

Learning Controllers for Continuum Soft Manipulators: Impact of Modeling and Looming Challenges

Egidio Falotico,* Enrico Donato, Carlo Alessi, Elisa Setti, Muhammad Sunny Nazeer, Camilla Agabiti, Daniele Caradonna, Diego Bianchi, Francesco Piqué, Yasmin Tauqeer Ansari, and Marc Killpack

Soft manipulators, renowned for their compliance and adaptability, hold great promise in their ability to engage safely and effectively with intricate environments and delicate objects. Nonetheless, controlling these soft systems presents distinctive hurdles owing to their nonlinear behavior and complicated dynamics. Learning-based controllers for continuum soft manipulators offer a viable alternative to model-based approaches that may struggle to account for uncertainties and variability in soft materials, limiting their effectiveness in real-world scenarios. Learning-based controllers can be trained through experience, exploiting various forward models that differ in physical assumptions, accuracy, and computational cost. In this article, the key features of popular forward models, including geometrical, pseudo-rigid, continuum mechanical, or learned, are first summarized. Then, a unique characterization of learning-based policies, emphasizing the impact of forward models on the control problem and how the state of the art evolves, is offered. This leads to the presented perspectives outlining current challenges and future research trends for machine-learning applications within soft robotics.

use of compliant materials and sometimes compliance in the actuators means that soft robots may have a larger dynamic workspace that can more easily make use of stored potential energy. The use of such robots in real-world environments would benefit several emerging robotic applications for interactions with human collaborators or interaction with unmodeled environments, such as medicine, search and rescue, disaster relief, and human assistance.^[2] In this article, we focus on continuum soft manipulators (CSMs), a subclass of soft robots tailored for manipulation tasks and characterized by continuous elastic deformations.^[3]

The large-scale deployment of soft robots is hampered by several challenges that still need to be addressed. While their compliance makes them suitable for unstructured environments and uses alongside people, it also makes them have

a theoretically infinite number of degrees of freedom (DoFs) in space with a limited number of actuators, hence they are considered under-actuated systems.^[4] This property, along with the fact that the deformation of the robot during motion cannot be neglected,^[5] makes traditional robot control methods not directly applicable to soft robots.

Soft robot control methods include both low-level and high-level formulations.^[6] The former is related to the

1. Introduction

Soft robots have the potential to fill the functionality gap left by rigid robots which are suitable for fast, precise, and repetitive tasks. Specifically, the use of compliant materials and the flexibility possible in design choices related to geometry or actuation of soft robots enable a more dexterous range of motions than what can be achieved by traditional robots.^[1] In addition, the


E. Falotico, E. Donato, C. Alessi, E. Setti, M. S. Nazeer, C. Agabiti, D. Caradonna, D. Bianchi
The BioRobotics Institute and the Department of Excellence in Robotics and AI
Sant'Anna School of Advanced Studies
56025 Pontedera, Italy
E-mail: egidio.falotico@santannapisa.it

M. S. Nazeer
Department of Mechanical Engineering
National University of Singapore
Singapore 127575, Singapore

F. Piqué
Department of Information Engineering
University of Pisa
56125 Pisa, Italy

Y. T. Ansari
Bioinspired Soft Robotics Lab
Istituto Italiano di Tecnologia
16163 Genova, Italy

M. Killpack
Department of Mechanical Engineering
Brigham Young University
Provo, UT 84602, USA

 The ORCID identification number(s) for the author(s) of this article can be found under <https://doi.org/10.1002/aisy.202400344>.

© 2024 The Author(s). Advanced Intelligent Systems published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202400344

actuation-dependent part and deals with problems resulting from the variability in the performance of the actuators used^[7] (e.g., low actuation frequency, limited bandwidth, high-frequency-induced vibrations). Conversely, high-level control methods for soft robots implement planning and decision-making for task fulfillment but must do so under significant uncertainty.

The literature on soft robotic control suggests two main methods to control soft robots for difficult tasks and handle the complexity of soft structures. These include using model-based^[8] and learning-based^[6] controllers. The former utilizes models to predict and react to the robots' behavior. The latter extracts knowledge of the system by collecting data from simulated or real-world environments. Despite the result being a robot- and task-specific method, the learning-based controller finds an implicit predictive model of the system producing a computationally efficient algorithm, and avoiding model identification (which is required for model-based control and which is a difficult and unsolved problem).

The focus of our work is on the emerging field of learning-based methods for high-level control of CSMs.^[9] Given the potentially infinite number of DoFs, training complex policies directly on soft robotic platforms may be infeasible. At least one reason includes the fact that the physiological degradation of soft materials would not sustain the large number of iterations needed to learn a policy. Therefore, the learning process is often conducted in simulation, leveraging forward models. In the last few decades, four main categories of forward models for CSMs were proposed:^[10] 1) geometrical models, 2) discrete models, 3) continuum mechanical models, and 4) machine-learning (ML) models. These approaches differ in the degree of accuracy, realism, and computational cost.

In this article, we introduce fundamental soft robotics concepts and overview the main features of the related modeling approaches (Section 2). Then, we characterize learning-based controllers in terms of the models employed, analyzing the impact of the model on the learning process. Furthermore, we trace the evolution of state-of-the-art controllers for each category (Section 3). Despite recent remarkable achievements, there remain significant unanswered questions before we can deploy soft robots in the real world. We discuss persistent challenges and research gaps in ML applications to soft robotics (Section 4) and conclude by summarizing the main observations (Section 5).

2. Modeling CSMs

The unique continuum nature of CSMs introduces novel motion capabilities to the robotics domain but, simultaneously, poses a considerable challenge in their modeling.^[10] Theoretically, a CSM exhibits an infinite number of DoFs, rendering the design of models and controllers highly complex.^[8] Furthermore, the elastic nature of the system gives rise to nonlinear phenomena such as hysteresis and degradation.^[11] To address these complexities, CSMs often require discretization for control and implementation purposes, thereby reducing the infinite number of variables to a manageable set. This process is commonly referred to in the literature as spatial discretization.^[12]

In this context, state-of-the-art models can be sorted into four main categories: ML-based, geometrical, discrete, and continuum mechanical models.^[10] These approaches differ by the spatial discretization technique used and/or the assumptions made about the CSM, and are described in more detail later.

2.1. General Concepts

Consider a CSM (Figure 2a), an elastic body with length L , parameterized by the curvilinear abscissa $s \in [0, L]$. We refer to the cross section at $s = 0$ as base and the one at $s = L$ as tip.

Regardless of the employed spatial discretization, it is always possible to define a vector of generalized coordinates $\mathbf{q} \in \mathbb{R}^n$ that fully describes the shape and the time evolution of the CSM. Typically, \mathbf{q} has geometrical information about the robot's shape (e.g., curvature and torsion), or instead it includes kinematic information (e.g., position or velocity). The subspace containing all the possible values of \mathbf{q} is called the configuration space $C \subseteq \mathbb{R}^n$. Using this definition, the forward kinematics (FK) and the differential forward kinematics (DFK) can be defined as

$$\mathbf{g}(s, t) = \text{FK}(\mathbf{q}, s) \quad \text{FK} \quad (1)$$

$$\boldsymbol{\eta}(s, t) = \mathbf{J}(\mathbf{q}, s) \dot{\mathbf{q}} \quad \text{DFK} \quad (2)$$

In Equation (1), $\mathbf{g}(s, t) \in SE(3)$ is the homogeneous transformation associated with the cross section s and $\text{FK}: C \times [0, L] \rightarrow SE(3)$ is the FK function. Whereas, in Equation (2), $\boldsymbol{\eta}(s, t) = [\boldsymbol{\omega}^T \quad \mathbf{v}^T]^T \in \mathbb{R}^6$ is the velocity twist associated with the cross section s , such that $\boldsymbol{\omega}$ and \mathbf{v} are the angular and linear velocities, and $\mathbf{J}(\mathbf{q}, s) \in \mathbb{R}^{6 \times n}$ is the soft geometric Jacobian. Using the FK equation (Equation (1)), the task space $T \subseteq \mathbb{R}^l$ can be defined, containing all the possible poses (i.e., positions and orientations) of the robot's tip $\mathbf{g}(L, t)$. Theoretically, the dimension of T is much less than the dimension of the configuration space C , highlighting the redundancy of a CSM.

Similarly to the rigid case, the forward dynamics (FD) can be written in a Lagrangian form,^[13] such as

$$\mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{K}(\mathbf{q}) \mathbf{q} + \mathbf{D}(\mathbf{q}) \dot{\mathbf{q}} = \mathbf{B}(\mathbf{q}) \mathbf{u} + \mathbf{F}_{\text{ext}} \quad (3)$$

where $\mathbf{u} \in \mathbb{R}^m$ is the vector of the actuators' magnitude, $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ is the Coriolis matrix, $\mathbf{G} \in \mathbb{R}^n$ is the distributed gravity vector, $\mathbf{K} \in \mathbb{R}^{n \times n}$ and $\mathbf{D} \in \mathbb{R}^{n \times n}$ are the stiffness and damping matrices, respectively, $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the actuation model, and $\mathbf{F}_{\text{ext}} \in \mathbb{R}^n$ is the distributed external forces vector. Unlike FD for standard rigid manipulators,^[13] the presence of the mechanical impedance (i.e., stiffness and damping) is not negligible and represents the compliance of the soft structure.

For many soft robots, the actuation model described by \mathbf{B} can be nontrivial to define. However, based on the actuation model, it is possible to define another subspace called the actuation space $A \subseteq \mathbb{R}^m$, which contains all the possible values of the actuators' magnitude. It is worth highlighting that the \mathbf{B} matrix shows the under-actuated nature of the system, where typically $m \ll n$. This matrix exhibits how the actuators drive the CSM and it is closely related to the controllability of the system.

As seen from Equation (1) and (3), CSMs are redundant, under-actuated, and often highly compliant. These characteristics may appear as undesirable behaviors since they complicate the design of a controller. However, redundancy, under-actuation, and compliance are the keys to exploiting the continuum nature of CSMs.

The redundancy allows the design of a controller that satisfies the task, while at the same time minimizing some useful cost function or respecting additional constraints. This fact is evident in the inverse kinematics (IK) problem. Due to the different dimensions of configuration and task space ($l \ll n$), a pose of the tip $\mathbf{g}(s=L)$ can be mapped by a great variety of configurations \mathbf{q} . Consequentially, it is possible to find an optimal \mathbf{q}^* that satisfies the IK and minimizes a meaningful cost function (e.g., distance from obstacles, and magnitude of the velocity twist at the tip).

Under-actuation couples the finite number of actuators with the infinite DoFs of the CSM exploiting the compliance of the structure and reducing the actuation energy. Compliance, whether in actuators or material properties of the soft robot, enables safer contact and interaction with the environment. Additionally, compliance may facilitate dynamic energy storage and release for tasks that are often impractical or unsafe for rigid robots.

Finally, the CSM can be sensorized with proprioceptive and exteroceptive sensors, providing measurements as $\mathbf{y} \in \mathbb{R}^p$. It is therefore possible to write an observation model, defined as

$$\mathbf{y} = \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{u}, \mathbf{s}) \quad (4)$$

where $\mathbf{h}(\cdot) \in \mathbb{R}^p$ is the observation function. We can then define the observation space $\mathcal{O} \subseteq \mathbb{R}^p$ as the subset containing all the possible measurements, provided by the CSM sensing framework. **Figure 1** shows the relationships between the subspaces.

