2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Neural Operators for Design-Space Surrogate Modeling of Tendon-Actuated Continuum Robots

Branden Frieden[1], James M. Ferguson[1], Alan Kuntz[2], and Varun Shankar[1]

***Abstract*— Continuum robots enable dexterous manipulation in constrained environments, but require accurate and efficient models for real-time manipulation and control. Traditional physics-based models can be computationally expensive and may suffer from inaccuracies due to unmodeled effects, while current learning-based methods often generalize poorly beyond the specific robot on which they are trained. We present a formulation of surrogate modeling for tendon-driven continuum robots as an operator learning problem that maps robot design parameters and tendon actuation inputs to resulting configurations. This formulation enables a single trained model to generalize across a large class of robot designs. We develop four novel neural operator architectures–two based on Deep Operator Networks (DeepONets) and two based on Fourier Neural Operators (FNOs)–and train them on simulation data to predict robot configurations. All architectures achieve good accuracy while allowing for fast and accurate generalization across designs. Our results demonstrate that operator learning provides an effective and generalizable surrogate for continuum robot mechanics in the design space, enabling fast modeling for control, planning, and design optimization in surgical and industrial applications.**

I. Introduction

Continuum robots (CRs) are manipulators with highly deformable backbones, capable of smooth, continuous motion rather than discrete jointed articulation. Their inherent compliance makes them well-suited for applications in minimally invasive surgery, inspection, and confined-space manipulation [34], [13], [7], [5]. Surveys highlight the breadth of CR designs and control approaches, emphasizing tendon-driven continuum robots (TDCRs) as a widely used actuation paradigm [38], [29], [7], [5]. Tendon actuation enables compact, dexterous designs, but introduces nonlinearities from friction, coupling, and backbone elasticity [26], [27].

Physics-based mechanics models of CRs typically use Cosserat rod theory, which treats the backbone as an geometric three dimensional structure rod [2], [27]. This framework captures bending, torsion, shear, and extension, and forms the foundation of many practical applications. However, numerical integration of these models, especially when coupled with tendon routing and external loads, can be computationally demanding [26], [33] and may not be as expressive as learned approaches which can capture unmodeled effects (e.g. friction).

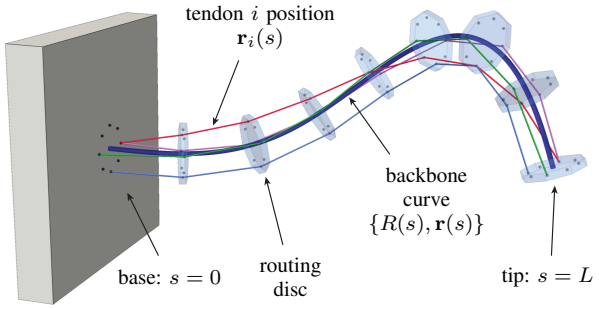

Fig. 1. Tendon-driven continuum robot (TDCR) that we model with neural Operators. Tendons are routed through holes in the attached discs at a constant pitch θ, and the flexible backbone bends continuously. Given tendon tensions, and robot design parameters, our model predicts the shape of the robot at any position along the backbone.

A promising recent paradigm for overcoming these challenges is in learned or surrogate models of CRs [18]. Prior work in soft and continuum robotics has emphasized the importance of efficient models for real-time control [8], [32], [30]. These modern methods allow models to *learn* kinematic or static mappings directly from data, either generated in simulation via physics-based models or via sensing of the physical robot as it is actuated. Examples include surveys of machine learning–based control [35], data-driven methods to compute the inverse kinematics [36], [20] and forward kinematics [3], [11], [15], [9], [14] of continuum robots, deep visual inverse kinematics for static shape control [1], model-less visual control of TDCRs using recurrent neurodynamic optimization [12], tip-force estimation using recurrent neural networks [10], and deep decoder approaches to hysteresis compensation [6]. While these approaches are typically efficient and accurate, they are generally *design specific*. That is, if a robot's tendon routing, disk spacing, or material properties change, new training data must be generated, either in simulation or by fabricating and actuating the physical robot.

In parallel, the scientific machine learning community has developed *neural operators*, architectures that learn mappings between infinite-dimensional function spaces rather than finite-dimensional vectors [17]. Deep Operator Networks (DeepONets) [22] and Fourier Neural Operators (FNOs) [19] are leading frameworks, with demonstrated success in modeling complex physical systems including fluid dynamics, materials modeling, and climate forecast-

[1]The Robotics Center and the Kahlert School of Computing at the University of Utah, Salt Lake City, UT 84112, USA.
[2]The Departments of Computer Science and Electrical and Computer Engineering at Vanderbilt University, Nashville, TN, 37203, USA.

VS was supported by NSF DMS 2505986. Research reported in this publication was also supported by the Advanced Research Projects Agency for Health (ARPA-H) under Award Number D24AC00415-00 for ALISS. The ARPA-H award provided 25% of total costs with an award total of up to $10,741,534.20. The content is solely the responsibility of the authors and does not necessarily represent the official views of ARPA-H.

ing [19], [37], [25]. Transformer-based neural operators have also begun to appear, for example ViTO, which applies vision transformers to PDE surrogates [24]. Neural operators combine superior generalization capabilities with fast inference, making them natural candidates for surrogate modeling in a wide variety of applications. Operator learning is also beginning to impact robotics. Recent examples include a neural-operator predictor for delay-compensating control with closed-loop stability guarantees and experiments on a 5-link manipulator [4]; an operator formulation of motion planning that learns a cost-to-value map via the Planning Neural Operator (PNO), enabling zero-shot super-resolution and generalization across environments [23]; and neural-operator-based haptic sensing along slender rods, relevant to interventional and continuum-robot settings [31].

Inspired by these developments, we develop *design-agnostic* surrogate models for TDCRs using two foundational neural operators: DeepONets and FNOs. To facilitate the use of neural operators as surrogates, we develop a mathematical formulation of TDCR surrogate learning as an operator learning task. Using data generated from Cosserat rod simulations of multiple designs with tendon actuation, we design and train custom DeepONets and FNOs to approximate the mapping from the design space of a tendon-actuated continuum robot to its equilibrium configuration space. This results in neural operator architectures that are able to output both positions in $\mathbb{R}^3$ and rotations in $SO(3)$. Our results show that these architectures achieve high accuracy on inference for unseen designs and fast evaluation times, demonstrating the potential of operator learning for design-agnostic modeling of continuum robots.

