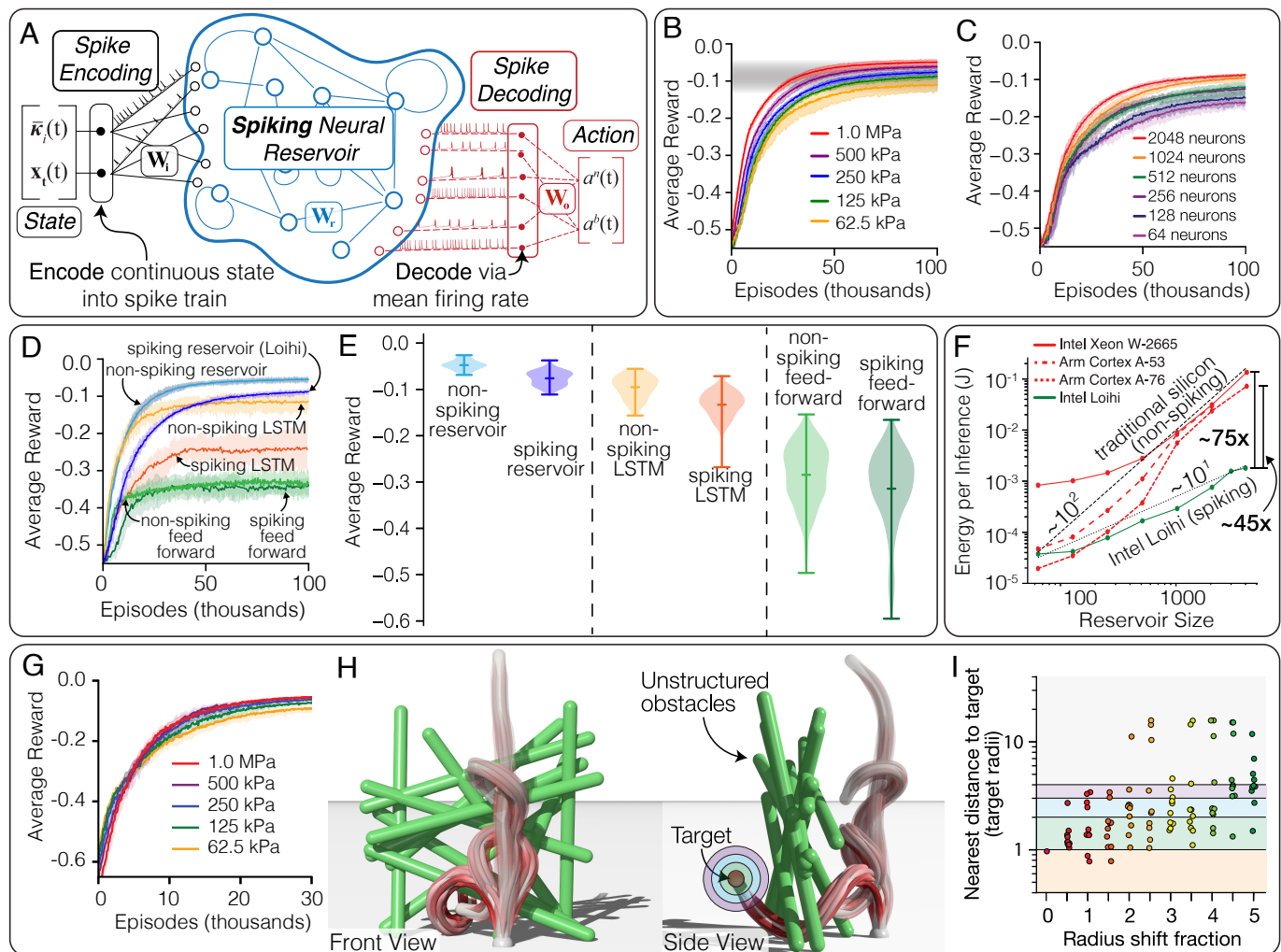


Fig. 4. Spiking neural reservoir control on neuromorphic hardware. (A) Schematic showing how the state is encoded into a spike train (SI Appendix) that is sent to a spiking neural reservoir running on an Intel Loihi chip. Spike train outputs are then decoded (SI Appendix) into continuous actions (contractions of the arm muscles). (B) The spiking reservoir exhibits a compact performance envelope as the backbone stiffness decreases. All cases are trained using five random initialization seeds as in Fig. 1B. (C) Learning performance for the 3D tracking problem (backbone stiffness of 125 kPa) of a spiking neural reservoir improves as the reservoir size increases up to 2,048 neurons (Nengo Loihi limitations precluded running 4,096 neurons). Performance trend is similar to increasing reservoir size for a reservoir made of artificial neurons (Fig. 2A). (D) Learning performance of a 2,048 neuron spiking reservoir vs. alternative network architectures (spiking and nonspiking) for 3D tracking and a backbone stiffness of 125 kPa. (E) Performance comparison over 250 episodes of a trained spiking neural reservoir on Loihi (backbone stiffness: 125 kPa; reservoir size: 2,048 neurons), a neural reservoir running on traditional silicon using nonspiking artificial neurons, and both nonspiking and spiking versions of LSTM and FF networks. (F) Energy use of the spiking reservoir running on the Intel Loihi chip compared with a nonspiking reservoir running on several traditional silicon CPUs (Intel Xeon W-2665, ARM Cortex A-53, and ARM Cortex A-76). The Cortex A-76 is the CPU used in Raspberry Pi 5s, while the Cortex A-53 is used in the Raspberry Pi Zero W2, making them representative of low-power optimized CPUs. While ARM-based CPUs exhibit lower power draw during evaluation, their lower clock speeds resulted in longer evaluation times, negating much of their advantage over the Intel Xeon chip when energy per inference is considered for larger reservoirs. (G) Initial 30k episodes of learning performance of neural reservoirs ($n = 1,024$) controlling arms of decreasing backbone stiffness tasked with reaching through a cluttered nest of obstacles to a fixed target (SI Appendix for expanded results). Video of the performance of final trained policies for all stiffness levels is available in Movie S6. (H) Timelapse front/side views of a spiking reservoir guiding a soft arm through unstructured obstacles to reach a target (backbone stiffness: 125 kPa). Shaded circles around target correspond to shaded bands in part (I) of distance from target. (I) Performance of a policy trained on a reference nest of obstacles (for a 125 kPa backbone), when evaluated on perturbed nest configurations. Perturbations are generated by randomly shifting (from the reference nest) each obstacle by a portion of its radius.

is found to eventually achieve better performance. In all cases though, throughout training, arms are observed to extensively exploit (instead of avoiding) solid boundaries, leaning against them to passively reconfigure and favorably redirect the tip (Movie S6). This facilitates learning and control, as confirmed by the fact that the target can be reached even by the softest arm (Movie S6), despite being the most challenging to coordinate. Thus, our RC approach is found to naturally recover a hallmark of mechanical intelligence, whereby passive mechanics and the environment are leveraged to simplify the task at hand.