2.2. ML-Based Models

ML is a popular data-driven approach for modeling CSMs.^[14] Unlike analytical methods, it does not require explicit geometrical and physical representations for mapping between actuation and configuration spaces and between configuration and task

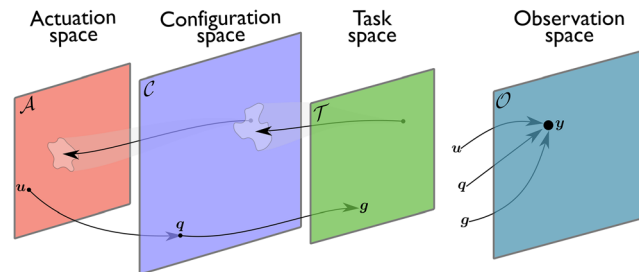


Figure 1. Subspaces. The four special subspaces related to a general model for CSMs. The actuation space \mathcal{A} (red) contains all the possible input values. The configuration space \mathcal{C} (blue) represents all the possible configurations of the CSM. Through the forward kinematics (FK), the task space \mathcal{T} (green) is composed of all the possible poses of the tip of the CSM. Finally, the observation space \mathcal{O} is the subset of all the measurements provided by the CSM's sensors.

spaces (**Figure 2b**). Instead, using function approximators like artificial neural networks (ANNs), we can directly learn a mapping between the actuation space \mathcal{A} and the task space \mathcal{T} as

$$\hat{\mathbf{g}}_{t+1} = f(\mathbf{u}_t, \dots, \mathbf{u}_{t-k}, \hat{\mathbf{g}}_t, \dots, \hat{\mathbf{g}}_{t-k}; \boldsymbol{\theta}) \quad (5)$$

Here, the function $f: \{\mathcal{A} \times \dots \times \mathcal{A}\} \times \{\mathcal{T} \dots \times \mathcal{T}\} \rightarrow \mathcal{T}$ is an ANN, which generally takes as input a sequence of actuations $\mathbf{u}_{t-k:t}$ and tip poses $\mathbf{g}_{t-k:t}$ and outputs a prediction of the next pose $\hat{\mathbf{g}}_{t+1}$. The horizon term k (where typically $k \leq 3$) allows the model to infer time-dependent information such as velocity and acceleration^[15,16] while $\boldsymbol{\theta}$ is the vector of the ANN weights.

Forward models of CSMs are derived by supervised learning (SL), which requires solving a multivariate regression task. First, motion data is collected on simulated or physical robots from a dataset of pseudo-random actuations \mathbf{u}_t and resulting tip poses \mathbf{g}_t . Usually, the data are split into training and validation sets. Then, a neural network architecture is defined, which includes choosing the input–output interface, the structure of the hidden layers, and neuron types. Subsequently, variants of stochastic gradient descent are used to train the network to find the optimal weights $\boldsymbol{\theta}^*$ that minimize a loss function as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}((\mathbf{g}_t)_{t \in \mathbb{R}^+}, (\hat{\mathbf{g}}_t)_{t \in \mathbb{R}^+}; \boldsymbol{\theta}) \quad (6)$$

where $(\mathbf{g}_t)_{t \in \mathbb{R}^+}$ and $(\hat{\mathbf{g}}_t)_{t \in \mathbb{R}^+}$ are the time series of the measured motion and predicted motion, respectively. Once the training is complete, the ANN serves as an emulator of the robot behavior for unseen actuations.^[15,16] However, we should note that forward models based on ML allows us to be flexible in the desired mappings. For instance, learning a map from actuation to configuration space,^[17] $\mathbf{u} \mapsto \hat{\mathbf{q}}$, can facilitate the achievement of secondary tasks like reaching a specific configuration \mathbf{q} while targeting a pose $\mathbf{g}(L)$ at the tip. Alternatively, other approaches learn the forward model of the task.^[18]

In summary, ML provides computationally efficient forward models with a simple formulation that only requires the ANN interface (i.e., input–output spaces and mapping) and capacity (i.e., number of parameters). In most cases, a small model is sufficient to represent complex motions. Since ML models rely on motion data, they could encode the entire robot mechanics expressed in Equation (3), including the effects of distributed gravity forces $\mathbf{G}(\mathbf{q})$ and external forces \mathbf{F}_{ext} . Moreover, they can represent complex physical phenomena (e.g., friction and hysteresis) that could be difficult to model analytically. However, the challenge of collecting labeled interaction data usually limits the actual representation power of ML models. Moreover, overfitting could prevent the deployment of ML models in scenarios that differ significantly from the training data. Concerning the training, the weights $\boldsymbol{\theta}$ can be learned within hundreds of epochs, with a variable computational cost depending on the network architecture and the dataset size. Finally, the black-box nature of ANNs yields forward models that lack interpretability, unlike analytical models, where each parameter has an explicit meaning.

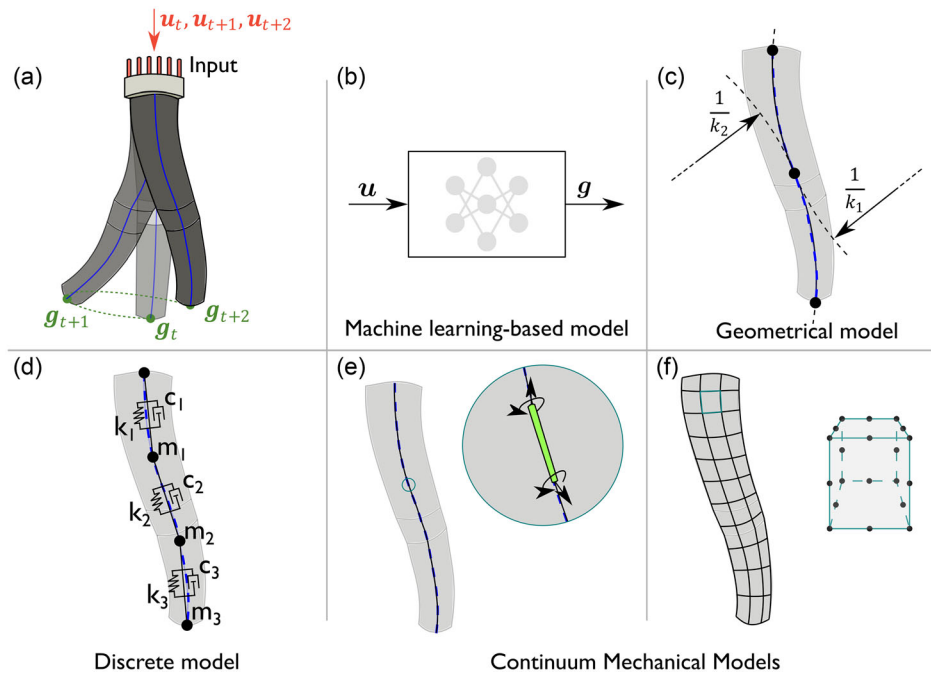


Figure 2. Modeling of soft robots. a) A generic CSM. The actuation inputs u_t, u_{t+1}, u_{t+2} can represent different quantities depending on the application, from tendon positions or torques to the pressure of pneumatic chambers going along the CSM. Different approaches to model a soft robot, respectively: b) ML-based model, the behavior of the CSM is learned by ML algorithms characterized by their black-box nature; c) geometrical model, the CSM shape is described by a specific mathematical curve; d) discrete model, the behavior of a CSM is approximated by a set of lumped masses (m), springs (k), and dampers (c); e) in the framework of continuum models, the CSM is approximated with a rod. Based on the rod theory, the model can include different strain modes: f) the structure of a CSM is discretized by a mesh at which nodes the set of partial differential equations (PDEs) is solved.

2.3. Geometrical Models

The concept behind the geometrical approach to modeling soft robots is to describe the robot's shape with a specific mathematical curve. This idea makes it possible to characterize the CSM kinematics with a finite number of configuration variables which describe the curve. Consequently, the vector q assumes a clear geometrical meaning (e.g., curvature, torsion, arc length). In the literature, many mathematical curves have been proposed, each with its own specific application and parameterization of configuration variables. The simplest and most widely used approach is piecewise constant curvature (PCC),^[19] shown in Figure 2c. The CSM's backbone is segmented into N pieces and each segment is treated as a single-oriented circumference arc. The main assumption of this model is that the torsion is zero; hence, it applies to CSMs in which the twisting deformation is negligible. In particular, it implies that the configuration space of a 3D CSM can be parameterized by three variables: the curvature κ , the arc length s , and the angle of the bending plane ϕ . The FK of a single soft segment (Equation (1)) can be represented as

$$g(q) = \begin{bmatrix} c_\phi c_{\kappa s} & -s_\phi & c_\phi s_{\kappa s} & \frac{1}{\kappa} c_\phi (1 - c_{\kappa s}) \\ s_\phi c_{\kappa s} & c_\phi & s_\phi s_{\kappa s} & \frac{1}{\kappa} s_\phi (1 - c_{\kappa s}) \\ -s_{\kappa s} & 0 & c_{\kappa s} & \frac{1}{\kappa} s_{\kappa s} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

where $s(\cdot) = \sin(\cdot)$ and $c(\cdot) = \cos(\cdot)$. Equation (7) can be seen as the FK of an equivalently rigid manipulator described

by modified Denavit–Hartenberg parameters.^[20] This observation eases the application of controllers for rigid robots to CSMs.^[21]

As can be seen from the aforementioned assumptions for PCC, this model may not be applicable to every task or CSM. For this reason, other models with increasing levels of detail have been created, such as variable curvature,^[22] Pythagorean hodograph,^[23] cubic Hermite splines,^[24] or polynomial curvature (Euler's spirals).^[25] Clearly, the more the level of detail in the model increases, the more the formulation, identification, and computational cost also grows.

For all of these models, FK similar to Equation (7) can be computed. Consequently, it is possible to write the DFK in Equation (2) by differentiation and the FD in Equation (3) through the Euler–Lagrange equation.

2.4. Discrete Models

The main concept of discrete models is to discretize the continuum body directly, finding an equivalent rigid system. The main two methods are referred to as lumped-mass and pseudo-rigid models.

The former method consists of approximating the CSM as a set of N lumped masses, springs, and dampers, as represented by Figure 2d. In this case, the vector $q \in \mathbb{R}^N$ collects the displacements in the space of every lumped mass. Consequently, the FD can be computed as

$$M\ddot{q} + D\dot{q} + Kq = F_{\text{ext}} \quad (8)$$

where F_{ext} includes the gravity effect.

The main advantage of this method is the simplicity of the representation. In addition, this model is particularly suitable for modeling hybrid kinematic chains (i.e., rigid and soft links). However, to accurately describe the behavior of a CSM, the number of required lumped masses is typically very high, negatively impacting the computational efficiency of the model. A simulator that implements the lumped-mass approach is Soft Motion (SoMo).^[26]

The pseudo-rigid approach for modeling CSM involves treating them as hyper-redundant rigid robots. This approach allows designing controllers using the standard controllers from rigid robotics directly.^[13] However, this strategy does not perform well in terms of spatial accuracy and requires a great number of DoFs.

2.5. Continuum Mechanical Models

With the advancement of research on soft robotics, numerous tasks involving contact with objects and cluttered environments have been proposed. In this case, shear deformation is not negligible and neither geometrical nor pseudo-rigid models account for this type of deformation. Moreover, most of these approaches cannot accurately describe the shape of CSMs, because of misrepresenting twist deformation.

To tackle these issues, it is possible to use continuum mechanics theory to describe the elastic body of the robot. However, implementing the general theory can be computationally onerous and incompatible with real-time controllers. In this context, many researchers have proposed various approximations and spatial-discretization techniques to ease the design of controllers for real-world CSMs.

2.5.1. Cosserat Rod Theory

Under the assumption of a slender body, the CSM can be treated as an elastic rod and described by Cosserat rod theory (CRT).^[12,27,28] The main idea behind CRT is to associate each cross section s with a reference system, in which the axes are known as directors. Each cross section can translate and rotate with respect to the neighboring cross sections (Figure 2e), capturing bending, twisting, shear, and elongation/compression. The FK of the soft filament can be found as a solution of the following differential equation:

$$g'(s, t) = g(s, t) \hat{\xi}(s, t) \quad (9)$$

where $(\cdot)' = \partial/\partial s$, $\hat{(\cdot)}$ the hat operator,^[29] and $\xi = [\kappa^T \sigma^T]^T \in \mathbb{R}^6$ is the strain twist, composed by the angular $\kappa \in \mathbb{R}^3$ and linear strain vector $\sigma \in \mathbb{R}^3$. The former represents bending and twisting and the latter represents shear and elongation/compression deformations.