The remainder of this paper is organized as follows. In Section II, we present a review of the Cosserat rod model and of operator learning. Next, in Section III, we present a mathematical formulation of surrogate modeling of TDCRs as an operator learning problem, and present four new neural operator architectures (two based on DeepONets and two based on FNOs) suited to this task. We also discuss loss functions, important architectural details (for reproduction), and training details (including data generation). Then, in Section IV, we present convergence results for our model on the task of learning equilibrium configurations, explore the parsimony (or lack thereof) of our models, the behavior of these new models on out-of-distribution predictions, and also present both training and evaluation timings. We conclude in Section V with a summary and a description of future work.

II. Background

A. Continuum Robots as Cosserat Rods

We briefly review the physics-based mechanics model commonly used for tendon-actuated continuum robots, in which the backbone is modeled as a geometrically exact Cosserat rod [2], [28]. The shape of the rod is parameterized by the arc-length coordinate $s \in [0, L]$. Specifically, the centerline of the backbone is represented by a space curve $\mathbf{r}(s) \in \mathbb{R}^3$ together with an attached orthonormal frame $R(s) \in SO(3)$ describing the orientation of the cross-section; see Fig. 1 for an illustration.

The kinematic variables evolve along the rod according to

$$\boldsymbol{\nu}(s) = R(s)^\top \mathbf{r}'(s), \qquad R'(s) = R(s)\widehat{\boldsymbol{\kappa}}(s), \tag{1}$$

where $\boldsymbol{\nu}(s)$ and $\boldsymbol{\kappa}(s)$ denote the shear/extension and bending/torsion strain vectors, respectively.

Static equilibrium of the rod is governed by the balance of internal force $\mathbf{n}(s)$ and moment $\mathbf{m}(s)$ [28]:

$$\mathbf{n}'(s) + \mathbf{f}(s) = 0, \tag{2}$$

$$\mathbf{m}'(s) + \mathbf{r}'(s) \times \mathbf{n}(s) + \boldsymbol{\ell}(s) = 0, \tag{3}$$

where $\mathbf{f}$ and $\boldsymbol{\ell}$ denote distributed forces and moments acting along the rod. The stresses and strains inside the rod are related by the material constitutive law:

$$\mathbf{n} = K_s(\boldsymbol{\nu} - \boldsymbol{\nu}_0), \qquad \mathbf{m} = K_b(\boldsymbol{\kappa} - \boldsymbol{\kappa}_0), \tag{4}$$

where K_s, K_b are stiffness matrices encoding sensitivity to shear/elongation and bending/torsion, which can be computed from material properties.

Tendons routed at fixed cross-sectional offsets $\boldsymbol{\rho}_i$ generate distributed loads along the rod, resulting the shape deformation. The path of the ith tendon is given by

$$\mathbf{r}_i(s) = \mathbf{r}(s) + R(s)\boldsymbol{\rho}_i, \tag{5}$$

with unit tangent $\mathbf{t}_i = \frac{\mathbf{r}_i'}{\|\mathbf{r}_i'\|}$. The tendon tension τ_i, causes distributed loads along the rod according to [28]:

$$\mathbf{f}_i(s) = -\frac{d}{ds}\big(\tau_i \mathbf{t}_i\big), \qquad \boldsymbol{\ell}_i(s) = R(s)\,\boldsymbol{\rho}_i \times \mathbf{f}_i(s), \tag{6}$$

which contribute to the toal loads in(2)–(3) [26], [28].

Together, these equations describes the static equilibrium of the coupled rod–tendon system under prescribed base conditions and tendon actuation. Solving the resulting boundary value problem yeilds the robot configuration. The resulting system of equations serves as the high-fidelity physics model underpinning many continuum-robot simulators and controllers [13], [26], [38], [29]. Consequently, in this work, we use this formulation as our ground truth representation of a tendon-actuated continuum robot.

B. Operator Learning

We now provide a brief mathematical background on operator learning. Let $\mathcal{U}\left(\Omega_u; \mathbb{R}^{d_u}\right)$ and $\mathcal{V}\left(\Omega_v; \mathbb{R}^{d_v}\right)$ be two separable Banach spaces of $\mathbb{R}^{d_u}$- and $\mathbb{R}^{d_v}$-valued functions, respectively, taking values in $\Omega_u \subset \mathbb{R}^{d_x}$ and $\Omega_v \subset \mathbb{R}^{d_y}$, respectively. Further, let $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ be a general (nonlinear) operator. The operator learning problem involves approximating $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ with a parametrized operator $\hat{\mathcal{G}} : \mathcal{U} \times \Theta \to \mathcal{V}$ from a finite number of function pairs $\{(u_i, v_i)\}$, $i = 1, \ldots, N$ where $u_i \in \mathcal{U}$ are typically called *input functions*, and $v_i \in \mathcal{V}$ are called *output functions*, *i.e.*, $v_i = \mathcal{G}(u_i)$. The parameters Θ are chosen to minimize $\|\mathcal{G} - \hat{\mathcal{G}}\|$ in some norm. In practice, the problem must be discretized. This requires samples of the input and output functions at a finite set of function sample locations $X \in \Omega_u$ and $Y \in \Omega_v$,

respectively. One then requires that $\|v_i(y) - \hat{\mathcal{G}}(u_i)(y)\|_2^2$ is minimized over (u_i, v_i), $i = 1, \ldots, N$, where u_i are sampled at $x \in X$ and v_i at $y \in Y$, for some chosen architecture for $\hat{\mathcal{G}}$. In the context of surrogate modeling for partial differential equations (PDEs), the spaces $\mathcal{U}$ and $\mathcal{V}$ often correspond to spaces initial and final conditions (in the case of time-dependent PDEs), all of which are functions, and the sets X and Y correspond to spatial sampling locations. DeepONets and FNOs correspond to specific architectural choices for $\hat{\mathcal{G}}$.

III. Neural Operators as Surrogate Models for TDCRs

A. Operator Learning Formulation

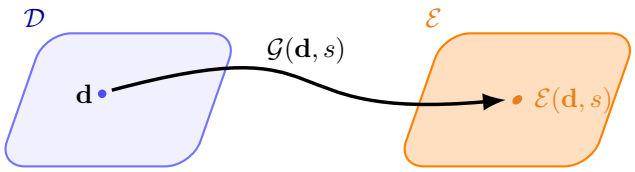

Fig. 2. Surrogate modeling of tendon-driven continuum robots as an operator learning problem. A design $\mathbf{d} \in \mathcal{D}$ (blue) is mapped by the operator $\mathcal{G}(\mathbf{d}, s)$ to its equilibrium $\mathcal{E}(\mathbf{d}, s) \in \mathcal{E}$ (orange).