We next challenge the policy trained on the nest configuration of Fig. 4H to reach the same target through a perturbed obstacle

arrangement, thereby probing whether the arm's compliance can enhance control robustness (Fig. 4I). Perturbed nests are generated by shifting each obstacle within its cross-sectional plane in a random direction by a specified fraction of its radius (within each evaluation, all obstacles are displaced by the same radius fraction, but each receives a unique random direction). We quantify the arm's ability to reach through these new nests using the minimum tip-to-target distance achieved during each episode. As shown in Fig. 4I, the originally learned policy (critically, without any additional training) continues to reach (or closely approach) the target for perturbations up to 200% of the obstacle radius. Beyond this level of perturbation, the

arm is occasionally obstructed by the rearranged obstacles, yet still succeeds in reaching the target more than 50% of the time even with obstacle shifts as large as 500% of the radius. These results highlight how arm compliance enables robust interaction with the environment. The soft arm is indeed observed to make contact with, conform to, and push through displaced obstacles, effectively offloading part of the control burden to passive, interfacial dynamics (Movie S7). Such interactions are challenging to model with conventional control approaches and underscore the advantages of combining learning-based control—such as the reservoir-based method proposed here—with soft robots.

Discussion. In this work, we tackle the challenging problem of learning to control a heterogeneous, soft bio-hybrid arm. We consider a musculoskeletal architecture designed to encompass a range of biomechanically relevant elements, and modeled it via assemblies of Cosserat rods. The physical infrastructure is matched with a neural infrastructure (the reservoir), to encode the arm's sensing and spatiotemporal dynamics. This representation is coupled with reinforcement learning to achieve muscle coordination and allow the arm to track and reach targets.

Our approach is demonstrated to be effective, outperforming classic neural network methods. In particular, the synergy between neural and physical dynamics is observed to produce control policies that minimize (relative to FF and LSTM) the arm's bending and kinetic energies, across a range of material stiffnesses. In contrast, FF and LSTM approaches are found unable to cope with material property changes. Parallel self-modeling on the same reservoir is also demonstrated for robust control, in the case of unavailable or corrupted sensory information. Further, spiking neural reservoirs on neuromorphic hardware are shown to deliver up to $\sim 75\times$ and $\sim 45\times$ energy savings relative to nonspiking reservoirs on conventional and high-efficiency CPUs, while conserving control performance. Spiking reservoirs are finally employed to robustly drive the arm through a set of unstructured obstacles, taking advantage of solid boundaries and compliance, a hallmark of mechanical intelligence.

The capacity of neural reservoirs to synthesize compliant dynamics presents much potential for soft robotic control, combining learning, versatility, adaptivity, and robustness to mechanical changes or environmental disturbances. Higher-order situational information (velocities or accelerations) can be extracted with minimal additional computational costs (parallel maps running on the same reservoir), reducing onboard sensing and energy requirements. This aspect is compounded by the use of spiking neural reservoirs on neuromorphic hardware, to deliver significant energy savings, a necessity in many robotic applications.

These results not only advance soft robotic control, but may also further the use of biological constructs in robotics and engineering, with both spike-activated cultured muscles (6–8, 42) and neural tissue reservoirs (37, 39–41) amenable to integration within this computing paradigm.

Materials and Methods

Modeling a Neuromuscular Arm. We model the muscular arm as an assembly of Cosserat rods, which are slender, one-dimensional elastic structures that can undergo all modes of deformation: bending, twisting, stretching, and shearing. Representing the backbone and muscle groups as Cosserat rods entails a number of advantages. Assembled together they naturally capture the heterogeneity of muscular architectures while the Cosserat formulation

can be extended to account for nonlinear material properties, connectivity, active stresses, internal pressure, and environmental loads. The numerical implementation of Cosserat rods is computationally efficient as they accurately capture large 3D deformations through a one-dimensional representation, alleviating time-consuming remeshing difficulties and compute costs of 3D elasticity.

Dynamics of a single Cosserat rod. An individual Cosserat rod is described by its center line position $\bar{\mathbf{x}}(s, t) \in \mathbb{R}^3$ and an oriented reference frame of (row-wise) orthonormal directors $\mathbf{Q}(s, t) \in \mathbb{R}^{3 \times 3} = [\bar{\mathbf{d}}_1, \bar{\mathbf{d}}_2, \bar{\mathbf{d}}_3]^{-1}$ along its length $s \in [0, L(t)]$, where $L(t)$ is the current length, for all time $t \in \mathbb{R} \geq 0$. Any vector defined in the global *lab frame* ($\bar{\mathbf{v}}$) can be transformed into the *local frame* ($\mathbf{v}$) via $\mathbf{v} = \mathbf{Q}\bar{\mathbf{v}}$ and from the local to the lab frame via $\bar{\mathbf{v}} = \mathbf{Q}^T\mathbf{v}$. For an unshearable and inextensible rod, $\bar{\mathbf{d}}_3$ is parallel to the rod's local tangent ($\partial_s \bar{\mathbf{x}} = \bar{\mathbf{x}}_s$), and $\bar{\mathbf{d}}_1$ (normal) and $\bar{\mathbf{d}}_2$ (binormal) span the rod's cross-section. However, under shear or extension, the rod's tangent direction $\bar{\mathbf{x}}_s$ and $\bar{\mathbf{d}}_3$ are no longer the same, with the difference represented by the shear strain vector $\epsilon = \mathbf{Q}(\bar{\mathbf{x}}_s - \bar{\mathbf{d}}_3)$. The gradient of the directors ($\bar{\mathbf{d}}_j$) with respect to the rod's length is defined by the curvature vector $\bar{\kappa}(s, t) \in \mathbb{R}^3$ through the relation $\partial_s \bar{\mathbf{d}}_j = \bar{\kappa} \times \bar{\mathbf{d}}_j$. In the local frame ($\kappa = \mathbf{Q}\bar{\kappa}$), the components of the curvature vector $\kappa = [\kappa_1, \kappa_2, \kappa_3]$ relate to bending (κ_1 and κ_2) and twisting (κ_3) of the rod. Similarly, the gradient of the directors with respect to time is defined by the angular velocity vector $\bar{\omega}(s, t) \in \mathbb{R}^3$ through the relation $\partial_t \bar{\mathbf{d}}_j = \bar{\omega} \times \bar{\mathbf{d}}_j$. The linear velocity of the center-line is $\bar{\mathbf{v}}(s, t) \in \mathbb{R}^3 = \partial_t \bar{\mathbf{x}}$ while the second area moment of inertia $\mathbf{I}(s, t) \in \mathbb{R}^{3 \times 3}$, cross-sectional area $A(s, t) \in \mathbb{R}$, and density $\rho(s) \in \mathbb{R}$ are defined based on the rod's material properties. The dynamics of a Cosserat rod are then described as (58)