The DFK can be derived by spatial integration of the mixed partial derivative equality,^[27] resulting in

$$\eta(s, t) = \text{Ad}_{g^{-1}} \int_0^s \text{Ad}_g \xi(\epsilon) d\epsilon \quad (10)$$

where $\text{Ad}_{(\cdot)}$ is the adjoint operator.^[29] Finally, the FD can be computed by applying Hamilton's principle and obtaining a set of PDEs. Generally, the solution can be found only through numerical methods, with the main goal being to find the optimal trade-off between accuracy and computational efficiency. Various methods exist in literature, such as discrete elastic rod (DER)^[12] and strain parameterization.^[30,31]

The main idea of the former approach is to consider the elastic rod divided into a N finite set of nodes, connected by straight line segments. The equation of motion (EoM) is then computed by evaluating the internal and external forces applied to each node. This algorithm is implemented in the Elastica simulator,^[32] in which the user can find a variety of tools for learning-based controllers.

In contrast to DER, the latter method is based on a discretization of the configuration space. Since strain twist belongs to a functional space, $\xi(s, t)$ can be generated by a basis matrix of functions in s . The idea is to truncate the basis matrix to a finite number of columns n ,^[30] in such a manner that

$$\xi(s, t) = B_q(s) q(t) \quad (11)$$

where $B_q \in \mathbb{R}^{6 \times n}$ is the truncated basis matrix. In this case, it is possible to neglect a specific deformation (e.g., twisting) and choose the desired degree of accuracy. This discretization technique is implemented in the SoRoSim simulator^[33] in MATLAB, easing the development of controllers.

2.5.2. Finite-Element Method

To describe a general 3D elastic body with the continuum mechanics theory, it is necessary to solve a set of PDEs. For this purpose, one of the most common numerical methods is the finite-element method (FEM). In this approach, the continuum structure of the body is discretized using finite 1D elements (e.g., lines), 2D elements (e.g., triangles or quadrilaterals), or 3D elements (e.g., tetrahedra or hexahedra). The discretized structure is called a mesh and it contains all the elements for which the numerical solution can be computed. The value of a continuous function on each element is approximated by interpolating the values at the edges of the same elements, called nodes (see Figure 2f). In FEM, the statics and the dynamics are solved by evaluating the internal and external forces acting on the considered body. In particular, the FD is given by the system of equations

$$\begin{cases} \dot{q} = v \\ M\dot{v} + \mathcal{F}_{\text{int}} = \mathcal{F}_{\text{ext}} \end{cases} \quad (12)$$

where v and q are the vector of nodal velocities and positions, respectively. \mathcal{F}_{int} contains the internal forces due to the elastic and damping effects, and \mathcal{F}_{ext} contains the external forces applied to the nodes of the mesh.

To have an accurate solution of the dynamics, the number N of finite elements is usually large, implying an excessive computational burden for real-time applications. To solve this issue,^[34] proposed an asynchronous FEM that satisfies real-time constraints. In particular, the algorithm is formulated in two multi-rate loops. The former is a low-frequency loop

(i.e., 15 Hz) that manages the softness of the material and the latter computes the inverse model through a quadratic programming optimization at a very high frequency (i.e., 600 Hz). The algorithm is implemented in the SOFA^[35] simulator, a very popular simulator that implements FEM with a various number of tools and algorithms for accurately simulating soft robots.

2.6. Modeling Actuation

The distinction between rigid and soft robotic models is especially pronounced in terms of actuation. In particular, the most common technologies are cables^[27] and fluidic chambers,^[36] which guarantee distributed active loads along the length of the soft arm.^[37] The main difference between the two actuation sources in terms of deformation is that the cables can only compress the CSM, while the fluidic chambers can also elongate the CSM, significantly affecting the workspace of the arm. Furthermore, in real CSM prototypes, the soft actuators exhibit internal dynamics (e.g., limited bandwidth, delay) that can lead to nonlinear behaviors such as hysteresis^[38] and a delayed response. Additionally, to achieve desired deformations (e.g., twisting), actuators can be routed along specific paths within the continuum body. To address these challenges and integrate actuation into existing models, numerous researchers have provided various kineto-static models as outlined later for geometrical models, Cosserat-based models, and lumped-mass models.

In geometrical models, the actuation is modeled as a geometrical mapping between actuator lengths and the deformation of the CSM. In ref. [19], an actuation model for a CSM actuated by three tendons is presented under the PCC approximation. Recently, this model has been extended in the case of helical routing, allowing twisting deformation.^[39]

In the context of Cosserat-based models, Renda et al.^[40] proposed a model for general routing actuators, taking into account shear and stretching deformations induced by the actuators. More specifically, the distributed active load $\mathcal{F}_a \in \mathbb{R}^6$ can be written as

$$\mathcal{F}_a(s) = \mathbf{B}_\tau(\xi, \mathbf{d}_i(s))\tau \quad (13)$$

where $\mathbf{d}_i(s) \in \mathbb{R}^3$ is the distance from the cross section's center of the i th actuator. Regarding FEM-based models, the actuation is accurately described by considering the excited deformations on the cross sections, hyper-elasticity,^[41] and a model of hysteresis.^[38]

Finally, in lumped-mass models, the distributed actuation is applied to the series of mass-spring systems as an external force. Lumped-mass models also ease the description of nonlinear phenomena related to actuation, such as cable friction.^[42]

2.7. Simulators for CSMs

As mentioned in the previous sections, the theoretical frameworks proposed by different researchers are implemented in a wide set of simulators. Each of these simulators is characterized by 1) the modeling approach used, 2) implementation details (e.g., programming language), 3) the presence and the possibility of simulating contact forces, and finally 4) an experimental validation. These aspects are summarized in Table 1.

Table 1. Simulators for CSMs.

Simulators	Modeling approach	Contact forces	Implementation	Experimental validation
PyElastica ^[32]	CRT	✓	Python	✗
SoRoSim ^[33]	CRT	✓ (Custom)	MATLAB Toolbox	✗
SOFA ^[35]	FEM	✓	C++	✓ ^[43]
SoMo ^[26]	Lumped mass	✓	Python	✓ ^[26]

The choice of the discretization technique is crucial to balancing computational efficiency and accuracy. Many discretization techniques limit the frequency of the control loop, to preserve the numerical stability.^[32] In addition, some approaches neglect shear or torsion strain modes, which can be useful for reducing the complexity of the model or may be required for tasks that involve interactions with the environment. In this context, many simulators implement contact forces, increasing the realism of the simulations. In the case of SoRoSim, the contact forces are implemented as custom external forces and the user must adapt them for the specific application.

Simulators are usually validated through experiments to quantify and decrease the sim-to-real gap. In particular, SOFA received extensive validations in various robots.^[43,44] SoMo implements a calibration algorithm, reducing the sim-to-real gap and allowing accurate simulations. SoRoSim is validated through numerical examples, using high-accuracy FEM simulation. PyElastica, although not validated by experiments, was effectively used to reproduce several musculoskeletal architectures in simulation^[45] and as the basis to simulate soft robotic arms.^[46]

Thanks to the pseudo-rigid modeling approach, other simulators can be adapted to CSMs. In the works,^[47,48] the authors adapted MuJoCo^[49] as a training environment for their learning-based controllers.

Finally, the simulators are also different in their implementation. In particular, the programming language used can significantly increase the simulator's performance, satisfying real-time constraints. To the best of the authors' knowledge, there are no simulators based on ML or geometrical approaches. The control designer must implement the physics engine from scratch, using the EoM of the chosen approach.

3. Learning-Based Controllers for CSMs: Characterization and Evolution

ML has been proposed as a tool to overcome limitations due to nonlinearity and hysteresis of deformable materials.^[50] Indeed, complex behaviors that arise from the intrinsic nature of these materials can then be modulated by learning-based algorithms. Typically, the method requires defining a control policy π to act as a mapping from observation $\mathbf{y} \in \mathcal{O}$ to actuation $\mathbf{u} \in \mathcal{A}$, to achieve either a desired configuration \mathbf{q}_d or task-space goal \mathbf{g}_d :

$$\mathbf{u} = \pi(\{\mathbf{q}_d, \mathbf{g}_d\}, \mathbf{y}; \mathbf{w}) \quad (14)$$

The policy parameters w can be acquired via either SL or reinforcement learning (RL), and the performance is contingent upon the strategy employed for constructing the learning dataset.^[9] Such dataset is mathematically defined as $\mathcal{H} = A \times O$, built upon pairs of commanded actuation u and resulting observation y . Supervised approaches train ANNs to minimize the loss between the target $\{q_d, g_d\}$ and the actual observation y , exploiting labeled information to implement generalizable control strategies, for which the dataset \mathcal{H} is generalizable and informative to learn the task. The performance of the controller will be highly dependent on how \mathcal{H} has been sampled. Conversely, in RL, the desired behavior is defined by a reward function, and the agent learns to maximize it by trial and error while interacting with the environment. In both cases, the learning agent aims to exhibit optimal behaviors based on the experiences garnered from the training dataset, simultaneously evaluating generalizability on not-yet-experienced events. Additionally, the complexity of formulating a learning-based controller is tied to the forward robot representation utilized in the training loop. The accuracy of such a representation to the actual robotic arm significantly impacts the control outcome, giving rise to sim-to-real challenges when the control is deployed on the physical prototype.^[51,52] The choice of learning strategy depends on the generalizability of the dataset, as shown in **Figure 3**. Once the task for the CSM is chosen, we need to determine if the dataset from the robot is generalizable and informative enough to implement the task. For instance, generating \mathcal{H} is straightforward for tasks that only involve payloads and do not require interaction with the environment (like external forces or contacts). In such cases, we can use either SL with the dataset \mathcal{H} or RL with the simulation environment to learn the controller. If the forward model also includes the task's physics, it acts as the complete simulation environment. However, if the control strategy is unattainable because the task involves more complex interactions and \mathcal{H} cannot be generated, RL is necessary.

In general, as depicted in **Figure 4**, it is evident that the precision of the learning-based controller can be observed for a specific target $\{q_d, g_d\}$ within either the configuration C or the task T spaces. However, this is not always true: while it holds for an SL task, where the controller guides the arm toward a desired state, it may not necessarily apply to RL-based controllers. The latter aims to learn a policy solely dependent on the environmental

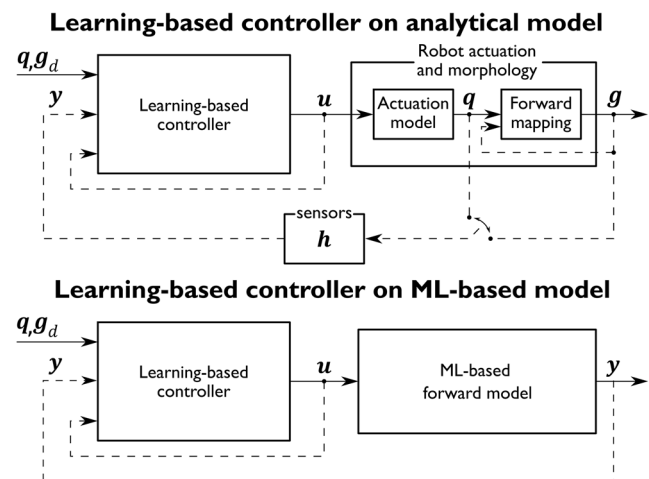


Figure 4. Learning-based controllers for CSMs using analytical or ML-based models. Both schemes take as input in an open loop the target configuration $q_d \in C$ or pose $g_d \in T$, and output the actuation $u \in A$. Optionally, they consider actuation dynamics in closed loop. Their difference lies in the forward model and how the observations $y \in O$ are obtained. (Top) In analytical models, an actuation model maps the actuation space A to the configuration space C , while a forward mapping gives the task space T . Here, sensors or an observation function h provide the observations y . (Bottom) In ML-based models, a neural network directly maps the actuation $u \in A$ to observations $y \in O$, which may include information about various robot spaces depending on the sensorization employed when the forward model was trained. Therefore, the observation function is encapsulated within the ML-based forward model, as well as the actuation model.