We now formulate the Cosserat rod model as an *operator* that maps from a *design space* $\mathcal{D}$ to a corresponding *space of equilibrium configurations* $\mathcal{E}$ for a given fixed initial configuration of the rod. This differs from the standard setup of operator learning in Section II-B in one crucial way: while the elements of $\mathcal{E}$ are indeed *functions* like the elements of $\mathcal{V}$, the elements of $\mathcal{D}$ are instead finite collections of parameters and hence can be modeled as vectors (unlike the elements of $\mathcal{U}$, which are indeed functions). We now describe our formulation in detail; this formulation is visualized in Figure 2.

As mentioned previously, let $\mathcal{D}$ denote the *design space* of tendon–driven continuum robots (TDCRs). Each element $\mathbf{d} \in \mathcal{D}$ encodes a robot design with N_t tendons consisting of several parameters:

$$\mathbf{d} = \{\{\boldsymbol{\rho}_i\, \phi_i, \tau_i\}_{i=1}^{N_t}, r, L, E\}, \tag{7}$$

where $\boldsymbol{\rho}_i$ is once again the cross-sectional offset of each tendon, ϕ_i is the tendon path, τ_i is the tension on each tendon, L is the backbone length, r its radius, and E its Young's modulus. As implied above, we consider tendon tensions as the only actuation input for this work. For each $\mathbf{d}$ the Cosserat model defines an *equilibrium configuration*

$$\mathcal{E}(\mathbf{d}, s) := \{(\mathbf{r}_\mathbf{d}(s), R_\mathbf{d}(s))\},$$

where $\mathbf{r}_\mathbf{d} : [0, L] \to \mathbb{R}^3$ is the centerline and $R_\mathbf{d} : [0, L] \to SO(3)$ the frame field– both for a specific design $\mathbf{d}$ – satisfying the rod balance equations with tendon loads [2], [26]. This leads naturally to the definition of the infinite-dimensional equilibrium operator $\mathcal{G}$, which is a mapping

$$\mathcal{G} : \mathbf{d} \longmapsto (\mathbf{r}(\cdot), R(\cdot)), \qquad \mathcal{G} : \mathcal{D} \to \mathcal{X}, \tag{8}$$

with $\mathcal{X}$ the space of equilibrium configurations. Interestingly, the problem also directly admits a simpler alternative definition of the space $\mathcal{E}$ which we label $\mathcal{E}'$. It is possible to simply define the equilibrium configurations purely in terms of the tendon positions so that

$$\mathcal{E}'(\mathbf{d}, s) = \{\mathbf{r}_i(\mathbf{d}, s)\}_{i=1}^{N_t}. \tag{9}$$

Let $\mathcal{X}'$ be the space of all such equilibrium configurations now defined as the set of all $\mathcal{E}'$ for every $\mathbf{d}$. Then, naturally, we have the alternative definition of the equilibrium operator as $\mathcal{G}' : \mathcal{D} \to \mathcal{X}'$. We emphasize that $\mathcal{E}$ is a function that consumes a design $\mathbf{d}$ and an arclength s and outputs a vector in $\mathbb{R}^3$ and a 3×3 rotation matrix, while $\mathcal{E}'$ is a function that consumes the same inputs but instead outputs N_t vectors in $\mathbb{R}^3$. From the mathematical perspective, there is a strict one-to-one mapping between $\mathcal{E}$ and $\mathcal{E}'$ for each practical design $\mathbf{d}$, and consequently $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic.

We now discuss realizations of these operators through two neural operator architectures: DeepONets and FNOs. It is important to note that these operators could also be learned from actual data rather than Cosserat rod simulations.

B. Why Operator Learning

The Cosserat rod equations define a nonlinear boundary-value problem in arclength, and their solution induces a nonlinear solution operator. In the classical setting, this operator maps prescribed input functions and boundary conditions to rod configurations and internal force and moment fields. In our setting, we instead parameterize the inputs by the robot design variables together with the distributed loading, and seek the corresponding equilibrium state variables as output functions. This still defines a well-posed operator mapping from a finite-dimensional design/load space into an appropriate function space of rod states. Because repeated evaluation of this operator requires the numerical solution of a nonlinear system, it can become computationally expensive in design, control, and optimization workflows. This motivates the use of operator-learning methods to construct a data-driven surrogate of the solution operator, with the goal of substantially reducing evaluation cost while preserving the underlying input–output structure of the problem.

C. DeepONet Architectures

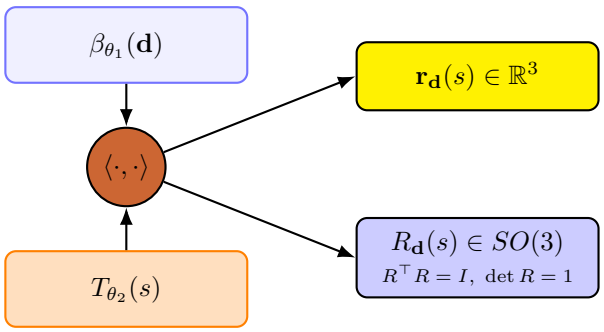

Fig. 3. DeepONet: branch $\beta_{\theta_1}(\mathbf{d})$ and trunk $T_{\theta_2}(s)$ fuse via an inner product $\langle \cdot, \cdot \rangle$ to produce $\mathcal{G}_{\text{DON}}(\mathbf{d}, s, \theta) = (\mathbf{r}_\mathbf{d}(s), R_\mathbf{d}(s))$ from (10). $\mathcal{G}'_{\text{DON}}$ instead outputs N_t tendon position vectors.

We present two DeepONet architectures, $\mathcal{G}_{\text{DON}}$ and $\mathcal{G}'_{\text{DON}}$, that learn $\mathcal{G}$ and $\mathcal{G}'$ respectively. These architectures use the same inputs, but differ in their outputs: $\mathcal{G}_{\text{DON}}$ outputs in $\mathcal{X}$ for each design $\mathbf{d}$ (a backbone vector in $\mathbb{R}^3$ and a 3×3 rotation matrix for each s), while $\mathcal{G}'_{\text{DON}}$ outputs in $\mathcal{X}'$ (N_t vectors in $\mathbb{R}^3$ for each s). $\mathcal{G}_{\text{DON}}$ is given explicitly by

$$\mathcal{G}_{\text{DON}}(\mathbf{d}, s, \theta) = \langle \beta_{\theta_1}(\mathbf{d}), T_{\theta_2}(s) \rangle, \tag{10}$$

where $\beta_{\theta_1}(\mathbf{d})$ is a (vector-valued) neural network with trainable parameters θ_1 (the "branch net") that encodes the design space $\mathcal{D}$; $T_{\theta_2}(s)$ is another (vector-valued) neural network with trainable parameters θ_2 that spans equilibrium configuration space $\mathcal{X}$; $\theta = \{\theta_1, \theta_2\}$; and $\langle ., . \rangle$ represents the ℓ_2 inner product. The branch and trunk are trained jointly to minimize some loss function; see Figure 3 for an illustration. $\mathcal{G}'_{\text{DON}}$ has the same high level architecture as well, which we write as

$$\mathcal{G}'_{\text{DON}}(\mathbf{d}, s, \theta') = \langle \beta'_{\theta'_1}(\mathbf{d}), T'_{\theta'_2}(s) \rangle. \tag{11}$$

Note the primes over the symbols to distinguish (11) from (10).