$$\partial_t^2 (\rho A \bar{\mathbf{x}}) = \partial_s (\mathbf{Q}^T \mathbf{n}) + \bar{\mathbf{f}}, \quad [1]$$

$$\partial_t (\rho \bar{\omega}) = \partial_s \tau + \kappa \times \tau + (\mathbf{Q} \bar{\kappa}_s \times \mathbf{n}) + (\rho \bar{\omega}) \times \omega + \mathbf{Q} \bar{\mathbf{c}}, \quad [2]$$

where Eqs. 1 (lab frame) and 2 (local frame) represent the change of linear and angular momentum at every cross-section, respectively, $\mathbf{n}(s, t) \in \mathbb{R}^3$ and $\tau(s, t) \in \mathbb{R}^3$ are internal forces and couples, respectively, developed due to elastic deformations and muscle contractions while $\bar{\mathbf{f}}(s, t) \in \mathbb{R}^3$ and $\bar{\mathbf{c}}(s, t) \in \mathbb{R}^3$ capture external forces and couples applied to the arm, respectively.

The above continuous representation is discretized into $(n_{\text{elem}} + 1)$ nodes of position $\bar{\mathbf{x}}_i$ that are connected by n_{elem} cylindrical elements. Linear displacements are determined by the internal and external forces acting at the nodes, while rotations are accounted for via couples applied to the cylindrical elements. Energy losses due to internal friction and viscoelastic effects are captured through use of Rayleigh potentials. The dynamic behavior of a rod is then computed by integrating the discretized set of equations, along with appropriate boundary conditions, in time via a second-order position Verlet scheme (58). In this study, we used PyElastica (59), which is an open-source, Python-based implementation of this numerical scheme.

Assembling a muscular arm. We assemble our muscular arm by arranging active and passive rods into a representative muscular architecture. The arm consists of an elastic backbone (modeled as a single rod) onto which sixteen muscle-tendon units (each modeled as a single rod) are attached around the outside for a total of seventeen rods. This mechanical connectivity enables the arm to translate the one-dimensional internal contraction forces generated by individual muscles into the three-dimensional dynamic motions of the arm as a whole. A description of the arm's geometry and material parameters is provided in SI Appendix.

Modeling the elastic backbone. The backbone is modeled as a constant cross-section, elastic rod. For a linear elastic material such as the backbone, the internal forces $\mathbf{n} = [n_1, n_2, n_3]$ are proportional to the shear strain of the rod $\mathbf{n} = \mathbf{S}\epsilon$ where $\mathbf{S} = \text{diag}(\alpha_c GA, \alpha_c GA, EA)$ is the rod's shear/stretch stiffness matrix, E is the rod's Young's modulus, G is the rod's shear modulus, and α_c is the Timoshenko shear correction factor (58). We additionally model internal elastic torques τ using the linear relationship $\tau = \mathbf{B}\kappa$ where $\mathbf{B} \in \mathbb{R}^{3 \times 3} = \text{diag}(EI_1, EI_2, GI_3)$ is the rod's bend/twist stiffness matrix, and I_1, I_2 , and I_3 are

the rod's second moments of inertia about $\bar{\mathbf{d}}_1$, $\bar{\mathbf{d}}_2$, and $\bar{\mathbf{d}}_3$, respectively. Finally, zero-displacement and zero-rotation boundary conditions are defined on the node and element of the backbone nearest to the base, respectively, to anchor the arm.

Modeling the muscle-tendon unit. We model muscle-tendon units based on frog semitendinosus muscle, in line with previous bio-hybrid demonstrations (97). Biological muscles actively generate internal forces that cause them to axially contract while both muscle and tendon tissues exhibit hyperelastic passive behavior when stretched. Both effects render a linear stress-strain treatment of the Cosserat rod's axial stretch inaccurate. Instead, we model the axial component (n_3) of the internal force vector $\mathbf{n}$ by considering the active $\sigma_a(s, t) \in \mathbb{R}$ and passive $\sigma_p(s, t) \in \mathbb{R}$ axial stresses developed along the muscle $n_3 = A(\sigma_a^m + \sigma_p^m)$ and the passive response of the tendon $n_3 = A(\sigma_p^t)$ while modeling its shear components (n_1, n_2) using the above presented linear elastic formulation. Internal elastic torques are represented using a similar approach as the elastic backbone, though we note that the nonlinear effects of the muscle's axial stretch extends to encompass angular momentum via the third term in Eq. 2. Material properties (provided in *SI Appendix*) of the muscle-tendon unit are based on the sliding-filament model of muscle (98, 99) and experimental measurements (100).

Patterning muscle-tendon units onto backbone. We pattern the 16 muscle-tendon units onto the elastic backbone. Units are arranged in four layers. Each layer consists of four units, organized in two agonist-antagonist pairs oriented orthogonal to each other. Neighboring pairs overlap lengthwise by 50% while being rotated 45° relative to each other to avoid intersecting. Muscle-tendon units are arranged on the elastic backbone's surface such that the muscle-tendon unit's center-line lies one (backbone) radius away from the backbone's center-line. The arm in its rest configuration presents no muscle activation or residual stresses.

To join the muscle-tendon units to the backbone, we define zero-displacement boundary conditions at the ends of the muscle-tendon unit relative to point on the outer surface of the backbone to model to first order the enthesis point where tendon and bone join together. The location of these connections a radial distance from the backbone's center-line then generates a bending torque on the backbone then the muscle-tendon unit linear shortens when activated. Beyond these primary connections, we also implement distributed displacement-force boundary conditions to lightly adhere the muscle-tendon unit to the backbone during bending motions (52, 55), analogous to the effect of connective fascia tissue that helps maintain muscle organization.