observation y , contingent upon how the observation function h is analytically modeled or implicitly learned by the ML-based forward models, and whether the feedback loop is closed in either the configuration C or the task T spaces. Herewith, the feedback on y assumes significance in distinguishing between open-loop and closed-loop controllers. In an open-loop setting, y is not fed back, forcing the controller to provide control actions deemed optimal during the training phase without guaranteeing convergence to the target. Conversely, closed-loop controllers relying on sensors information monitor the robotic arm's evolution overtime, delivering a control action that can ensure both

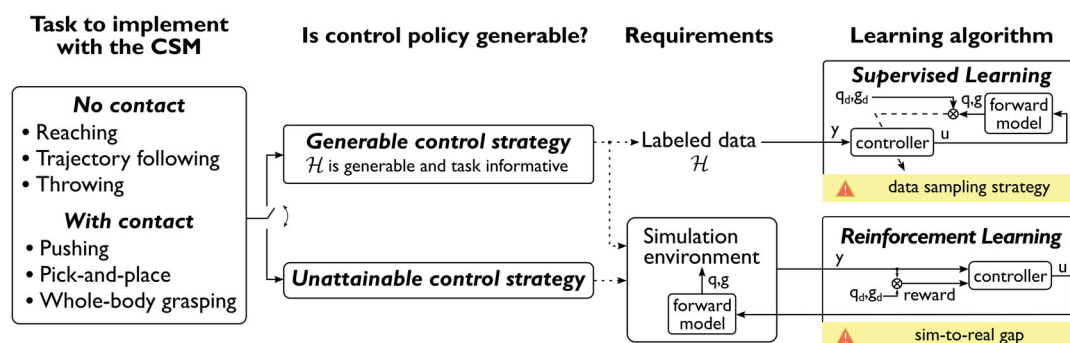


Figure 3. Learning CSMs controllers using their forward model. The choice for the learning algorithm depends on the control strategy generalizability for the selected task. We recur either to SL or RL. SL uses labeled data to train the controller by minimizing its discrepancy with the forward model output. RL requires a simulation environment where the controller optimizes the reward function through trial and error.

optimal performance and convergence to the target. Furthermore, this feedback need not be restricted to the last time step but may encompass a set of observations from the past. The learning-based controller may also receive as input a copy of previous output actions \mathbf{u} , particularly for decision-making in the dynamic domain.^[15,53] Such actions can be manifested in various domains depending on the employed model. Specifically, if the controller provides the control actions based on the configuration space, the actuation model may be considered as an identity.^[4] Similar arguments may also hold for controllers working with learning-based models, where observations may not strictly conform to an injective mapping to either configuration or task space.

In summary, the learning-based controller generates a control action, drawing information from a goal state in a specific space, from the historical data of actual robot states, and from the control dynamics. The learning agent addresses and integrates the inverse observation problem, concurrently mapping the observation state to the target space, performing an inverse mapping from the target space to the action domain, and correcting the action based on its historical context.

We next give an overview of the main features, advantages, and limitations of learning-based controllers for CSMs, highlighting the influence of the forward model or solution on their evolution.

3.1. Control from Learning-Based Forward Models

Learning-based CSM models can yield precise robot simulations, which would otherwise be challenging to derive analytically, even when provided with accurate system parameters, but with uncertainty about different physical phenomena that are missing from the model. Conversely, learning-based models may require extensive hyperparameter tuning and optimization time. Model parameters are updated according to the learning dataset; therefore, low errors depend on the availability of representative data that best encodes the robot mechanics and observed physical phenomena, such as friction and hysteresis.

The main advantage of learning a controller (by either SL or RL) over ML-based models concerns the computational efficiency of the forward model evaluation over its analytical counterpart. It also facilitates the training in simulation and model embedding in the deployment phase. However, collecting representative data is usually time consuming and brings the difficulty of labeling the data in the case of SL. In addition, the difficulty of generalizing the controller over unseen data might prevent the deployment of learning-based forward models in scenarios that differ from the training data. This is why such control strategies are currently limited to relatively simple tasks where the interaction with the environment is predictable and quasi-static.

Table 2 summarizes the ML-based controllers, particularly highlighting those experimentally validated on physical CSM. Initially, simpler tasks such as point reaching and trajectory tracking are considered. Then, their complexity increases in terms of actuation space and constraints, including modifications to the dynamics by adding a mass to the end effector (payload) or the body (weight) of the CSM. Finally, there is a shift in the literature toward more dynamic and interactive tasks, such as

throwing and simultaneous force and position control. Such evolution can also be observed with respect to employed learning strategies, as next paragraphs will explain. Indeed, both SL and RL have been first implemented for contactless tasks and further refined to include interactions and variable dynamics.

3.1.1. Supervised Learning

Several studies have proposed the use of SL methods to derive control policies for CSMs. However, their reliance on labeled data limits them to simpler tasks where informative data can be collected. Several ML methods have been used to model the IK of CSMs. For example, in ref. [54], the authors used a multilayer perceptron (MLP) to learn the IK of a simulated CSM for point-reaching tasks. Meanwhile, Melingui et al.^[55] developed a controller using distal SL to invert the forward model for trajectory tracking with a pneumatically actuated CSM. This approach resolves redundancy locally and lacks a feedback error correction scheme.

To derive more complex control policies, ML methods that rely on the learned forward model have begun to emerge. This approach avoids collecting data directly from the robot by simulating its behavior offline. Thuruthel et al.^[56] extended their previous work,^[57] where they used a Nonlinear AutoRegressive network with eXogenous (NARX) inputs formulation to learn the forward model of the CSM. Specifically, they used a feedforward neural network to learn the control policy for both dynamic point reaching and trajectory tracking.

Recently, more interactive tasks have been performed using SL methods. By combining multitask Gaussian process (GP) and locally weighted projection regression (LWPR), Tang et al.^[58] achieved combined position and force control with a CSM. This work was extended in ref. 59, where meta-learning was used to find the optimal combination of parameters for the GP, while the control policy was determined with model-based optimal control. Finally, in ref. 60, the accuracy of the task was improved by using a model predictive control (MPC) framework to compute the feedforward action, which was then combined with the feedback action found by LWPR to minimize tracking error, as reported in Table 2.

3.1.2. RL

The first step toward RL for soft robotic control was made by Malekzadeh et al.^[61] in an attempt to transfer reaching skills from an octopus arm to a simulated CSM. Despite being promising, it would have required a physical robot deployment to validate its potential. The need for RL-based controllers stems from the robustness/adaptability requirements for the task or the platform at hand. The challenges associated with developing these types of controllers are listed in refs. [14,50].

Approaches using RL within learned synthetic environments for tasks like reaching or trajectory following were proposed in refs. [63,64]. In the first paper, a recurrent FD model was trained using data generated from a safe, mean-reverting random walk in the actuation space, allowing for exploration of the partially observed state space. This forward model enables the efficient

Table 2. Learning-based controllers for CSMs. List of controllers applied to a physical CSM is reported, grouped by model and sorted by task. The task error column indicates 1) error interval (min–max, or <max); 2) error (avg) with its standard deviation (avg \pm std); 3) average error as a single value; or 4) N/A if the value is not available. In the “task” column, additional conditions posed to the task are indicated in brackets. The term payload refers to a mass attached to the end effector of the CSM, while weights refer to masses attached to the body of the CSM to modify its dynamics.

References	Year	Task	Model	Learning controller	CSM characteristics		Task error
					Dimension [mm]	Actuation (# inputs)	
Learning-based controllers from learning-based models							
[64]	2020	Point reaching (payload)	MLP	DQL	150	Tendon (3)	5–10 mm
[55]	2015	Trajectory tracking	MLP	MLP	500	Pneumatic (6)	5 mm
[56]	2019	Trajectory tracking (payload)	NARX	MLP	400	Pneumatic (3)	22 ± 22 mm
[15]	2022	Trajectory tracking (payload)	LSTM	TRPO	440	Pneumatic (6)	11–16 mm
[51]	2023	Trajectory tracking (payload)	LSTM	SAC + ORN	430	Pneumatic (6)	14–21 mm
[16]	2022	Trajectory tracking (weights)	RNN	CL + MLP	200	Pneumatic (3)	10.44 ± 0.89 mm
[52]	2024	Trajectory tracking (obstacle avoidance)	LSTM	PPO + GPRCA/ BOAC	598	Pneumatic (9)	<5 mm
[58]	2022	Position/force control	GP	LWPR	400	Pneumatic (6) + tendon (3)	<10 mm, <0.06 N
[60]	2024	Position/force control	GP	MPC + LWPR	375	Pneumatic (6) + tendon (3)	<5 mm, <0.05 N
[18]	2022	Throwing	MLP	PPO	409	Pneumatic (6)	61.7–68.3 mm
Learning-based controllers from geometrical models							
[68]	2017	Point reaching (planar)	Pretraining with PCC + ANN	Q-learning	660	Pneumatic (16)	8–59 mm
[65]	2014	Point reaching	PCC	MLP	800	Pneumatic (9)	7–20 mm
[67]	2021	Point reaching	Pretraining with PCC	Q-learning	660	Pneumatic (16)	20–23 mm
[66]	2016	Trajectory tracking	Geometrical model	MLP	500	Pneumatic (6)	8–11 mm
[70]	2024	Trajectory tracking (payload)	Pretraining with PCC + ANN	Q-learning	140	Pneumatic (6)	11.3–14.3 mm
[71]	2022	Trajectory tracking (payload)	Pretraining with PCC	DDPG	70	Pneumatic (3)	1.09–1.85 mm
[69]	2022	Pick-and-place (pose constraints)	Pretraining with PCC	Q-learning	660	Pneumatic (16)	<5° rotation error
Learning-based controllers from discrete models							
[74]	2023	Point reaching	Pseudo-rigid	PPO	500	Pneumatic (1)	N/A
[48]	2023	Crank rotation	Pseudo-rigid	SAC	690	Pneumatic (9)	N/A
Learning-based controllers from continuum mechanical models							
[77]	2020	Trajectory tracking	Rod (static)	DDPG	310	Pneumatic (3)	5–70 mm
[86]	2023	Trajectory tracking	FEM (dynamic)	RBF NN	108	Pneumatic (3)	0.44 ± 1.5 mm
[75]	2019	Trajectory tracking (payload)	Rod (static)	DQL	310	Pneumatic (3)	30.5 ± 9.7 mm
[76]	2020	Trajectory tracking (payload)	Rod (static)	DDPG	310	Pneumatic (3)	12 mm
[80]	2024	Pushing (pose/force control)	Rod (dynamic)	PPO + DR	440	Pneumatic (6)	34 ± 14 mm, 23° ± 17°

learning of high-performance control behaviors over longtime horizons, without requiring prior knowledge of the robot’s functionality or capabilities. In the latter work, data-driven modeling is integrated with RL to achieve position control of a soft robotic arm using deep Q-learning (DQL). To address slow convergence and instability in real-world applications, the authors developed a method that uses experimental data to create a simulation environment for training control strategies, which are then transferred to the real robot. An MLP model is built from this data, enabling RL to pretrain control strategies efficiently in simulation before deployment on real tasks.

The use of the ML-based forward model for RL training in trajectory-tracking tasks with a payload attached to the robot has been proposed in ref. [15]. In this work, the capability to adapt to variable payloads, without losing dynamic capabilities, has been trained via trust region policy optimization (TRPO), leveraging an approximation of the robot FD model obtained by an LSTM network.^[15]

More complex dynamic tasks have also been performed with deep RL controllers using the proximal policy optimization (PPO) algorithm and keeping an recurrent neural network (RNN) as a representation for the forward model of the task.^[18]

In this work, the authors train a controller to throw objects in target boxes with a CSM.