D. FNO Architectures

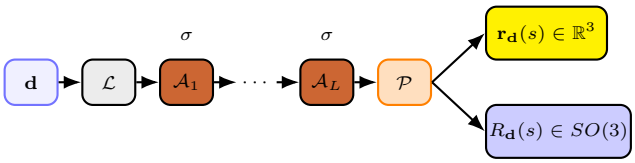

Fig. 4. FNO: the input design $\mathbf{d}$ is lifted and passed through activated Fourier layers (13) before it is projected using a DNN down to the equilibrium configuration, giving $\mathcal{G}_{\text{FNO}}$ in (12). $\mathcal{G}'_{\text{FNO}}$ instead outputs N_t tendon position vectors.

We also developed two FNO architectures, $\mathcal{G}_{\text{FNO}}$ and $\mathcal{G}'_{\text{FNO}}$, that learn $\mathcal{G}$ and $\mathcal{G}'$ respectively. FNO architectures differ significantly from DeepONets in that they are constructed by compositions of kernel-based integral operators parametrized by the fast Fourier transform (FFT), rather than from "vanilla" DNNs. More precisely, the FNO $\mathcal{G}_{\text{FNO}}$ can be written as

$$\mathcal{G}_{\text{FNO}}(\mathbf{d}) = \mathcal{P} \circ \mathcal{A}_L \circ \sigma \circ \ldots \circ \sigma \circ \mathcal{A}_1 \circ \mathcal{L} \circ \mathbf{d}, \tag{12}$$

where $\circ$ indicates composition; $\mathcal{L}$ is a linear layer that takes an $n \times |\mathbf{d}|$ matrix containing the tensor product of n arclength samples along the backbone with the design vector $\mathbf{d}$ and outputs an $n \times p$ matrix where p is the "channel dimension"; σ is a nonlinear (elementwise) activation; $\mathcal{P}$ is a standard DNN that projects (undoes $\mathcal{L}$); and $\mathcal{A}_k$ is a "Fourier layer", the functional equivalent of a standard DNN's affine transform but with an additional residual connection. To see the structure of $\mathcal{A}$, we first define the argument $\mathbf{q} = (\mathbf{d}, s)$, then write

$$\mathcal{A}(\mathbf{u}(\mathbf{q})) = \int_{D_{\mathbf{q}}} K(\mathbf{q}, \mathbf{q}')\mathbf{u}(\mathbf{q}')d\mathbf{q}' + W\underline{\mathbf{u}}, \tag{13}$$

where $\underline{\mathbf{u}}$ is the vector obtained by sampling the input function to the layer $(\mathbf{u}(\mathbf{q}))$ on the tensor-product of the design and arclength grids. The integral in (13) is handled via the FFT in that entries of the Gramian of K are learned directly in frequency space; we retain the 5 lowest frequency Fourier modes (we experimented with others, but found this performed the best). As in the DeepONet case, we also created a variant $\mathcal{G}'_{\text{FNO}}$ to directly output tendon equilibrium positions (i.e., to learn $\mathcal{G}'$); again as in the DeepONet case, $\mathcal{G}_{\text{FNO}}$ is trained to output a backbone position and two columns of the rotation matrix for each s.

E. Loss functions

We now discuss the loss functions used to train our neural operators. For all our models, we generate the same training dataset of N training pairs $\{\mathbf{d}_j, \mathcal{E}'(\mathbf{d}_j)\}_{j=1}^N$ (design– tendon equilibrium position) using a Cosserat rod simulator, where $\mathcal{E}'$ is given by (9). However, since the outputs of both types of models ($\mathcal{G}$ and $\mathcal{G}'$) are different, we use different loss functions for each. Both the DeepONet and FNO variants $\mathcal{G}_*$ that approximate $\mathcal{G}$ output a backbone position and rotation matrix, so the loss function is defined as

$$e(\mathbf{d}, s) = \sum_{i=1}^{N_t} \|\mathbf{r}_i(\mathbf{d}, s) - \left(\tilde{\mathbf{r}}_{\mathbf{d}}(s) + \tilde{R}_{\mathbf{d}}(s)\boldsymbol{\rho}_i\right)\|_2^2, \tag{14}$$

where $\tilde{\mathbf{r}}_{\mathbf{d}}(s)$ is the backbone position *estimated* by the neural operator $\mathcal{G}_*$ and $\tilde{R}_{\mathbf{d}}(s)$ is the estimated rotation matrix. As mentioned previously, we ensure the columns of this 3×3 rotation matrix are orthonormal by using the Gram-Schmidt process to orthonormalize the two columns output by $\mathcal{G}_*$ and to generate a third column as well; we do this during both training and inference. On the other hand, to train neural operators $\mathcal{G}'_*$, we instead define the loss

$$e'(\mathbf{d}, s) = \sum_{i=1}^{N_t} \|\mathbf{r}_i(\mathbf{d}, s) - \tilde{\mathbf{r}}_i(\mathbf{d}, s)\|_2^2, \tag{15}$$

where $\tilde{\mathbf{r}}_i(\mathbf{d}, s)$ are now the tendon equilibrium positons directly predicted by the neural operators $\mathcal{G}'_*$. Both e and e' are averaged over all N training designs $\mathbf{d}$ and evaluated on an n point grid for the variable $s \in [0, L]$ to obtain fully discrete losses.