3D tracking environment. The muscular arm defined above is challenged to track a target moving in 3D space along to a smoothly varying, random trajectory. The target trajectory $\bar{\mathbf{x}}_t(t) = [\bar{x}_1, \bar{x}_2, \bar{x}_3]$ is defined at the beginning of each episode with each spatial component of the trajectory generated according to

$$\bar{x}_j(t) = b_j \sin(f_{j,1} 2\pi t) \sin(f_{j,2} 2\pi t) \sin(f_{j,3} 2\pi t) \quad \forall j \in [1...3], \quad [3]$$

where $f_{j,i} \in \mathbb{R}$ is a randomly selected constant from the interval [0.5, 1.0] drawn at the beginning of each episode and b_j is a velocity scaling constant, whose sign is also randomly selected each episode. This structure of Eq. 3 results in a smoothly varying trajectory of the target while the random selection of $f_{j,i}$ each episode renders it not possible for the controller to simply memorize a target trajectory. The state and muscle activations of the arm are updated at a 4 Hz control frequency ($\Delta t = 250$ ms), leading to a mapping between the physical time t of the simulation environment and the control time step number n of $n = [4t]$. At each timestep, muscle activations are directly mapped from the action space $\mathbf{a}^n = [a_i] \in \mathbb{R}^{16}$, $a_i \in [-\infty, \infty]$ to an individual muscle activation level $\alpha_i(t) = (\tanh(a_i(t)) + 1)/2 \in [0, 1]$ which is applied for 0.25 s (until the next control time step). The arm's state consists of the current 3D target location $\bar{\mathbf{x}}_t(t)$ and the current averaged normalized curvature of the arm $\mathbf{K}_i(t)$ over $i = 4$ equidistant intervals in the rod's local normal and binormal directions

$$\mathbf{K}_i(t) = \frac{L}{2\pi} \frac{\int_{d_i}^{d_{i+1}} \kappa(s, t) ds}{d_{i+1} - d_i} \quad \forall i \in [0...3], \quad [4]$$

where $\kappa(s, t) \in \mathbb{R}^2$ is the local normal and binormal curvature of the arm, $d = \{kL/4 | k \in [0..4]\}$, and L is the length of the arm. This results in a state

$\mathbf{s}^n = [\bar{\mathbf{x}}_t(t), \mathbf{K}(t)] \in \mathbb{R}^{3+8}$. The reward is the square of the Euclidean distance between the tip of the arm $\bar{\mathbf{x}}(L, t)$ and the 3D moving target $\bar{\mathbf{x}}_t(t)$ integrated over the control time step

$$r_n = \int_t^{t+\Delta t} \|\bar{\mathbf{x}}(L, t') - \bar{\mathbf{x}}_t(t')\|^2 dt'. \quad [5]$$

The integration of the reward over the time step ensures the tip of the arm continuously follows the target trajectory over the entire time step. Each episode is terminated after 100 s.

Unstructured nest environment. In Fig. 4, the arm is challenged to reach through an unstructured nest of rigid cylindrical obstacles toward a stationary target. Here, the target and nest are stationary and fixed in the same location for all episodes. Between the target and the arm's initial configuration, an unstructured nest of cylindrical obstacles is placed, which the arm must navigate through to reach the target on the other side. The location of the obstacles is the same for all episodes and each episode is terminated after 5 s. External contact forces between the arm and the nest are implemented (described in *SI Appendix*) as well as contact between the arm and the floor, allowing the arm to interact with its external environment. The state, action, and reward of this environment are the same as with the 3D tracking environment, but here successful control of the neuromuscular arm requires addressing a more complex physical environment by learning to interact with and account for the presence of solid obstacles that restrict reaching.

Neural Reservoir Formulation. We consider two formulations of a neural reservoir: artificial neuron reservoirs and spiking neuron reservoirs. Artificial neuron reservoirs consist of a discrete-time, recurrent network of artificial neurons typically used in machine learning applications while spiking neuron reservoirs utilize a continuous-time, spike-based approach.

Neural reservoir with artificial neurons. Neural reservoirs using artificial neurons are implemented using an Echo State Network (ESN) approach (34). The reservoir consists of a large pool of n sparsely and recurrently connected artificial neurons. Recurrent connections within the reservoir are represented using the reservoir connection matrix $W_r \in \mathbb{R}^{n \times n}$ while connections into the reservoir from the input state (of size s) are represented using the input connection matrix $W_i \in \mathbb{R}^{n \times s}$. The state of the reservoir $\mathbf{u}_r^n$ at time step n is then

$$\mathbf{u}_r^n = f(W_i \mathbf{s}^n + W_r \mathbf{u}_r^{n-1}), \quad [6]$$

where $f(\cdot)$ is a nonlinear activation function. Here, a hyperbolic tangent function (TanH) is used. Notably, the connection weight matrices are fixed at initialization. No adjustment to their weights is made during the learning process. Thus, their proper initialization is critical to the information processing capability of the reservoir. Following the ESN approach, both the reservoir connection matrix W_r and the input connection matrix W_i are randomly initialized from a normal distribution with zero mean, a SD of 0.50, and a sparse connection density ρ of 0.10. The reservoir connection matrix is then normalized based on its spectral radius λ (the magnitude of its largest eigenvector) to ensure the echo state property, which is that information from initial conditions will asymptotically disappear over time. To achieve the echo state property it is generally sufficient (though not necessary) to enforce $\lambda < 1$ (34).

The mapping from the reservoir state $\mathbf{u}_r^n$ to the output (in this case action, with size j) is determined according to the linear output mapping $W_o \in \mathbb{R}^{j \times n}$ according to

$$a^n = W_o \mathbf{u}_r^n. \quad [7]$$

The linear map W_o can be learned using a variety of different algorithms. For the control policy, the unsupervised reinforcement learning PPO algorithm was used while for the parallel output maps of Fig. 3, supervised learning algorithms such as ridge regression were used.

Neural reservoir with spiking neurons. To implement a neural reservoir using spiking neurons, we use leaky-integrate and fire (LIF) neurons (101) whose dynamics are governed by the differential equation

$$\tau_m \frac{dv}{dt} = (v_r - v(t)) + RI(t), \quad [8]$$

where τ_m is the membrane constant of the RC circuit simulating voltage leakage, R is the resistance, $I(t)$ is the input current, $v(t)$ is the output voltage, and v_r is the reset membrane potential. When $v(t)$ exceeds some threshold voltage level ($v(t) > v_t$), a spike in voltage occurs at time t followed by the neuron returning to v_r which it is held at for some refractory time τ_r .