These approaches show some limitations related to the sim-to-real gap when the controllers trained on the forward model are applied on the real robot. We believe RL has the potential to enable robust decision-making capabilities. However, currently the expected performance falls short due to training environment inaccuracies and the soft robot's inherent stochasticity. One possible solution could involve online optimization agents capable of policy improvement in a few iterations or sample-efficient agents to specifically learn the difference in performance from different environments.^[51,52] In ref. [51], an online regressing network (ORN) is proposed to compensate for the sim-to-real gap after learning the controller with the soft actor critic (SAC) algorithm. In ref. [52], two methods are proposed to overcome the intrinsic stochasticity of the CSM. The first is based on GP-based recurrent cerebellar architecture (GPRCA), the second on Bayesian optimization-assisted coaching (BOAC). Both approaches allow reaching a final point with the desired accuracy while following a trajectory and avoiding obstacles up to 5 mm (see Table 2).

3.2. Control from Analytical and Numerical Forward Models

The integration of learning-based controllers with analytical and numerical forward models typically occurs using a physics engine, which accounts for contact forces and facilitates learning complex control policies π while interacting with unstructured environments.^[32,35] Combining analytical and numerical models and learning-based approaches offers the advantage of reducing the number of unknown parameters to estimate. This occurs as fundamental phenomena like gravity require no learning, and the robot's geometric and material properties can already be incorporated into the model.

These models can play a crucial role in enhancing SL or RL frameworks for controller learning. They can especially provide access to diverse data sets that are crucial for learning, and often difficult or impossible to obtain in real-world settings. However, employing analytical models for control also introduces challenges, as the forward models demand intricate mathematical formulations, particularly in explicitly describing actuator-induced deformations. Similar considerations can be done with numerical models (e.g., Section 2.5), where the main difficulty is the computational effort of solving the PDEs.

Despite these complexities, the effort is rewarded with highly interpretable results. Discrepancies in experimental data can typically be attributed to model approximations or unmodeled effects. Despite the benefit of interpretability, depending on the level of abstraction and the space-time discretization, these models are often computationally expensive, hindering real-time performance. Consequently, this limitation may slow down the learning process and preclude their practical deployment on physical platforms or microcontrollers. Herein, we report the evolution of learning-based controllers for CSMs leveraging geometrical (Section 3.2.1), discrete (Section 3.2.2), rod (Section 3.2.3), and FEM-based (Section 3.2.4) forward models.

3.2.1. Learning from Geometrical Models

Closed-loop tracking in the task space T using a planar CSM has been implemented by Rolf et al.^[65] The mapping between configuration C and task T space was obtained using the PCC approximation, while the mapping between actuation A and configuration C space was learned to account for viscoelastic effects. This control implementation enables accurate and reliable positioning of the end effector. The methodology is robust enough to handle inherent sensor noise, execution delays, and varying actuator ranges. A similar robot has been modeled as multiple sections with three universal-prismatic-spherical and one universal-prismatic joints DoFs.^[66] An approach based on ANN is used to provide approximated solutions of the IK for real-time implementation, to resolve redundancy, and to obtain the mapping from the task T to the high-dimensional configuration C space. However, the mapping from configuration C to actuator A space was performed analytically in a straightforward fashion. A noticeable limitation of such a method is the high amount of sensory information required. The IK is then used to synthesize a static controller for trajectory following.

There is growing interest in integrating RL with geometrical models for control systems. The adopted strategy involves the use of model data for the pretraining of model-free RL algorithms, which makes it possible to reduce the data collection time significantly. This strategy was adopted by Li et al.^[67] who proposed a model-free RL approach, Q-learning, for point reaching on a physical CSM. This method was pretrained using data from a PCC-based robot simulator. The results show that the Q-table that was pretrained in simulation achieved nearly the same performance as one pretrained with real robot data. Nonetheless, a significant advantage of using geometrical models is the decreased time needed to acquire data for pretraining. A similar approach is used by You et al.^[68] for planar point reaching. In this case, a two-level geometrical approach uses an ANN to map between actuation A and configuration C space, in combination with a PCC approximation to map between configuration C and task T space. This is then used to find a good training target set and to speed up the training process. Gan et al.^[69] extended this RL pretraining approach for path-following with pose constraints in pick-and-place tasks. In this case, the Q-table that was pretrained with geometrical model data was combined with convolution-replacing processing to meet pose constraints under multiple conditions of loads and interactions. The resulting control framework was successfully employed in tasks such as delivering a cup of water and a spoon with food. In contrast, a path following with payloads but without pose constraints was executed in ref. 70. To handle the presence of a payload, they decided to equip the PCC model with an ANN to fit the error between the geometrical model and the robot coordinate under various loading conditions, thereby reducing the FK MAEs. Such a model is used to build the RL virtual environment and the PPO algorithm was then used to train control policies. In another study, Li et al.^[71] used geometrical models to pretrain a deep deterministic policy gradient (DDPG)-based control system for continuous task-space manipulation with soft robots. Domain randomization (DR) in PCC-based simulations facilitated fast control policy initialization, while an offline retraining strategy was employed

to update controller parameters for incremental learning. The control policy was then applied to a path-following task with variable payloads, demonstrating the adaptive performance and stability under changing loading conditions along the path.

All the aforementioned papers are summarized in Table 2 ordered by task. The table also specifies some information about the CSM platform used and the maximum error reached by the controller. To summarize, learning-based controllers from geometrical models began with a direct use of the model in conjunction with ANNs to execute simple point-reaching and trajectory-tracking tasks without payloads. The subsequent combination with RL algorithms enabled increasingly complex tasks involving static and variable payloads, and pick and place.

3.2.2. Learning from Discrete Models

Discrete models simplify the representation of soft robots by treating them as a series of spring-loaded rigid joints (pseudo-rigid) or as mass-spring systems (lumped mass). While this approach makes the mathematical description of CSMs more straightforward, it also requires a high number of DoFs. This can pose challenges for learning algorithms that must manage high-dimensional control actions. Furthermore, discrete models generally account only for bending DoFs, often neglecting torsional or axial deformations.

In Table 2, we summarize the key aspects of the following works. Graule et al. introduce SoMoGym,^[72] an environment for training RL-based controllers on the physics engine provided by the SoMo simulator,^[26] showcasing the implementation of planar block pushing and reaching with obstacles using a continuum manipulator. Abondance et al. leveraged the same methodology to implement in-hand manipulation for a soft hand,^[73] where each finger has been modeled as an independent continuum slender body. The choice of the RL algorithm and the reward function shaping are crucial, whose appropriate selection enables rapid calibration of simulation parameters to match the hardware setup and reduce the sim-to-real gap. Notably, learned policies from a pseudo-rigid model can be effectively transferred to a physical prototype with minimal deviation during policy execution.

Pseudo-rigid models have also been employed by Morimoto et al. as a virtual representation of the actual robot to train RL-based controllers for point reaching^[47] and to perform tasks such as crank rotation, peg in hole, and ball throwing.^[48] Despite its use, the simulation in MuJoCo served only as a training environment for model-free RL. Conversely, Jitoshio et al. implemented a policy-learning approach using PPO that exploits the pseudo-rigid model to derive the joint torques from the actions commanded by the agent, which are subsequently used to command the simulation in Isaac Gym.^[74] A sim-to-real policy transfer is then applied to deploy the learning-based controller onto the physical robot.

Pseudo-rigid models, despite the approximation of continuum bodies as hyper-redundant, provide a good representation to learn controllers for CSMs. In particular, they are particularly beneficial to perform interaction tasks, under the assumption of rigid bodies.

3.2.3. Learning from Rod Models

Control during complex physical interactions requires models with higher fidelity to capture the deformability of CSMs. In such cases, leveraging continuum mechanical models is a promising solution. Furthermore, RL emerges as a natural framework to learn a policy π by letting the agent explore the best actuation by trial and error guided by a reward signal. Since continuum rod models can consider arbitrarily complex geometries and materials, the learning process can access rich information about the environment state $y \in O$, which would be limited using purely geometrical models or scarce with empirical ML models. Moreover, this approach accommodates the sim-to-real transfer of the policies with a moderate sim-to-real gap, provided the model matches the physical system well. This performance depends on the level of abstraction and the space-time discretization.

The first learning-based controllers for soft robots modeled using rod models involved static interactions due to payloads and did not address the sim-to-real gap. DQL with experience replay was effectively applied to learn an open-loop quasi-static position controller of a CSM capable of bending and twisting.^[75] The policy was trained on a static Cosserat rod model and was validated in simulation and on the physical platform subject to various external loads. This research was further extended by increasing the dexterity of the CSM and learning a closed-loop controller for precise quasi-static positioning using DDPG.^[76] Experiments in simulation investigated the robustness of the control policy for effects of loading, reachability, and workspace discontinuity. The controller was also deployed on the physical robot. The DDPG algorithm was also employed to derive a closed-loop controller for the quasi-static positioning of a CSM modeled with Kirchhoff rod theory.^[77] In addition, the CSM was combined with a mobile platform with a rigid arm and a sensorized gripper. The teleoperated system performed an agricultural task (picking berries) using different maneuvering strategies.

Recent research efforts investigated dynamic interactions with the environment leveraging simulators. Naughton et al.^[32] applied various deep RL algorithms to control a simulated CSM in PyElastica, a simulator of dynamic Cosserat rods, without using physical robots. The control tasks included point reaching, trajectory tracking, and maneuvering through structured and unstructured obstacles. The simulated CSM learned to navigate and adaptively interact with obstacles without explicit rewards. Also, Shih et al.^[78] proposed a hierarchical framework to coordinate multiple rod-based soft arms in simulation. A high-level RL policy selected behavioral primitives like reaching and crawling. Then, they developed an energy-shaping controller using ANNs to actuate the arms. Challenging simulations of a foraging CyberOctopus in an arena littered with obstacles validated the framework. The PyElastica simulator was also adopted to model a 3D-printed pneumatic CSM by Alessi et al.^[46] Leveraging this model, a closed-loop control policy for dynamic trajectory tracking was learned in simulation using the PPO algorithm.^[79] Simulation tests evaluated how the policy generalized to new observations, dynamics, and tasks. Experiments in simulation included tracking trajectories subject to unknown external

forces, using different materials, or intercepting a moving object. Recently, the authors extended the simulation environment to consider the dynamics of a dexterous CSM interacting with the environment.^[80] Herein, an RL policy for pose/force control enabled dynamic pushing in the real world. The sim-to-real transfer of the policy was facilitated by DR, emphasizing soft materials and contact properties, demonstrating the benefit of continuum mechanical models in interaction tasks.

Table 2 reports the controllers that use continuum mechanical models with experimental validation on physical robots. In summary, the literature began with quasi-static tracking tasks using kinematic rod models of single-section CSMs with and without payloads, and it is now shifting toward employing dynamic rod models of multi-section CSMs for physical interaction tasks.

3.2.4. Learning from FEM

A promising alternative to rods for continuum mechanical models is presented by FEM. FEM can provide highly accurate predictions of soft robot deformations. A recently introduced open-source software, SofaGym, combines FEM with RL for various control tasks.^[81] Here, Ménager et al. showcase the capability of controlling soft robots within a learning environment, managing deformations caused by soft actuators, as well as their interactions with the environment. For instance, leveraging the SofaGym framework, Agabiti et al.^[82] proposes a whole-arm grasping control strategy implemented by an elephant-trunk-inspired soft arm for the application of space debris capture. Candidate contact points on the target object are employed to induce the whole-arm grasping configuration of the arm. The policy was trained in simulation, and simulation tests evaluated its capability to generalize to unknown object sizes and positions. Recently, Ménager et al.^[83] proposed a method for soft robot control involving the definition of a graph of configuration spaces. The method is formalized as hierarchical RL, and they used contact configurations of a CSM with a rod to generate the configuration spaces, improving the learning efficiency. Additionally, Ménager et al.^[84] introduced a motion planning method for soft robots that utilizes proxies as simplified models of the soft robot. These proxies are employed to generate feasible trajectories in the configuration space through RL. The strategy is then applied to the full model of the CSM to determine the corresponding actuations. Lastly, in a recent work,^[85] RL is used to train a soft robot in a simulated environment with DR, which introduces variations in the simulation parameters to make the learned policy robust to real-world uncertainties. This trained policy can be eventually deployed on the real robots for closed-loop control. Concerning learning controller over FEM model, Zhang et al.^[86] present the control architecture validation on a physical soft robotic platform. A low-order dynamic model of a pneumatic soft robotic arm is first derived using FEM. Based on this, an adaptive radial basis function neural network is developed for trajectory-tracking tasks, with low tracking errors as shown in Table 2. In summary, learning-based controllers derived from FEM models began with the use of RL algorithms and have been successfully applied to complex tasks involving interactions with the environment, such as grasping and manipulation. Various methods, including hierarchical RL and proxies, have improved

control and motion planning. These approaches have been tested in simulations demonstrating robustness and accuracy of the control architecture.