F. More Architectural Details

We next describe the practical implementation of the DeepONet and FNO architectures evaluated in this work.

a) DeepONet architectures: For all our DeepONets, we use the "stacked" vanilla-DeepONet architecture (single branch rather than many) with multilayer perceptrons (MLPs) as both the branch β and trunk T. The branch network has five layers with hidden dimension 64, using $\tanh$ activations after each layer except the output (which is unactivated, as is customary). The trunk network again has five layers and a hidden dimension of 128, once again with $\tanh$ activations. For the DeepONet $\mathcal{G}'_{\text{DON}}$, the output dimension is $12p$ for vanilla DeepONet (corresponding to three spatial coordinates for each of the four tendons), while the output dimension for the DeepONet $\mathcal{G}_{\text{DON}}$ is $9p$ (three coordinates for the centerline offset and six values used to

construct the rotation matrix via the Gram–Schmidt method). In our experiments, we found that $p = 100$ performed the best in both architectures. This results in the vanilla DeepONet $\mathcal{G}'_{\mathrm{DON}}$ having 219,904 trainable parameters; $\mathcal{G}_{\mathrm{DON}}$ has 205,668 parameters.

b) FNO architectures: For FNOs, evaluation points are taken along the arclength of the centerline as an equispaced grid on $[0, L]$, concatenated with the model inputs. This two-dimensional input array is lifted to the channel dimension $p = 128$ and passed through five Fourier layers, each retaining five modes with ReLU activations in between. A final linear layer maps the output to the target dimension of 12 for $\mathcal{G}'_{\mathrm{FNO}}$ (three coordinates for each of four tendons) and 9 for $\mathcal{G}_{\mathrm{FNO}}$ (three coordinates for the centerline offset and six values for the rotation matrix via Gram–Schmidt). These architectural choices results in $\mathcal{G}'_{\mathrm{FNO}}$ having 168,844 trainable parameters and $\mathcal{G}_{\mathrm{FNO}}$ having 168,457 parameters.

G. Training

1) Data generation and preprocessing: We briefly describe our data generation and preprocessing. Input parameters were uniformly sampled over predefined ranges; these are shown in Table I) to obtain N designs. For each design $\mathbf{d}$, a Runge–Kutta solver was used to solve for $N_t = 4$ tendon positions in $\mathbb{R}^3$. Each of the N equilibrium configurations thus consists of 12 values (three spatial coordinates for four tendons) recorded at $n = 42$ equispaced points along the robot centerline/backbone. The corresponding arclength values were also stored and used as inputs to the neural operators.

This dataset was then partitioned into training and test sets by holding out 20% of the samples for testing, with the remaining 80% used for training. For our largest dataset, we generate 100,000 training examples, using $N = 80,000$ for training and 20,000 for testing. Models are trained using the full 80,000 examples unless stated otherwise, but the data is put into batches of size 5000 to allow the data to fit into GPU memory. As seen in Table I our parameters have several

TABLE I

RANGES OF INPUT PARAMETERS USED TO GENERATE TRAINING AND TEST DATA.

Parameter	Symbol	Range
Tendon tensions	τ_i	$[0.0, 5]$ N
Length of robot	L	$[0.1, 0.35]$ m
Tendon pitches	ϕ_i	$[-20, 20]$ rads/m
Tendon offset	ρ_i	$[0.005, 0.01]$ m
Young's modulus	E	$[15.5, 45.5] \times 10^9$ Pa
Backbone radius	r	$[.0005, .0015]$

magnitudes of difference between them. To mitigate the risk of a single parameter dominating the others, we regularized our inputs by dividing or multiplying certain parameters to ensure that they were within at most one order of magnitude from the others (e.g. dividing the Young's modulus by 10^9 to reduce its range to [15.5, 46.5]). This allowed us to keep our values from drowning out our smaller values while keeping the feature space of the parameters linear.

2) Training procedure: Our training procedure uses (14) and (15) sampled at $n = 42$ arclength locations and N training examples as our discrete loss. We update all model parameters using the Adam optimizer [16]. The learning rate is scheduled using the cyclical cosine annealing scheme [21]. Our schedule uses an initial learning rate of 10^{-4}, a warm up fraction of 0.3, a peak value of 3×10^{-3}, and end value of 5×10^{-6}, with 4 cycles and a γ of 0.7. We initialize this schedule assuming we will train for 100,000 epochs. However, because we stop training once the ℓ_2 training error converges, only the first part of the schedule is typically seen. These convergences occurred after around 20000 epochs for the DeepONets and around 1500 epochs for the FNOs.

IV. RESULTS

A. Preliminaries

For our results section, we represent the effectiveness of our models by measuring their relative l_2 error on generalization, e_{ℓ_2}, defined by:

$$e_{\ell_2} = \frac{\|y - \hat{y}\|_2}{\|y\|_2}, \tag{16}$$

where y is the ground truth vector, and $\hat{y}$ is the model's prediction of y. The accuracy of the model can be computed straightforwardly as $(1 - e_{\ell_2}) * 100\%$ to give us a percent accuracy of our model. For each result in this section. We train the models over 3 different seeds and report the average of the relative errors for our results. For the remainder of this section, we will refer to each of our models with the following labels:

- $\mathcal{G}_{\mathrm{DON}}$ as "DeepONetPose"; $\mathcal{G}'_{\mathrm{DON}}$ as "DeepONet".
- $\mathcal{G}_{\mathrm{FNO}}$ as "FNOPose"; $\mathcal{G}'_{\mathrm{DON}}$ as "FNO".

The rationale for this naming scheme is that the "Pose" variants explicitly output a "pose": a backbone and a local orthonormal basis to describe the offset to the tendons. All experiments were run on an Nvidia RTX 4090 GPU with code written using the JAX python library.

B. Generalization Errors

First, we explored the generalization error as the number of design-equilibrium pairs $\{\mathbf{d}_i, \mathcal{E}(\mathbf{d}_i\}_{i=1}^N$ is increased by training each model to (training) convergence. This type of convergence study (error as a function of N) is common in the neural operator literature. The idea is that the limit $N \to \infty$ corresponds to recovering the infinite-dimensional operator; in this setting, an absence of error decay would be concerning. The results are shown in Figure 5.

Figure 5 shows that the accuracy of the model improves rapidly as N increases, but the rate of decrease of the error begins to slow down around $N = 20,000$ training pairs with the minimum error achieved for different models at different values of N. This is likely due to the fact that the model capacity has been reached; further decrease in error will likely require wider or deeper networks, or perhaps a higher value of p (for both DeepONets and FNOs), or possibly more advanced architectures (such as those based on transformers and attention, for instance). Another trend that can be clearly

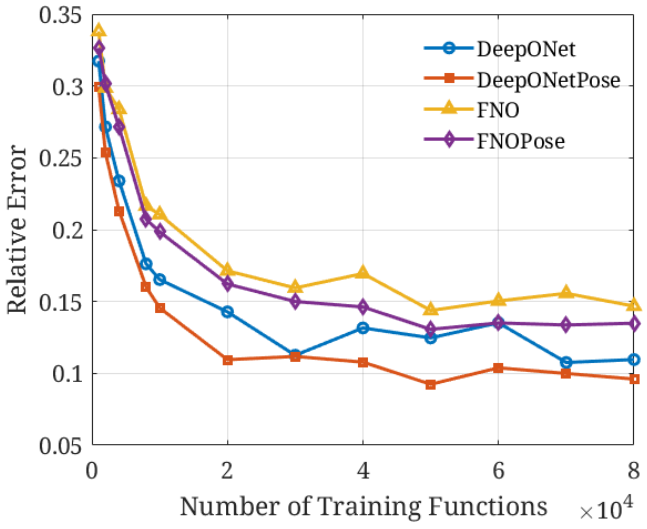

Fig. 5. Decay in generalization error as a function of number of training pairs N. We increase the number of training designs and the corresponding equilibrium configurations for all four models and compute errors according to (16).

seen is that DeepONets consistently achieve lower relative errors than FNOs. In addition, the "Pose" models (which output backbone positions and rotation matrices) outperform the baseline counterparts. The specific number of training functions required for a problem will in general depend on the ranges of the input design.