We begin by converting the discrete time, continuous state input of the soft muscular arm to a continuous time, spike-based input compatible with a spiking neural reservoir. Each state variable is presented as current into two LIF input neurons for a time of Δ (which we decouple from the control time Δt). These two neurons have opposite tuning curves such that one will positively rate encode increasing values while the other will rate encode decreasing values. The resulting spike trains are multiplied by input weights W_i and fed into the spiking reservoir. The input weights W_i are randomly drawn from a uniform distribution between -1 and 1 . The spiking neural reservoir consists of a set of N LIF neurons recurrently connected by a sparse set of weights W_r . The recurrent connection weights have a density of 0.1 and are drawn from the normal distribution $\mathcal{N}(0.0, 0.5^2)$. LIF neuron properties such as refractory period and membrane voltage decay as well as reservoir connection properties were optimized using a CMA evolutionary search algorithm (102) as detailed in *SI Appendix*. For each control time step, the number of spikes over the time interval Δ is counted for each reservoir neuron. This spike count is then treated as the output of the reservoir for that control time step. This approach then enables the same RL algorithms as used to train the artificial neural reservoir to be employed to learn the output weights W_o to control the arm.

This neural reservoir setup was deployed on Intel's Kapoho Bay device (35), which contains 2 Loihi chips. The reservoir was defined using the Nengo Python package (103) and its NengoLoihi extension. The input spike conversion layer and output spike counting layer are implemented off-chip, with the communication between the CPU host and the Loihi chip occurring only via spike communication.

Energy efficiency. The energy usage of neural reservoirs consisting of either artificial neurons or spiking neurons was analyzed by measuring the average energy consumption for each inference, or control time step. For a neural reservoir with artificial neurons, this refers to a single evaluation of the system defined in Eq. 6 while for spiking neurons it refers to the presentation of the input state for a time Δ . Energy consumption of an artificial neural reservoir was measured on three different traditional CPUs, an ARM Cortex A-53, an ARM Cortex A-76, and an Intel Xeon W-2665. ARM-based CPUs are well established for their low power consumption. The ARM Cortex A-76 was used in a Raspberry Pi 5 while the ARM Cortex A-53 was used in a Raspberry Pi Zero W2. Power consumption of the ARM-based CPUs was measured using a Power-Z KM003C (ChargerLAB) power meter to determine the relative increase in power draw of the CPU when evaluating the reservoir compared to an idle state. Power use on the Intel Xeon W-2665 CPU was measured using the pyRAPL Python package to identify the running power with idle power subtracted. Energy usage on Loihi was measured using utilities provided by Intel's NxSDK API for Loihi to similarly measure running power.

Reinforcement Learning Integration. To learn the mapping from the neuromuscular arm's state to the action required to control it, reinforcement learning (RL) is used. We employ the state-of-the-art, model-free, on-policy PPO algorithm to optimize W_o (83) based on the open-source Stable Baselines 3 implementation (104). When used with feed-forward and LSTM network architectures, the environment state $\mathbf{s}^n$ was used as the input state into the artificial neural network in the usual manner. When integrated with the neural reservoir, the current reservoir state $\mathbf{u}_r^n$ is used as the input state to the PPO algorithm, with the PPO algorithm then tasked with learning the single layer, linear mapping W_o from the reservoir state to the optimal muscle action. When a neural reservoir consisting of artificial neurons is used, the reservoir state is directly used as the input into the PPO algorithm while for the spiking neural reservoir, output spikes are accumulated and counted over each control time step to determine an analogous reservoir state that can be then directly used by the PPO algorithm.

Training and evaluation. All network architectures were trained using the same approach. For the 3D tracking case (Figs. 1, 2, and 4 B and C), the policy was trained using 120 parallel environments. The policy was updated every 4,800 time steps (1,200 s at 4 Hz), resulting in each environment collecting 10 s of data

between updates. Each environment was reset every 100 s with a new trajectory generated and the arm initiated in a straight configuration. Policies were trained for either two or four million time steps and five separate policies with random initial seeds were trained for each case. Trained policies were evaluated on new, unseen trajectories for up to 100 s (400 time steps). Hyperparameter tuning for all algorithms was performed as detailed in *SI Appendix*. All training and hyperparameter tuning was performed on the Pittsburgh Supercomputing Center's Bridges 2 cluster.

For the spiking neural reservoir, only one Kapoho Bay device was available, so NengoLoihi's emulation environment, which emulates Loihi hardware on CPUs, was used to train using the same 120 parallel environment approach for the 3D tracking case (Fig. 4 B and C). For the unstructured obstacle case (Fig. 4 E and F), 120 parallel environments were also used (using the NengoLoihi emulator), resulting in each environment having a 5s episode, however, environments were then reset after each episode. All training using the Loihi emulation environment was performed on the Pittsburgh Supercomputing Center's Bridges 2 cluster. Trained policies were then evaluated on Loihi hardware to determine performance and energy usage. All evaluation and visualization results reported in Fig. 4 and in *SI Appendix* videos were run on Loihi hardware.

Spiking LSTM and feedforward networks. The open-source Python package SNNtorch (105), which integrates spiking neurons and surrogate-gradients with PyTorch, was used to implement spiking versions of the LSTM and feedforward architectures. This allowed use of the same PPO algorithm for training and all environment and training parameters were directly adopted from the relevant nonspiking case. Specifically, the same PPO hyperparameters and architectures as the nonspiking versions of the LSTM and FF networks were used. Details of spiking neuron-specific hyperparameter selection and additional training results are available in *SI Appendix*.

Kinetic and Bending Energy Measurements. To report the kinetic and bending energy measurements in Fig. 2 B and C, the average kinetic and bending energy of the arm's backbone over an episode of length T was measured as

$$\bar{e}_i = \frac{\int_0^T e_i(t) dt}{\int_0^T dt}, \quad [9]$$

where $e_i \in \{e_k, e_b\}$. Here e_k is the kinetic energy of the backbone and e_b is the elastic bending energy of the rod. The kinetic energy of the rod at each time is

$$e_k(t) = \frac{1}{2} \int_0^L \rho A(\mathbf{v} \cdot \mathbf{v}) ds, \quad [10]$$

where ρ is the local density, A is the local cross-sectional area, and $\mathbf{v}$ is the local velocity. The bending energy of the rod at each time is

$$e_b(t) = \frac{1}{2} \int_0^L \boldsymbol{\kappa} \cdot \boldsymbol{\tau} ds, \quad [11]$$

where $\boldsymbol{\kappa}$ is the local curvature of the rod, $\boldsymbol{\tau} = \mathbf{B}\boldsymbol{\kappa}$ is the local internal elastic torque vector, and $\mathbf{B}$ is the rod's bend/twist stiffness matrix. Arm dynamics from trained policies were recorded for 100 unique episodes (100 s each) and the average energy of each episode was calculated to determine the distribution of values reported in Fig. 2B.