4. Challenges and Perspectives on Learning-Based Controllers for Soft Manipulators

Despite the promise of CSMs for real-world, forceful, and unmodeled interactions, this reality has still not been achieved in many difficult applications. This is not likely because soft robots are incapable of achieving the desired outcomes, but because of remaining challenges in modeling (both the dynamics of soft robots and contact interactions), in changes to the system overtime, in careful integration with perception, and in interpreting learned controllers. The following sections outline specific challenges, summarized and commented in Table 3, which, when addressed, will advance state-of-the-art results for real-world deployment.

4.1. Sim-to-Real Gap

The evolution of the state of the art highlights the impact of forward models in learning-based controllers. However, policies trained only in simulation are likely to face a performance drop when transferred to the physical world due to the approximations made by simulated environments. This phenomenon is known as the sim-to-real gap.^[87] We must mitigate these gaps to achieve effective sim-to-real policy transfers. The three major strategies are 1) improving forward models, 2) leveraging sim-to-real techniques, and 3) learning directly on the physical soft robot.

4.1.1. Improving Forward Models

The first thing to consider to reduce the sim-to-real gap is to improve the forward models and the overall fidelity of simulated environments. We can achieve this through research efforts in modeling, system identification, and data collection.

For example, if actuation effects cannot be assumed instantaneous (i.e., they have their own dynamic effects), detailed transient models of actuation systems^[88] and refined damping mechanisms could improve the overall accuracy in dynamic experiments.^[46] Coupling between the configuration variables and the actuation variables can also cause significant discrepancies, such as in the case of pressure-controlled variable volume actuators. Moreover, as hysteresis can be significant in soft materials, forward models capturing hysteretic behaviors could increase the accuracy when performing prolonged operations.^[89] In addition, to bring CSMs closer to contact-rich and distributed interactions with unstructured environments in the real world, contact modeling will have a considerable impact, especially for interacting with soft deformable objects.^[35]

Another promising frontier to address the reality gap is hybrid modeling. It combines analytical and ML-based models, unifying their advantages while compensating for identified drawbacks. Although no prevalent methodology has yet been identified, three main research lines can be highlighted. In the complementary case, ML and analytical formulations can represent specific

Table 3. Challenges for learning-based controllers for CSMs have been identified and the main issues in tackling them are summarized in the second column. Perspective research and necessary actions are then reported as well as their expected impact.

Challenge	Current issues	Perspectives and actions	Impact
Sim-to-real gap (Section 4.1)	<ul style="list-style-type: none"> • Large deformations and hysteresis^[89] • Analytical modeling inaccuracies^[27] • Sim-to-real gap 	<ul style="list-style-type: none"> • Transient models of actuation^[88] • Integrate physical principles with ML, toward hybrid models^[90,92] • Domain adaptation^[51] and DR^[80] • Learning on physical robots,^[104] with initial training in simulation^[101] 	<ul style="list-style-type: none"> • Enhance accuracy and reliability of controllers • Deployment of CSMs in practical tasks, reducing associated cost and time
Real-world robot deployment (Section 4.2)	<ul style="list-style-type: none"> • Contact estimation (force, location, etc.)^[110] • Variation of dynamic parameters overtime^[16] • Asynchronous and novel tasks must be learned by the robot^[114] 	<ul style="list-style-type: none"> • Realistic and tractable contact modeling^[32] • Simulations of dynamic environments^[32,33,35] • CL to adapt to changes overtime^[16] • Incrementally learn novel tasks^[146] 	<ul style="list-style-type: none"> • Expand the range of tasks CSMs can handle • Broader applications in environments where CSMs must operate reliably under varying conditions
Robot perception (Section 4.3)	<ul style="list-style-type: none"> • Inaccuracy of multimodal sensing^[128] • Modality-specific data handling algorithms^[131,135] • No direct coupling with robot action, and lack of prediction abilities^[136] 	<ul style="list-style-type: none"> • Use of differentiable filters^[133] and generative models^[136] • Cross-modal inference^[126] and deep learning for high-dimensional data^[135] • Optimal sensorization to improve accuracy and robustness^[121,122] 	<ul style="list-style-type: none"> • CSMs capable of complex decision-making and adaptive behavior in real-world settings • Increase the versatility and reliability of CSMs
Learning-based controllers interpretability (Section 4.4)	<ul style="list-style-type: none"> • Explanation of decision-making process of learning-based controllers^[143] • Safety^[138] 	<ul style="list-style-type: none"> • Provide global and local explanations^[141] of ML-based models and control • XAI for controller refinement and to enable trusting controllers decisions^[145] 	<ul style="list-style-type: none"> • Enhance interpretability in safety-critical applications • AI-driven CSMs with transparency and accountability

parts of the system. Here, the typical approach is to use ML models to map the transformation from actuation A to configuration C space, while the analytical model maps configuration C to task T space.^[17,65,68] Another approach integrates physical principles given by the analytical model in ML templates to increase accuracy and computational efficiency for system identification, widely used in physics-informed neural networks.^[90,91] As an alternative, ML can compensate for the error introduced by the assumptions of analytical formulations while modeling real-world scenarios.^[70,92,93]

Concerning system identification and validation, detailed FEM simulations or extensive experimental motion data should be leveraged to find the optimal set of model parameters that reproduce the expected behaviors. Using FEM, simulations are conducted in a controlled environment, enabling the manipulation of parameters and conditions to understand their effects on the robot's behavior, without the risk of damaging the physical hardware. Therefore, FEM could be a valid intermediate step for parameter identification, rather than directly learning the model on the robot itself. In addition, detailed FEM could serve as a validation tool to ensure robot models reliability before deploying them in practical experiments. Also, considering data from mechanical tests might further increase the simulation fidelity in controlled settings. Finally, researchers could evaluate the contributions of model components through ablation studies.^[27,46]

In the case of ML-based models, the sim-to-real gap decrease might also depend on the quality of the learning data. Achieving convergence of the forward model to the actual robot mapping necessitates comprehensive coverage of the robot's workspace during data collection. Inadequate modeling performance can lead to subpar control capabilities. Currently, the workspace of a CSM is typically achieved through pseudo-random actuation^[15,16,46] or predefined actuation patterns empirically designed to cover the volume of interest. However, these approaches do not guarantee optimal and complete workspace coverage within a finite time frame, resulting in highly time-consuming tasks.

Consequently, effective and time-efficient data collection strategies are crucial for accurate modeling. Future research could focus on achieving optimal workspace coverage by minimizing unexplored areas. This may involve implementing closed-loop exploration strategies that guide the robot's motion toward poorly explored sections of the workspace^[94] with an existing controller, akin to methodologies employed in active perception.^[95,96] Such an approach aims to maximize the information in collected data uniformly across the workspace, thus achieving optimal coverage within a finite time frame. The choice of which areas of the workspace to explore can be either comprehensive coverage of the entire CSM for a general-purpose model or tailored to specific tasks. Furthermore, understanding the types of trajectories (in input or configuration space) that are necessary to model a CSM, and the trade-offs between the levels of required data

and required accuracy (in simulation or on hardware) may better guide future efforts.

4.1.2. Learning Data and Models for Sim-to-Real Gap Mitigation

Despite all of the proposed efforts in modeling and system identification, the simulation may still only roughly approximate the physical soft robot and its interactions with the environment. A residual reality gap could persist due to unmodeled factors and calibration shifts that may occur because of the physiological degradation of the soft materials. Nonetheless, there are several ways to aid the sim-to-real transfer of the control policy.^[97]

One prospective method to mitigate the remaining reality gap in soft robotics is to augment the policy training with DR.^[87] Rather than training on one simulation and transferring the policy directly to the real world, DR trains on a distribution over simulations obtained by randomizing the simulation parameters with high uncertainty. The process usually randomizes physics and visual properties, including injecting noise in the observation space O and actuation space A . In soft robotics, DR achieved the sim-to-real transfer for modeling an optical tactile sensor,^[98] vision-based pose estimation,^[99] position control in trajectory tracking using a PCC model,^[71] and pose/force control in dynamic pushing using a Cosserat rod model.^[80]

Conversely, domain adaptation usually transfers the knowledge learned in simulation (source domain) to the physical environment (target domain), unifying the different observation spaces $O_S \neq O_T$.

An alternative approach to address the sim-to-real gap is to employ imitation learning (IL).^[100] This class of algorithms relies on expert demonstrations of the desired task without any bias regarding the robot space. These algorithms circumnavigate the sample inefficiency of RL. However, their adaptation is limited by the quality of the expert teachings, requiring an additional synthetic agent for generalization improvement.

Soft robots increase this requirement due to their inherent stochasticity.^[101,102]

Other viable techniques that could emerge to reduce the sim-to-real gap are meta-learning and knowledge distillation.^[97]

4.1.3. Learning Directly on the Physical Soft Robot

One way to avoid the sim-to-real gap, especially when practical forward models are unavailable, is to train the policy directly on the physical robotic platform. **Figure 5** provides an overview on how learning algorithms can be optioned according to the control policy to generate directly on the robot.

A few attempts were presented using tabular RL algorithms. In particular, Q-learning-derived position controllers were used for a CSM in simulation and directly on the soft robot subject to tip loads.^[103] Interestingly, they also used the action-value function learned in simulation as an initial estimate for training on the real robot. Similarly, the State-Action-Reward-State-Action (SARSA) algorithm with tile coding obtained a static controller of position and stiffness for a CSM.^[62] Concerning SL, Bianchi et al.^[104] performed throwing tasks with a pneumatic CSM by directly learning the inverse model of the task from the physical robot. Similarly, Chen et al.^[105] studied the interchangeability of soft robot modules presenting a hybrid adaptive controller for trajectory tracking, combining an LSTM trained from offline motion data with an online optimization-based kinematic controller. Figure 5 also introduces IL as an alternative learning methodology for CSMs controllers. Oikonomou et al.^[106] present a control architecture based on a library of learned probabilistic movement primitives. Assuming that trajectories are a composition of individual primitives, they propose a path segmentation process to partition demonstrations collected with a human hand. Combining an incremental GP regression (GPR) to approximate the IK, they qualitatively reproduced human demonstrations in a planar task. In this case, the

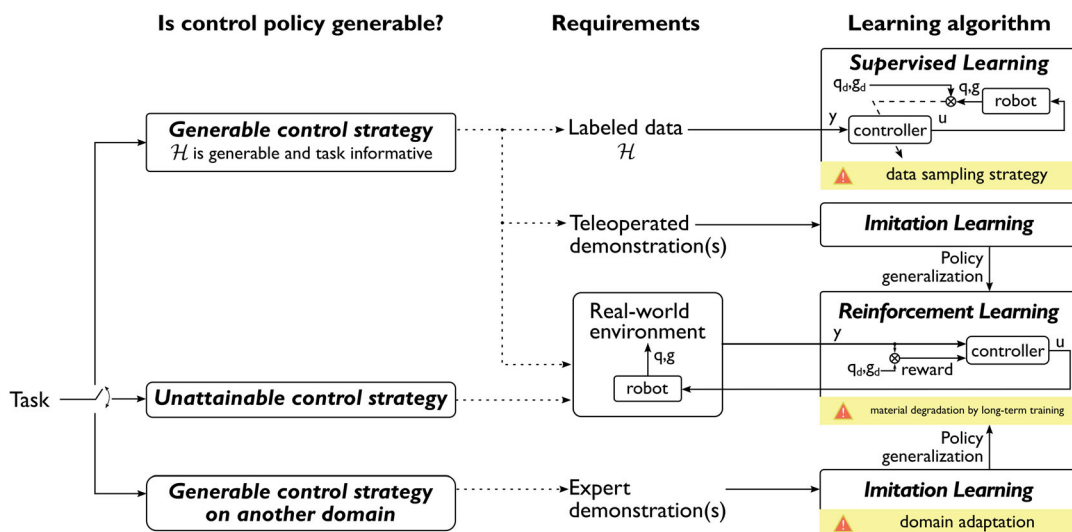


Figure 5. Learning CSMs controllers directly on the robot and the real-world environment. Such shift from methodologies reported in Figure 3 eliminates the need to mitigate the sim-to-real gap. Moreover, IL allows to learn control policy from (teleoperated) demonstrations directly on the robot or from experts in other domains yet requiring domain adaptation.

control strategy demonstration has been generated on a domain that does not coincide with the robot, thus requiring for domain adaptation to translate information. Also, Nazeer et al.^[101] proposed an efficient approach for training soft robot controllers, employing an incrementally improving dynamic behavioral map of the underlying CSM, mapping task space T to the actuation space A . The solution was validated for the task of writing letters.