C. Model Parsimony

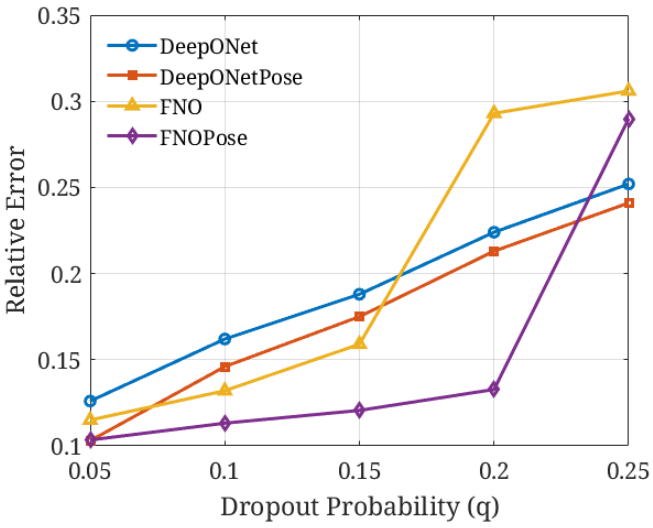

Fig. 6. The effect of dropout on generalization errors for $N = 80,000$. We see that dropout actually worsens the generalization errors across the board linearly, though it appears to aid the FNOPose model to a certain extent.

Neural operators typically have a large number of trainable parameters (as do our models). Naturally, it is reasonable to ask if such high parameter counts are necessary for learning the map from the design space to the equilibrium position space. To explore this question, we added dropout regularization to our training procedure by inserting dropout layers between the layers in our models. Since dropout is the procedure of zeroing out neuron weights randomly, we use this test as a method of discerning if our models are overparametrized or merely sufficiently parameterized. For the DeepONet, we inserted dropout layers between every layer in the branch and trunk MLPs; for the FNOs, we included dropout between the Fourier layers. We then varied the dropout probability q and trained our models.

The results are shown in Figure 6, which surprisingly shows that increasing q negatively impacts the relative error (in a linear fashion) even with small probabilities of zeroing out neurons. This overall behavior is indicative that our models appear to be fully utilizing all of their trainable parameters. Interestingly, there is a subtle difference between the DeepONet and FNO models: the DeepONet errors increases linearly with q, growing quicker than the FNO errors for small dropouts ($q \leq 0.15$); the latter grow sublinearly. For sufficiently large q, however, the FNO errors shoot up to be greater than the DeepONets. The FNO models seem to benefit slightly more from dropout; this is potentially due to dropout helping smooth away Gibbs' oscillations caused by the FFT (though this is merely speculation on our part). We observe that the "Pose" models and their counterparts seem to behave similarly, though the error in the FNOPose model does shoot up a slightly later q than its counterpart.

D. Out-of-Distribution Predictions

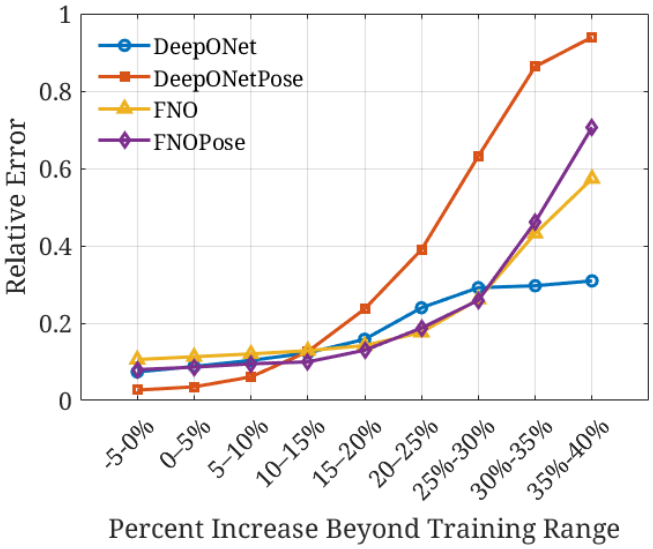

Fig. 7. Generalization errors for designs that are outside of the training distribution for models trained on $N = 80,000$ training pairs. The x-axis shows the percentage by which the design parameters are outside the training range.

We also investigated the ability of our models to extrapolate to out-of-distribution (OOD) inputs. Neural operators are widely regarded as capable of such extrapolation.

To generate data for this OOD extrapolation test, we chose design parameters of varying ranges beyond our training range. We first created a set of bins, each with an upper

and lower percent; specifically, we chose our bins to have a width of 5% and created four bins ranging the top end from 5%-20%. For each of these bins, we generated 1000 sets of random percentages, one random value for each input, within that bin's range. Then, we took the ranges from Table I, added the percentage to the right end of that range, and created parameters with those value. We then fed the designs composed of these new OOD inputs through the same Cosserat rod solver to obtain the corresponding equilibrium configurations. Finally, we fed these OOD pairs through each trained model and computed relative errors between the model outputs and the solver outputs. These errors are shown in Figure 7.

Figure 7 shows that our model accuracy falls off once we leave our training range, but with a strange caveat that small deviations actually improve the accuracy over the baseline! For instance, the 0-5% increase actually *improved* the accuracy of our model, in some cases to 97% (over the baseline 90%). A careful inspection of the model parameters reveals the true reason for this improvement: increasing the ranges of our parameters can in some cases make the TDCR backbones stiffer, thereby reducing their range of motion and making the equilibrium configuration easier to predict.