Training of Parallel Maps. Fig. 3 shows the use of parallel maps to increase the information extracted from the neural reservoir. To generate these parallel maps, an RL policy trained to track a moving target was recorded for 100 s over 400 different target trajectories. The state, action, reservoir state, and environment were recorded. These data were then used as the training data (using an 80/20 training/testing data split) for parallel maps using a supervised learning method via ridge regression. For target correction (Fig. 3D), the parallel map was evaluated every time step to predict the 3D position the target would be one time step in the future and used in place of the target's true position in the state vector as required.

Data, Materials, and Software Availability. This work utilized the open-source Elastica software package available online at <https://github.com/GazzolaLab/PyElastica> (59). Code for the setup and simulation of the neural reservoir controlled biohybrid soft arm demonstrated here is available at <https://github.com/GazzolaLab/SoftArmRC> (106).

ACKNOWLEDGMENTS. This study was jointly funded by ONR MURI N00014-19-1-2373 (M.G.), ONR N00014-22-1-2569 (M.G.), NSF EFRI C3 SoRo #1830881 (M.G.), NSF CAREER #1846752 (M.G.), NSF Expeditions in Computing "Mind in Vitro" #2123781 (M.G.), NSF EFRI BEGIN OI #2422340 (N.N.), and a Beckman Institute Postdoctoral Fellowship (N.N.). Computational support was provided

by the Bridges2 supercomputer at the Pittsburgh Supercomputing Center through allocation TG-MCB190004 from the Extreme Science and Engineering Discovery Environment (XSEDE; NSF Grant ACI-1548562) and by Virginia Tech's Advanced Research Computing through use of their Tinker Cliffs and Owl clusters.

Author affiliations: ^aDepartment of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061; ^bDepartment of Mechanical Science and Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801; ^cThe Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801; ^dNational Center for Supercomputing Applications, University of Illinois Urbana-Champaign, Urbana, IL 61801; and ^eCarl R. Woese Institute for Genomic Biology, University of Illinois Urbana-Champaign, Urbana, IL 61801