Learning directly on the soft robot brings pros and cons. A possible advantage could be that it is not required to model the complicated mechanics (Equation (3)) and the physical interactions. Indeed, the physical world already offers the most accurate representation of all the phenomena involved. As a result, the sim-to-real gap is already closed, and the focus shifts entirely to efficient episode collection and training. Theoretically, arbitrarily complex policies π could be learned provided accurate and sufficient sensory information. However, the limitation of this approach is that the physical trials are generally expensive. In particular, sequential data collection cannot be faster than real time, and parallelization with multiple copies of real soft robots may be difficult and costly. Moreover, training might be dangerous (to the environment or the robot) as the learning policies tend to behave randomly in the initial phases. In addition, intrinsic problems of soft materials (like wear and tear) and logistics (like resetting after each episode without an existing controller) make this approach not very feasible. For these reasons, rather than training a policy from scratch in the real world, it is preferable to jump-start the training process in simulation using any available forward model and then conclude the training on the physical platform.

4.2. Real-World CSM Deployment

Recent advancements in soft robotics have led to the emergence of startups producing commercial soft robots, these products are primarily limited to simpler devices like soft grippers, which require minimal control to perform basic tasks. Specialized CSMs, such as I-Support^[107] and STIFF-FLOP,^[108] are still relatively rare and remain largely in the research and prototype stages rather than being ready for widespread commercial use. Although the hardware development of CSMs has progressed significantly, enabling the creation of flexible and adaptive manipulators, the corresponding control strategies have lagged behind. Currently, most CSMs are still designed as general-purpose devices with open challenges in developing advanced control algorithms that can manage the highly nonlinear and compliant nature of these manipulators. As a result, existing control strategies are relatively basic and are primarily capable of handling simple tasks, as demonstrated in Table 2. A critical aspect is accurately estimating contact forces in unstructured environments. Specifically, both modeling and control domains are aggravated by undesired deformations and nonlinear behaviors of CSMs. Moreover, the soft materials used in CSMs, while providing flexibility and adaptability, are prone to degradation overtime, particularly when subjected to repeated mechanical stress or harsh environmental conditions. This degradation can lead to changes in the mechanical properties of the

CSM, further complicating control efforts and reducing the operational lifespan of the device. These material limitations, combined with the challenges in control technology, currently restrict CSMs use in real-world applications.

4.2.1. Contact Estimation

To exploit the compliance of CSMs in real-world tasks, it may be necessary to estimate the contact forces in an unstructured environment. However, to achieve this, there are two main difficulties from the modeling and control side. The former is based on the presence of shear deformations and elastic instability (i.e., buckling) due to the contacts. As seen in Section 2.1, since only the continuum mechanical models describe these types of deformation, the control designer must utilize only specific simulated environments for the training phase, such as PyElastica,^[32] SOFA,^[35] and SoRoSim.^[33] From the latter side, the control framework must extract meaningful information about the contacts in real-time, countering undesired deformations.

To tackle these issues, researchers can exploit data-driven sensor-fusion algorithms with proprioceptive and exteroceptive information, such as tactile and visual sensors.^[109] In addition, the advancement of contact modeling^[110] can be exploited in the training phase with realistic and meaningful data.^[32] Furthermore, motion control of actuated soft bodies with contact handling has been achieved in the context of inverse FEM simulations. An interactive control method has recently been proposed for the first time in soft robotics, based on FEM simulations of the soft robot and quadratic optimization to handle the linear complementarity constraint introduced by the contacts.^[111] Promising results have also been obtained by combining FEM for modeling the soft robot and RL for controlling interactive tasks,^[81] demonstrating the potential of RL to cope with complex scenarios. In addition, a CSM compliance can be exploited to intentionally execute specific tasks in the presence of contacts. For instance, a pushing task involves controlling the robot to push an object to a specific location on a surface. While seemingly straightforward, the control algorithm would need to interpret contact forces using tactile or visual sensors to counteract any undesired deformations while still achieving the desired displacement. As another task, pick and place in a cluttered environment extends the concept of pick and place to the realm of soft robotics. Apart from simply relocating an object from one point to another, the CSM must adjust its shape to navigate around physical obstacles or constraints, like the sides of the container in which the object is or has to be placed. One systematic approach to solve this task for multi-DoFs robots is to project the task space into the configuration space, thereby identifying a corresponding null space, whose belonging actions do not interfere in task space yet control the robot shape.^[112,113]

Finally, a CSM compliance can be exploited in the whole-body grasping task, in which the robot handles an object with its entire body or the entire manipulator. This capability can have a significant impact on any task that requires manipulating large, heavy, or unwieldy objects for tasks like logistics handling, search and rescue, or even senior care.

4.2.2. CL

As shown in Section 2 and 3, ML algorithms have been used extensively in soft robotics to model robots and develop controllers. Among the presented research, there is CL,^[114] a class of algorithms that promises to be more suitable for tasks where we want to 1) update a model without discarding previous information (catastrophic forgetting),^[115] 2) learn a sequence of tasks without storing their data, 3) learn multiple policies, or 4) learn from a stream of data that may change overtime.

Thanks to CL, it is possible to have robots that learn cumulative skills and that can progressively improve the complexity and diversity of the tasks they perform.^[114] Although CL approaches have mainly been used for object recognition,^[116] detection,^[117] and segmentation,^[118] there are some works where these techniques are also used for rigid robotic tasks, as seen in refs. [119,120].

CSMs will benefit from the use of CL on several levels. Because of the way they are manufactured, the sensors they use, and the materials they are made of, soft robots tend to change their behavior overtime. For these reasons, the behavior described by the data used to build the robot model may differ from the behavior shown during the testing phase of a control algorithm. This introduces additional uncertainties into the control policy, ultimately forcing the model to be rebuilt from scratch and the entire control strategy development pipeline to be repeated. The class of CL algorithms can help to develop a model capable of evolving and describing the behavioral changes of the robot. The ability to learn from new information without discarding previous knowledge can be used to learn new tasks. CL approaches could even learn multiple tasks with a single policy, generalizing well to different conditions.

One of the first steps in this direction is the pioneering work of Piqué et al.^[16] The authors generalize the CSM model to take into account the effect of different weights applied to the robot to modify its dynamics. They show how the proposed control architecture can improve its performance when exposed to previously experienced loading conditions. This solution is very useful in the context of soft robotics, where the robot is exposed to unknown interactions and unpredictable changes in its dynamics.

4.3. Robot Perception

The successful deployment of CSMs in real-world scenarios hinges on the intricate interplay between the interaction with the environment and, crucially, its perception and modeling of the world. Such perception forms the bedrock upon which goal-oriented decision-making processes are built.

Robot perception is tied to sensors outfitted on the robot and their strategic placement across its body.^[121–124] These decisions introduce two levels of complexity: first, the choice of sensors depends on the specific information required or modality desired, necessitating diverse sensor hardware. Second, their distribution across the body is pivotal for capturing localized information effectively.

The journey toward achieving robust CSM perception entails the adoption of a multimodal, distributed sensing. Integrating

vast and varied data streams is imperative to disambiguate information and mitigate disturbances inherent in sensor readings. This integration not only fosters redundancy but also bolsters the resilience of the perceptual process.^[125] Such resilience is facilitated through cross-modal inference techniques,^[126] where insights from one sensory modality are leveraged to inform inferences across others. This intricate process combines cues from diverse sensory inputs to construct a more comprehensive and coherent representation of the world. For instance, during interactions, robots should rely on a blend of vision for global observation, touch for localized sensing, and proprioception for internal body awareness.^[127]

Multimodal sensor-based robot models often rely on sophisticated fusion algorithms to integrate diverse data sources, which vary significantly in dimensionality, distribution, and sparsity.^[128] This diversity underscores the complexity of multimodal fusion, highlighting the need to synthesize each data stream into a state estimate,^[129] or to establish mappings across modalities to emphasize cross-modal reconstruction capabilities.^[130]

Multimodal fusion is commonly achieved through filters,^[131,132] that iteratively update state estimates based on observation sequences. Despite their effectiveness, these methods rely on analytical dynamic models, which can be challenging to obtain.^[133] In response, learning-based solutions have emerged to infer models from data distributions. Hybrid approaches, such as differentiable filters, allow for end-to-end learning while maintaining the recursive filter structure, particularly beneficial for handling high-dimensional sensor modalities with diverse characteristics.^[134] However, traditional fusion methods encounter challenges when dealing with multimodal data that is rich in inter-modality and cross-modal information. Deep-learning techniques offer a promising solution by automatically extracting and understanding associations within multimodal data.^[135] Generative deep-learning algorithms in particular learn representations and formulate inference spaces while considering complexity and reducing redundancy in heterogeneous data, albeit without supervision.^[136] Although generative models lack explicit label information, they infer implicit relationships, suggesting potential advantages in multi-sensor fusion tasks.

4.4. Interpretable Learning-Based Controllers

In Section 3, we explored the intricate learning strategies essential for robots operating in unstructured or complicated environments, necessitating the adoption of ML for modeling and control. While ML offers numerous advantages and applications, it often lacks interpretability,^[137] making it challenging to ascertain the rationale behind specific decisions and behaviors. Indeed, this issue is especially significant when attempting to grasp the process by which a CSM controller has been learned and how we expect it to interact with the environment. This importance is further amplified in scenarios such as ensuring compliance with safety guarantees^[138] or the imperative need to explain how a particular mapping (e.g., from configuration to task space or from observation space to action space) has been constructed in a comprehensible manner.

This necessity emphasizes the utilization of tools and methods from eXplainable AI (XAI), which facilitates the formal

understanding and measurement of explanatory factors in robot learning.^[139,140] In this context, model transparency is enhanced either globally, by examining how input features are mapped into the output and the underlying rationale, or locally, typically through post-hoc XAI techniques.^[141,142] Among various alternatives, such explanations might enable the exploration of control policies in terms of rule lists or decision trees,^[143] offering a human-readable understanding of the control policies for CSM systems.

Employing such a methodology would greatly benefit the learning-based control and modeling of CSMs, especially in understanding how they function and in optimizing their performance in terms of accuracy and precision. By scrutinizing the morphology of learning architectures and offering explanations of their operations, this approach can bridge the gap between learning-based and model-based approaches. It enables enhancement and explanation of learning-based methods in terms of latent variables used for learning,^[142,144] without being constrained by predefined parameterizations that are inherent in model-based solutions. In essence, it provides a physics-like explanation of learning variables, facilitating a deeper understanding of the underlying mechanisms. Within this field, incorporating pioneering research on large language models would offer a valuable contribution in terms of explainability,^[145] facilitating the translation of ML-interpretable decisions into human-readable contexts.