Our first data point for each model shows the in-distribution accuracy of our models (the -5% to 0% bin), once again showing that DeepONets are more accurate than FNOs, while the "Pose" models outperform their counterparts. The DeepONets errors appear to grow at a greater rate then the FNOs, with the DeepONetPose model's error increasing exponentially until it becomes the least accurate model. We similarly see the FNO and FNOPose model errors increasing exponentially, with the FNOPose overtaking the FNO model at around 20%, though not as quickly as the DeepONet-Pose model. Surprisingly, the vanilla-DeepONet begins to show this same behavior, but then flattens out and becomes the most accurate model at high percentages, with very reasonable errors. It seems that the "Pose" models (which output a backbone and a rotation matrix) extrapolate worse than those without. This makes some sense as deviations in the columns of the rotation matrices could be amplified through the Gram-Schmidt method, leading to less reliable predictions. In the future, we will explore forming a basis for $SO(3)$ and encoding that into the architectures directly.

In general, the OOD tests show that the actual physics governing the TDCR can dictate the generalization capabilities of the neural operator surrogates. The OOD performance can likely be improved by either incorporating Cosserat rod physics into the models as a soft loss term or designing neural operators that architecturally respect the Cosserat model. We will explore these avenues in future work.

E. Timings

Finally, we present timings for all four models on an RTX 4090 GPU. The training times are shown for $N = 1$, $N = 1000$, and $N = 80,000$ in Table II. While both DeepONets and FNOs take approximately the same time per training epoch, our DeepONets take O(20k) epochs to converge, while our FNOs require only O(2k) epochs for convergence. DeepONets were more accurate on generalization, but FNOs seem to possess a better accuracy-to-training-time tradeoff. It is certainly worth noting that our DeepONets required almost 2 hours to train on the largest N values while the FNOs only required 15 minutes.

TABLE II

MODEL TRAINING TIMES AS A FUNCTION OF N.

Model	$N = 1$	$N = 1000$	$N = 80,000$
DeepONet	149.4 s	150.6 s	108 min
DeepONetPose	144.2 s	148.3 s	106 min
FNO	13.66 s	14.66 s	15.5 min
FNOPose	14.56 s	15.28 s	15.8 min

TABLE III

MODEL INFERENCE TIMES.

Model	**1 design**	**1k designs**	**80k designs**
DeepONet	.0019 s	.0022 s	.0878 s
DeepONetPose	.0017 s	.0021 s	.0756 s
FNO	.0015 s	.0015 s	.1997 s
FNOPose	.0014 s	.0016 s	.2009 s

Of course, the modern perspective is that training is preprocessing, and that inference times are more important. In Table III, we show some inference times for predicting equilibrium configurations from designs using pretrained models. Here, we see that both the DeepONet and FNO models are capable of rapid inference, with DeepONets able to produce equilibria for $80,000$ designs in less than 1/10 of a second, and FNOs in about 2/10 of a second. Given that all models are able to achieve high accuracy (approximately 90%) on generalization to never-seen-before designs, these results provide strong evidence that our neural operator models are well-suited to accurate and rapid inference. An alternative perspective is that neural operators can be used as an efficient and accurate forward model for real-time tasks involving TDCRs.

V. CONCLUSION

In this work, we developed a mathematical framework for casting the problem of learning maps from TDCR designs to their equilibrium positions as an operator learning problem. We then developed four design-agnostic models based on neural operators that can generalize to new designs at runtime. We demonstrated the ability of these models to accurately and rapidly predict robot kinematics on new designs never seen before.

In this work, we demonstrated results for 4-tendon TDCRs. That said, our models are truly design-agnostic; they accept tendon paths, tensions, and offsets from the backbone. Adding more tendons will only require passing additional parameters (the number of tendons) and adding the extra paths, tensions, and offsets into the vector of parameters that determines a design. Further, in this work we designed surrogates based on DeepONets and FNOs. However, our mathematical framework is general and will lend itself well

to other neural operators.

This work takes significant steps toward a learned model that is accurate for the spectrum of designs rather than specific to one, enabling faster optimization, control, and planning for these robots.

REFERENCES

[1] E. ALMANZOR, C. LUCAS, J. BINGHAM, R. HARTMANN, AND M. YIP, *Static shape control of soft continuum robots using deep visual inverse kinematic models*, IEEE Transactions on Robotics, 39 (2023), pp. 4689–4706.

[2] S. S. ANTMAN, *Nonlinear Problems of Elasticity*, vol. 107 of Applied Mathematical Sciences, Springer, New York, 2 ed., 2005.

[3] C. BERGELES, F. Y. LIN, AND G. Z. YANG, *Concentric tube robot kinematics using neural networks*, in Hamlyn symposium on medical robotics, June 2015, pp. 1–2.

[4] L. BHAN, P. QIN, M. KRSTIC, AND Y. SHI, *Neural operators for predictor feedback control of nonlinear delay systems*, in Proceedings of the 7th Annual Learning for Dynamics & Control Conference, N. Ozay, L. Balzano, D. Panagou, and A. Abate, eds., vol. 283 of Proceedings of Machine Learning Research, PMLR, Jun 2025, pp. 179–193.

[5] J. BURGNER-KAHRS, D. C. RUCKER, AND H. CHOSET, *Continuum robots for medical applications: a survey*, IEEE Transactions on Robotics, 31 (2015), pp. 1261–1280.

[6] B. Y. CHO, D. S. ESSER, J. THOMPSON, B. THACH, R. J. WEBSTER, AND A. KUNTZ, *Accounting for Hysteresis in the Forward Kinematics of Nonlinearly-Routed Tendon-Driven Continuum Robots via a Learned Deep Decoder Network*, IEEE Robotics and Automation Letters, 9 (2024), pp. 9263–9270.

[7] P. DUPONT, N. SIMAAN, H. CHOSET, AND C. RUCKER, *Continuum robots for medical interventions*, Proceedings of the IEEE, 110 (2022), pp. 847–870.

[8] C. DURIEZ, *Control of elastic soft robots based on real-time finite element method*, in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2013, pp. 3982–3987.

[9] G. FAGOGENIS, C. BERGELES, AND P. E. DUPONT, *Adaptive non-parametric kinematic modeling of concentric tube robots*, in IEEE/RSJ international conference on intelligent robots and systems (IROS), Oct. 2016, pp. 4324–4329.

[10] F. FENG, W. HONG, AND L. XIE, *A learning-based tip contact force estimation method for tendon-driven continuum manipulator*, Scientific Reports, 11 (2021), p. 17482.

[11] R. GRASSMANN, V. MODES, AND J. BURGNER-KAHRS, *Learning the forward and inverse kinematics of a 6-DOF concentric tube continuum robot in SE(3)*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, Oct. 2018, pp. 5125–5132. tex.booktitle: 2018 IEEE Int. Conf. Intell. Robots Syst. (IROS).