1. L. Ricotti *et al.*, Biohybrid actuators for robotics: A review of devices actuated by living cells. *Sci. Rob.* **2**, eaag0495 (2017).
2. Y. Morimoto, H. Onoe, S. Takeuchi, Biohybrid robot powered by an antagonistic pair of skeletal muscle tissues. *Sci. Rob.* **3**, eaat4440 (2018).
3. T. Li, S. Takeuchi, Advancing biohybrid robotics: Innovations in contraction models, control techniques, and applications. *Biophys. Rev.* **6**, 011304 (2025).
4. X. Ren, Y. Morimoto, S. Takeuchi, Biohybrid hand actuated by multiple human muscle tissues. *Sci. Rob.* **10**, eadr5512 (2025).
5. S. Park *et al.*, Phototactic guidance of a tissue-engineered soft-robotic ray. *Science* **353**, 158–162 (2016).
6. G. J. Pagan-Diaz *et al.*, Simulation and fabrication of stronger, larger, and faster walking biohybrid machines. *Adv. Funct. Mater.* **28**, 1801145 (2018).
7. O. Aydin *et al.*, Neuromuscular actuation of biohybrid motile bots. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 19841–19847 (2019).
8. Y. Kim *et al.*, Remote control of muscle-driven miniature robots with battery-free wireless optoelectronics. *Sci. Rob.* **8**, eadd1053 (2023).
9. O. Yasa *et al.*, An overview of soft robotics. *Annu. Rev. Control. Rob., Auton. Syst.* **6**, 1–29 (2023).
10. P. Polygerinos *et al.*, Soft robotics: Review of fluid-driven intrinsically soft devices: Manufacturing, sensing, control, and applications in human-robot interaction. *Adv. Eng. Mater.* **19**, 1700016 (2017).
11. M. Cianchetti, C. Laschi, A. Menciassi, P. Dario, Biomedical applications of soft robotics. *Nat. Rev. Mater.* **3**, 143–153 (2018).
12. G. Chowdhary, M. Gazzola, G. Krishnan, C. Soman, S. Lovell, Soft robotics as an enabling technology for agroforestry practice and research. *Sustainability* **11**, 6751 (2019).
13. M. Russo *et al.*, Continuum robots: An overview. *Adv. Intell. Syst.* **5**, 2200367 (2023).
14. C. Della Santina, C. Duriez, D. Rus, Model-based control of soft robots: A survey of the state of the art and open challenges. *IEEE Control. Syst. Mag.* **43**, 30–65 (2023).
15. T. George Thuruthel, Y. Ansari, E. Falotico, C. Laschi, Control strategies for soft robotic manipulators: A survey. *Soft Rob.* **5**, 149–163 (2018).
16. Z. Chen *et al.*, Data-driven methods applied to soft robot modeling and control: A review. *IEEE Trans. Autom. Sci. Eng.* **22**, 2241–2256 (2024).
17. E. Falotico *et al.*, Learning controllers for continuum soft manipulators: Impact of modeling and looming challenges. *Adv. Intell. Syst.* **7**, 2400344 (2025).
18. G. Mengaldo *et al.*, A concise guide to modelling the physics of embodied intelligence in soft robotics. *Nat. Rev. Phys.* **4**, 595–610 (2022).
19. R. Pfeifer, J. Bongard, *How the Body Shapes the Way We Think: A New View of Intelligence* (MIT Press, Cambridge, Massachusetts, 2006).
20. N. T. Ulrich, Grasping with mechanical intelligence (Tech. Rep., 1988).
21. T. Wang *et al.*, Mechanical intelligence simplifies control in terrestrial limbless locomotion. *Sci. Rob.* **8**, eadi2243 (2023).
22. A. Alù *et al.*, Roadmap on embodying mechano-intelligence and computing in functional materials and structures. *Smart Mater. Struct.* **34**, 063501 (2025).
23. A. J. Ijspeert, Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks* **21**, 642–653 (2008).
24. P. Ramdaya, A. J. Ijspeert, The neuromechanics of animal locomotion: From biology to robotics and back. *Sci. Rob.* **8**, eadg0279 (2023).
25. O. Aydin *et al.*, Development of 3D neuromuscular bioactuators. *APL Bioeng.* **4**, 016107 (2020).
26. H. Tetsuka, S. Gobbi, T. Hatanaka, L. Pirrami, S. Shin, Wirelessly steerable bioelectronic neuromuscular robots adapting neurocardiac junctions. *Sci. Rob.* **9**, eado0051 (2024).
27. H. Min *et al.*, Optogenetic neuromuscular actuation of a miniature electronic biohybrid robot. *Sci. Rob.* **10**, eadu5830 (2025).
28. L. F. Seoane, Evolutionary aspects of reservoir computing. *Philos. Trans. R. Soc. B* **374**, 20180377 (2019).
29. W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
30. L. Gonon, J. P. Ortega, Reservoir computing universality with stochastic inputs. *IEEE Trans. Neural Networks Learn. Syst.* **31**, 100–112 (2019).
31. M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3**, 127–149 (2009).
32. W. Maass, P. Joshi, E. Sontag, Principles of real-time computing with feedback applied to cortical microcircuit models. *Adv. Neural Inf. Process. Syst.* **18**, 835–842 (2005).
33. G. Tanaka *et al.*, Recent advances in physical reservoir computing: A review. *Neural Networks* **115**, 100–123 (2019).
34. H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note (Tech. Rep., German National Research Center for Information Technology GMD, Bonn, Germany, 2001), vol. 148, p. 13.
35. M. Davies *et al.*, Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
36. P. A. Merolla *et al.*, A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
37. X. Zhang *et al.*, Mind in vitro platforms: Versatile, scalable, robust, and open solutions to interfacing with living neurons. *Adv. Sci.* **11**, 2306826 (2024).
38. J. P. Eckmann *et al.*, The physics of living neural networks. *Phys. Rep.* **449**, 54–76 (2007).
39. J. Soriano, Neuronal cultures: Exploring biophysics, complex systems, and medicine in a dish. *Biophysica* **3**, 181–202 (2023).
40. H. Cai *et al.*, Brain organoid reservoir computing for artificial intelligence. *Nat. Electron.* **6**, 1032–1039 (2023).
41. G. Upadhyay *et al.*, Living neurosheets: Engineering readily deployable neural architectures. *bioRxiv* [Preprint] (2025). <https://doi.org/10.64898/2025.11.28.691240> (Accessed 7 April 2026).
42. J. Wang *et al.*, Computationally assisted design and selection of maneuverable biological walking machines. *Adv. Intell. Syst.* **3**, 2000237 (2021).
43. B. J. Kagan *et al.*, In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. *Neuron* **110**, 3952–3969 (2022).
44. M. Duduta, E. Hajiesmaili, H. Zhao, R. J. Wood, D. R. Clarke, Realizing the potential of dielectric elastomer artificial muscles. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 2476–2481 (2019).
45. J. Wang, D. Gao, P. S. Lee, Recent progress in artificial muscles for interactive soft robotics. *Adv. Mater.* **33**, 2003088 (2021).
46. P. Won, S. H. Ko, C. Majidi, A. W. Feinberg, V. A. Webster-Wood, Biohybrid actuators for soft robotics: Challenges in scaling up. *Actuators* **9**, 96 (2020).
47. J. Gray, Studies in the mechanics of the tetrapod skeleton. *J. Exp. Biol.* **20**, 88–116 (1944).
48. T. J. Roberts, R. L. Marsh, P. G. Weyand, C. R. Taylor, Muscular force in running turkeys: The economy of minimizing work. *Science* **275**, 1113–1115 (1997).
49. W. M. Kier, K. K. Smith, Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats. *Zool. J. Linnean Soc.* **83**, 307–324 (1985).
50. B. C. Jayne, Muscular mechanisms of snake locomotion: An electromyographic study of lateral undulation of the Florida banded water snake (*Nerodia fasciata*) and the yellow rat snake (*Elaphe obsoleta*). *J. Morphol.* **197**, 159–181 (1988).
51. K. Y. Lee *et al.*, An autonomously swimming biohybrid fish designed with human cardiac biophysics. *Science* **375**, 639–647 (2022).
52. A. Tekinalp *et al.*, Topology, dynamics, and control of a muscle-architected soft arm. *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2318769121 (2024).
53. P. L. Gribble, L. I. Mullin, N. Cothros, A. Mattar, Role of cocontraction in arm movement accuracy. *J. Neurophysiol.* **89**, 2396–2405 (2003).
54. A. D. Koelewijn, A. J. Van Den Bogert, Antagonistic co-contraction can minimize muscular effort in systems with uncertainty. *PeerJ* **10**, e13085 (2022).
55. X. Zhang, F. K. Chan, T. Parthasarathy, M. Gazzola, Modeling and simulation of complex dynamic musculoskeletal architectures. *Nat. Commun.* **10**, 1–12 (2019).
56. G. M. Spinks, N. D. Martino, S. Naficy, D. J. Shepherd, J. Foroughi, Dual high-stroke and high-work capacity artificial muscles inspired by DNA supercoiling. *Sci. Rob.* **6**, eabf4788 (2021).
57. S. S. Antman, *Nonlinear Problems of Elasticity* (Springer, 2005).
58. M. Gazzola, L. Dudte, A. McCormick, L. Mahadevan, Forward and inverse problems in the mechanics of soft filaments. *R. Soc. Open Sci.* **5**, 171628 (2018).
59. A. Tekinalp *et al.*, GazzolaLab/PyElastica. Zenodo. <https://zenodo.org/records/16995297>. Deposited 20 February 2023.
60. X. Zhang, N. Naughton, T. Parthasarathy, M. Gazzola, Friction modulation in limbless, three-dimensional gaits and heterogeneous terrains. *Nat. Commun.* **12**, 6076 (2021).
61. A. Tekinalp, Y. Bhosale, S. Cui, F. K. Chan, M. Gazzola, Self-propelling, soft, and slender structures in fluids: Cosserat rods immersed in the velocity-vorticity formulation of the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.* **440**, 117910 (2025).
62. A. Porat, A. Tekinalp, Y. Bhosale, M. Gazzola, Y. Meroz, On the mechanical origins of waving, coiling and skewing in *Arabidopsis thaliana* roots. *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2312761121 (2024).
63. Y. Bhosale *et al.*, Micromechanical origin of plasticity and hysteresis in nestlike packings. *Phys. Rev. Lett.* **128**, 198003 (2022).
64. N. Weiner, Y. Bhosale, M. Gazzola, H. King, Mechanics of randomly packed filaments—The "bird nest" as meta-material. *J. Appl. Phys.* **127**, 050902 (2020).
65. A. Khairnar *et al.*, "Physical reservoir computing with compliant fiber networks" in *Smart Materials, Adaptive Structures and Intelligent Systems* (American Society of Mechanical Engineers 2025) vol. 89275, p. V001108A002.
66. N. Naughton *et al.*, Elastica: A compliant mechanics environment for soft robotic control. *IEEE Rob. Autom. Lett.* **6**, 3389–3396 (2021).
67. H. S. Chang *et al.*, "Energy shaping control of a cybertoctopus soft arm" in *2020 59th IEEE Conference on Decision and Control (CDC)* (IEEE, 2020), pp. 3913–3920.
68. H. S. Chang *et al.*, Energy-shaping control of a muscular octopus arm moving in three dimensions. *Proc. R. Soc. A* **479**, 20220593 (2023).
69. T. Wang *et al.*, "A sensory feedback control law for octopus arm movements" in *2022 IEEE 61st Conference on Decision and Control (CDC)* (IEEE, 2022), pp. 1059–1066.