[12] S. HE, C. ZOU, Z. DENG, W. LIU, B. HE, AND J. ZHANG, *Model-less optimal visual control of tendon-driven continuum robots using recurrent neural network-based neurodynamic optimization*, Robotics and Autonomous Systems, 182 (2024), p. 104811.

[13] R. J. W. III AND B. A. JONES, *Design and kinematic modeling of constant curvature continuum robots: A review*, The International Journal of Robotics Research, 29 (2010), pp. 1661–1683.

[14] M. KASAEI, K. K. BABARAHMATI, Z. LI, AND M. KHADEM, *A Data-efficient Neural ODE Framework for Optimal Control of Soft Manipulators*, in The Conference on Robot Learning, PMLR, 2023, pp. 1–14.

[15] ———, *Data-efficient non-parametric modelling and control of an extensible soft manipulator*, in IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 2641–2647.

[16] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, (2017).

[17] N. B. KOVACHKI, Z. LI, B. LIU, K. AZIZZADENESHELI, K. BHATTACHARYA, A. M. STUART, AND A. ANANDKUMAR, *Neural operator: Learning maps between function spaces*, Journal of Machine Learning Research, 24 (2023), pp. 1–97.

[18] A. KUNTZ, A. SETHI, R. J. WEBSTER, III, AND R. ALTEROVITZ, *Learning the Complete Shape of Concentric Tube Robots*, IEEE Transactions on Medical Robotics and Bionics, 2 (2020), pp. 140–147. Publisher: IEEE.

[19] Z. LI, N. KOVACHKI, K. AZIZZADENESHELI, B. LIU, K. BHATTACHARYA, A. M. STUART, AND A. ANANDKUMAR, *Fourier neural operator for parametric partial differential equations*, in Proc. Intl. Conf. on Learning Representations (ICLR), 2021. arXiv:2010.08895.

[20] N. LIANG, R. M. GRASSMANN, S. LILGE, AND J. BURGNER-KAHRS, *Learning-based Inverse Kinematics from Shape as Input for Concentric Tube Continuum Robots*, in 2021 IEEE International Conference on Robotics and Automation (ICRA), May 2021, pp. 1387–1393.

[21] I. LOSHCHILOV AND F. HUTTER, *Sgdr: Stochastic gradient descent with warm restarts*, arXiv preprint arXiv:1608.03983, (2017).

[22] L. LU, P. JIN, G. PANG, Z. ZHANG, AND G. E. KARNIADAKIS, *Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators*, Nature Machine Intelligence, 3 (2021), pp. 218–229.

[23] S. MATADA, L. BHAN, Y. SHI, AND N. ATANASOV, *Generalizable motion planning via operator learning*, in International Conference on Learning Representations (ICLR), 2025. Published as a conference paper at ICLR 2025.

[24] A. OVADIA, A. KAHANA, P. STINIS, E. TURKEL, AND G. E. KARNIADAKIS, *Vito: Vision transformer operator for pdes*, arXiv preprint, (2023).

[25] J. PATHAK, S. SUBRAMANIAN, P. HARRINGTON, S. RAJA, A. CHATTOPADHYAY, M. MARDANI, T. KURTH, D. HALL, Z. LI, K. AZIZZADENESHELI, P. HASSANZADEH, K. KASHINATH, AND A. ANANDKUMAR, *Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators*, arXiv preprint, (2022).

[26] P. RAO, Q. PEYRON, S. LILGE, AND J. BURGNER-KAHRS, *How to model tendon-driven continuum robots and benchmark modelling performance*, Frontiers in Robotics and AI, 7 (2021), p. 630245.

[27] D. C. RUCKER AND R. J. W. III, *Statics and dynamics of continuum robots with general tendon routing and external loading*, IEEE Transactions on Robotics, 27 (2011), pp. 1033–1044.

[28] D. C. RUCKER AND R. J. WEBSTER III, *Statics and Dynamics of Continuum Robots With General Tendon Routing and External Loading*, IEEE Transactions on Robotics, 27 (2011), pp. 1033–1044. Conference Name: IEEE Transactions on Robotics.

[29] M. RUSSO, M. GANDOLFI, M. PALPACELLI, S. LONGHI, R. MURADORE, V. ZIMMER, AND S. HADDADIN, *Continuum robots: An overview*, Advanced Intelligent Systems, 5 (2023), p. 2200367.

[30] C. D. SANTINA, C. DURIEZ, AND D. RUS, *Model-based control of soft robots: A survey of the state of the art and open challenges*, arXiv preprint arXiv:2110.01358, (2021). See also IEEE Control Systems Magazine, 43(3):30–65, 2023.

[31] X. TANG, T. YANG, J. WANG, G. HAN, Q. SUN, H. PENG, AND Z. SUN, *Neural operator-based haptic sensing along slender rods*, International Journal of Mechanical Sciences, 304 (2025), p. 110699. In Press, Journal Pre-proof.

[32] T. G. THURUTHEL, Y.-L. PARK, C. LASCHI, F. MASTROGIOVANNI, AND E. FALOTICO, *Control strategies for soft robotic manipulators: A survey*, Soft Robotics, 5 (2018), pp. 149–163.

[33] J. TILL, C. ALOI, AND D. C. RUCKER, *Real-time dynamics of soft and continuum robots based on cosserat rod models*, The International Journal of Robotics Research, 38 (2019), pp. 1575–1592.

[34] I. D. WALKER, *Continuous backbone "continuum" robot manipulators*, ISRN Robotics, 2013 (2013), p. 726506.

[35] X. WANG, H. CHEN, H. SADEGHIAN, G.-Z. YANG, P. VALDASTRI, N. SIMAAN, R. CARLONI, K. JIN, AND T. WANG, *A survey for machine learning-based control of continuum robots*, Frontiers in Robotics and AI, 8 (2021), p. 730330.

[36] W. XU, J. CHEN, H. Y. LAU, AND H. REN, *Data-driven methods towards learning the highly nonlinear inverse kinematics of tendon-driven surgical manipulators*, The International Journal of Medical Robotics and Computer Assisted Surgery, 13 (2017), p. e1774.

[37] H. YOU, Z. LI, K. AZIZZADENESHELI, N. B. KOVACHKI, A. ANANDKUMAR, Y. YU, AND G. E. KARNIADAKIS, *Learning physics-consistent neural operators: Towards robust deep learning for pdes*, Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114399.

[38] J. ZHANG, Q. FANG, P. XIANG, D. SUN, Y. XUE, R. JIN, K. QIU, R. XIONG, Y. WANG, AND H. LU, *A survey on design, actuation, modeling, and control of continuum robot*, Cyborg and Bionic Systems, 2022 (2022), p. 9754697.