70. C. H. Shih *et al.*, Hierarchical control and learning of a foraging cyberoctopus. *Adv. Intell. Syst.* **5**, 2300088 (2023).
71. N. Charles, R. Chelakkot, M. Gazzola, B. Young, L. Mahadevan, Topological dynamics of rapid non-planar gaits in slithering snakes. *Nat. Phys.* **21**, 856–860 (2025).
72. B. Calvo *et al.*, Passive nonlinear elastic behaviour of skeletal muscle: Experimental results and model formulation. *J. Biomech.* **43**, 318–325 (2010).
73. C. N. Maganaris, J. P. Paul, In vivo human tendon mechanical properties. *J. Physiol.* **521 Pt 1**, 307–313 (1999).
74. C. L. Evans, A. V. Hill, The relation of length to tension development and heat production on contraction in muscle. *J. Physiol.* **49**, 10–16 (1914).
75. R. F. Ker, Mechanics of tendon, from an engineering perspective. *Int. J. Fatigue* **29**, 1001–1009 (2007).
76. M. Vatanikhah-Varnosfaderani *et al.*, Mimicking biological stress-strain behaviour with synthetic elastomers. *Nature* **549**, 497–501 (2017).
77. Z. Tu *et al.*, Biomimetic high performance artificial muscle built on sacrificial coordination network and mechanical training process. *Nat. Commun.* **12**, 2916 (2021).
78. W. L. Lim, L. L. Liao, M. H. Ng, S. R. Chowdhury, J. X. Law, Current progress in tendon and ligament tissue engineering. *Tissue Eng. Regen. Med.* **16**, 549–571 (2019).
79. S. Aziz *et al.*, Plant-like tropisms in artificial muscles. *Adv. Mater.* **35**, e2212046 (2023).
80. P. Matthews, Muscle spindles and their motor control. *Physiol. Rev.* **44**, 219–288 (1964).
81. M. T. Hagan, H. B. Demuth, "Neural networks for control" in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)* (IEEE, 1999), vol. 3, pp. 1642–1656.
82. A. Perrusquia, W. Yu, Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview. *Neurocomputing* **438**, 145–154 (2021).
83. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. *arXiv [Preprint]* (2017). <https://doi.org/10.48550/arXiv.1707.06347> (Accessed 7 April 2026).
84. Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **31**, 1235–1270 (2019).
85. J. Zou, Y. Han, S. S. So, "Overview of Artificial Neural Networks" in *Artificial Neural Networks: Methods and Applications*, D. J. Livingstone, Ed. (Humana Press, Totowa, NJ, 2009), pp. 14–22.
86. J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**, 024102 (2018).
87. K. Nagdi, *Rubber as an Engineering Material: Guideline for Users* (Hanser Verlag, 1993).
88. D. Fish, J. Orenstein, S. Bloom, Passive stiffness of isolated cardiac and skeletal myocytes in the hamster. *Circ. Res.* **54**, 267–276 (1984).
89. A. Di Clemente, F. Maiole, I. Borgia, L. Zullo, Beyond muscles: Role of intramuscular connective tissue elasticity and passive stiffness in octopus arm muscle function. *J. Exp. Biol.* **224**, jeb242644 (2021).
90. B. Chen, R. Kwiatkowski, C. Vondrick, H. Lipson, Fully body visual self-modeling of robot morphologies. *Sci. Rob.* **7**, eabn1944 (2022).
91. Y. Hu, J. Lin, H. Lipson, Teaching robots to build simulations of themselves. *Nat. Mach. Intell.* **7**, 484–494 (2025).
92. M. Zhu, O. G. Schmidt, Batteries for small-scale robotics. *MRS Bull.* **49**, 115–124 (2024).
93. Y. Sandamirskaya, M. Kiboli, J. Conrad, T. Celikel, Neuromorphic computing hardware and neural architectures for robotics. *Sci. Rob.* **7**, eabl8419 (2022).
94. P. F. Dominey, Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biol. Cybern.* **73**, 265–274 (1995).
95. B. Rajendran, A. Sebastian, M. Schmucker, N. Srinivasa, E. Eleftheriou, Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches. *IEEE Signal Process. Mag.* **36**, 97–110 (2019).
96. A. Javanshir, T. T. Nguyen, M. P. Mahmud, A. Z. Kouzani, Advancements in algorithms and neuromorphic hardware for spiking neural networks. *Neural Comput.* **34**, 1289–1328 (2022).
97. H. Herr, R. G. Dennis, A swimming robot actuated by living muscle tissue. *J. Neuroeng. Rehab.* **1**, 1–9 (2004).
98. J. Van Leeuwen, Optimum power output and structural design of sarcomeres. *J. Theor. Biol.* **149**, 229–256 (1991).
99. J. Van Leeuwen, W. M. Kier, Functional design of tentacles in squid: Linking sarcomere ultrastructure to gross morphological dynamics. *Philos. Trans. R. Soc. London. Ser. B: Biol. Sci.* **352**, 551–571 (1997).
100. R. L. Lieber, C. G. Brown, C. L. Trestik, Model of muscle-tendon interaction during frog semitendinosus fixed-end contractions. *J. Biomech.* **25**, 421–428 (1992).
101. L. F. Abbott, Lapique's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* **50**, 303–304 (1999).
102. N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **11**, 1–18 (2003).
103. T. Bekolay *et al.*, Nengo: A Python tool for building large-scale functional brain models. *Front. Neuroinf.* **7**, 1–13 (2014).
104. A. Raffin *et al.*, Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.* **22**, 1–8 (2021).
105. J. K. Eshraghian *et al.*, Training spiking neural networks using lessons from deep learning. *Proc. IEEE* **111**, 1016–1054 (2023).
106. N. Naughton, A. Tekinalp, K. Shivam, S. H. Kim, M. Gazzola, GazzolaLab/SoftArmRC. GitHub. <https://github.com/GazzolaLab/SoftArmRC>. Deposited 8 April 2026.