5. Conclusions

CSMs represent a great challenge for AI-based controllers due to the complexity of these robotic platforms. Most of the existing AI-based controllers for CSMs focus on simplified tasks such as trajectory tracking and point reaching. Model-based and model-free approaches have demonstrated, so far, similar performance. This may be because current learning-based controllers for CSMs are not fully exploiting the capabilities of recent ML algorithms. Increasing the learning ability in complex tasks is the main challenge for roboticists in the field of soft robotics. Furthermore, RL may represent a valuable solution to derive the control policy but it also requires a substantial number of trials and potentially large amounts of data on real CSM platforms. This can be overcome through the effective use of simulations to generate an initial controller solution and then further optimize on the real robot to compensate for the sim-to-real gap. Another solution can be represented by IL, to avoid self-learning issues, but it should be expected that the soft robot's kinematic and dynamic models will change when interacting with another object/body. Additionally, the effectiveness of interaction data collection can hinder the learning capabilities. It is important to consider that due to their material properties, the behavior of these soft robots can change overtime. For this reason, it may help to train multiple policies for a single task. Of course, the goal in that case would be to rely on the already learned policies to adapt to behavior changes, avoiding relearning from scratch. For this reason, the introduction of CL in soft robots represents an essential feature to help bring soft robots out of the laboratory environment and into the real world to perform useful tasks.

Acknowledgements

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the PROBOSCIS Project under Grant Agreement 863212 and SHARE-CP (project n. 300/22 FONDAZIONE PISA - RICERCA SCIENTIFICA E TECNOLOGICA – CUP J83C24000530007).

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Egidio Falotico: Conceptualization (lead); Funding Acquisition (lead); Supervision (lead); and Writing—Original Draft (lead). **Enrico Donato:** Conceptualization (equal); Writing—Original Draft (equal). **Carlo Alessi:** Conceptualization (equal) and Writing—Original Draft (equal). **Elisa Setti:** Conceptualization (supporting); Writing—Original Draft (equal). **Muhammad Sunny Nazeer:** Conceptualization (supporting) and Writing—Original Draft (equal). **Camilla Agabiti:** Conceptualization (supporting) and Writing—Original Draft (equal). **Daniele Caradonna:** Conceptualization (supporting) and Writing—Original Draft (equal). **Diego Bianchi:** Conceptualization (supporting) and Writing—Original Draft (equal). **Francesco Piqué:** Writing—Original Draft (supporting). **Yasmin Tauqeer Ansari:** Conceptualization (supporting) and Writing—Review and Editing (supporting). **Marc Killpack:** Conceptualization (lead); Funding Acquisition (lead); Supervision (lead); and Writing—Original Draft (lead).

Keywords

controls, learnings, modelings, soft robots

Received: May 1, 2024
Revised: August 29, 2024
Published online:

- [1] S. Kim, C. Laschi, B. Trimmer, *Trends Biotechnol.* **2013**, 31, 287.
- [2] D. Rus, M. T. Tolley, *Nature* **2015**, 521, 467.
- [3] D. Trivedi, C. D. Rahn, W. M. Kier, I. D. Walker, *Appl. Bionics Biomech.* **2008**, 5, 99.
- [4] J. Wang, A. Chortos, *Adv. Intell. Syst.* **2022**, 4, 2100165.
- [5] G. M. Whitesides, *Angew. Chem., Int. Ed.* **2018**, 57, 4258.
- [6] T. G. Thuruthel, Y. Ansari, E. Falotico, C. Laschi, *Soft Rob.* **2018**, 5, 149.
- [7] P. Polygerinos, N. Correll, S. A. Morin, B. Mosadegh, C. D. Onal, K. Petersen, M. Cianchetti, M. T. Tolley, R. F. Shepherd, *Adv. Eng. Mater.* **2017**, 19, 1700016.
- [8] C. D. Santina, C. Duriez, D. Rus, *IEEE Control Syst. Mag.* **2023**, 43, 30.
- [9] C. Laschi, T. G. Thuruthel, F. Lida, R. Merzouki, E. Falotico, *IEEE Control Syst. Mag.* **2023**, 43, 100.
- [10] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, F. Renda, *IEEE Trans. Rob.* **2023**, 39, 1728.
- [11] L. Marechal, P. Bolland, L. Lindenroth, F. Petrou, C. Kontovounisios, F. Bello, *Soft Rob.* **2020**, 8, 284.
- [12] M. Gazzola, L. H. Dudte, A. G. McCormick, L. Mahadevan, *R. Soc. Open Sci.* **2018**, 5, 171628.
- [13] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed., Springer Publishing Company, Incorporated, New York, NY **2008**.
- [14] K. Chin, T. Hellebrekers, C. Majidi, *Adv. Intell. Syst.* **2020**, 2, 1900171.

- [15] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, E. Falotico, *IEEE Rob. Autom. Lett.* **2022**, 7, 4741.
- [16] F. Piqué, H. T. Kalidindi, L. Fruzzetti, C. Laschi, A. Menciassi, E. Falotico, *IEEE Rob. Autom. Lett.* **2022**, 7, 5469.
- [17] S. Terrile, A. López, A. Barrientos, *Biomimetics* **2023**, 8, 56.
- [18] D. Bianchi, M. G. Antonelli, C. Laschi, A. Maria Sabatini, E. Falotico, *IEEE Rob. Autom. Mag.* **2023**, 2.
- [19] R. J. Webster, B. A. Jones, *Int. J. Rob. Res.* **2010**, 29, 1661.
- [20] B. A. Jones, I. D. Walker, *IEEE Trans. Rob.* **2006**, 22, 43.
- [21] C. D. Santina, R. K. Katzschmann, A. Bicchi, D. Rus, *Int. J. Rob. Res.* **2020**, 39, 490.
- [22] T. Mahl, A. Hildebrandt, O. Sawodny, *IEEE Trans. Rob.* **2014**, 30, 935.
- [23] I. Singh, Y. Amara, A. Melingui, P. M. Pathak, R. Merzouki, *Soft Rob.* **2018**, 5, 425.
- [24] M. Wiese, K. Rustmann, A. Raatz, in *2019 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ, November **2019**.
- [25] C. D. Santina, D. Rus, *IEEE Rob. Autom. Lett.* **2020**, 5, 290.
- [26] M. A. Graule, C. B. Teeple, T. P. McCarthy, G. R. Kim, R. C. St. Louis, R. J. Wood, in *2021 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2021**, pp. 3934–3941.
- [27] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, C. Laschi, *IEEE Trans. Rob.* **2014**, 30, 1109.
- [28] J. Till, V. Aloï, C. Rucker, *Int. J. Rob. Res.* **2019**, 38, 723.
- [29] R. M. Murray, S. Shankar Sastry, L. Xexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st ed., CRC Press, Inc., Boca Raton, FL **1994**.
- [30] F. Renda, C. Armanini, V. Lebastard, F. Candelier, F. Boyer, *IEEE Rob. Autom. Lett.* **2020**, 5, 4006.
- [31] F. Boyer, V. Lebastard, F. Candelier, F. Renda, *IEEE Trans. Rob.* **2021**, 37, 847.
- [32] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, M. Gazzola, *IEEE Rob. Autom. Lett.* **2021**, 6, 3389.
- [33] A. T. Mathew, I. B. Hmida, C. Armanini, F. Boyer, F. Renda, *IEEE Rob. Autom. Mag.* **2023**, 30, 106.
- [34] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, C. Duriez, in *2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ, May **2015**.
- [35] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, S. Cotin, in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Springer Berlin, Heidelberg **2012**, p. 283.
- [36] L. Arleo, G. Stano, G. Percoco, M. Cianchetti, *Prog. Addit. Manuf.* **2021**, 6, 243.
- [37] N. El-Atab, R. B. Mishra, F. Al-Modaf, L. Joharji, A. A. Alsharif, H. Alamoudi, M. Diaz, N. Qaiser, M. M. Hussain, *Adv. Intell. Syst.* **2020**, 2, 2000128.
- [38] M. K. Mishra, A. K. Samantaray, G. Chakraborty, *Mech. Mach. Theory* **2022**, 173, 104841.
- [39] M. Russo, S. Wild, X. Dong, D. Axinte, *IEEE/ASME Trans. Mechatron.* **2024**.
- [40] F. Renda, M. Cianchetti, H. Abidi, J. Dias, L. Seneviratne, *J. Mech. Rob.* **2017**, 9, 041012.
- [41] M. S. Xavier, A. J. Fleming, Y. K. Yong, *Adv. Intell. Syst.* **2021**, 3, 2000187.
- [42] J. Jung, R. S. Penning, N. J. Ferrier, M. R. Zinn, in *2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2011**, pp. 5139–5146.
- [43] C. Duriez, E. Coevoet, F. Largilliere, T. Morales-Bieze, Z. Zhang, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, in *2016 IEEE Int. Conf. Simulation, Modeling, and Programming for Autonomous Robots (SIMPAP)*, IEEE, Piscataway, NJ **2016**, pp. 111–118.
- [44] S. Bhagat, H. Banerjee, Z. Tse, *Robotics* **2019**, 8, 4.
- [45] X. Zhang, F. K. Chan, T. Parthasarathy, M. Gazzola, *Nat. Commun.* **2019**, 10, 4825.
- [46] C. Alessi, E. Falotico, A. Lucantonio, *IEEE Access* **2023**, 11, 37840.
- [47] R. Morimoto, S. Nishikawa, R. Niiyama, Y. Kuniyoshi, in *IEEE-RAS RoboSoft*, IEEE, Piscataway, NJ **2021**.
- [48] R. Morimoto, M. Ikeda, R. Niiyama, Y. Kuniyoshi, *Front. Rob. AI* **2022**, 9, 9.
- [49] E. Todorov, T. Erez, Y. Tassa, in *2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2012**, pp. 5026–5033.
- [50] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, S. Jo, *Plos One* **2021**, 16, 0246102.
- [51] M. S. Nazeer, D. Bianchi, G. Campinoti, C. Laschi, E. Falotico, in *2023 IEEE Int. Conf. Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2023**, pp. 1–7.
- [52] M. S. Nazeer, C. Laschi, E. Falotico, *IEEE Trans. Rob.* **2024**, 40, 2498.
- [53] T. G. Thuruthel, B. Shih, C. Laschi, M. T. Tolley, *Sci. Rob.* **2019**, 4, eaav1488.
- [54] M. Giorelli, F. Renda, G. Ferri, C. Laschi, in *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, IEEE, Piscataway, NJ, November **2013**, pp. 5033–5039.
- [55] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, R. Merzouki, *IEEE/ASME Trans. Mechatron.* **2015**, 20, 2862.
- [56] T. G. Thuruthel, E. Falotico, F. Renda, C. Laschi, *IEEE Trans. Rob.* **2019**, 35, 124.
- [57] T. G. Thuruthel, E. Falotico, F. Renda, C. Laschi, *Bioinspiration Biomimetics* **2017**, 12, 066003.
- [58] Z. Tang, P. Wang, W. Xin, C. Laschi, *IEEE Rob. Autom. Lett.* **2022**, 7, 8315.
- [59] Z. Tang, P. Wang, W. Xin, Z. Xie, L. Kan, M. Mohanakrishnan, C. Laschi, in *2023 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2023**, pp. 982–988.
- [60] Z. Tang, W. Xin, P. Wang, C. Laschi, *Soft Rob.* **2024**, 11, 767.
- [61] M. S. Malekzadeh, S. Calinon, D. Bruno, D. G. Caldwell, *Rob. Biomimetics* **2014**, 1, 13.
- [62] Y. Ansari, M. Manti, E. Falotico, M. Cianchetti, C. Laschi, *IEEE Rob. Autom. Lett.* **2018**, 3, 108.
- [63] U. Berdica, M. Jackson, N. E. Veronese, J. Foerster, P. Maiolino, in *2024 IEEE 7th Int. Conf. Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2024**, pp. 933–939.
- [64] Q. Wu, Y. Gu, Y. Li, B. Zhang, S. A. Chepinskiy, J. Wang, A. A. Zhilenkov, A. Y. Krasnov, S. Chernyi, *Information* **2020**, 11, 310.
- [65] M. Rolf, J. J. Steil, *IEEE Trans. Neural Networks Learn. Syst.* **2014**, 25, 1147.
- [66] O. Lakhal, A. Melingui, R. Merzouki, *IEEE/ASME Trans. Mechatron.* **2016**, 21, 1326.
- [67] P. Li, G. Wang, H. Jiang, Y. Jin, Y. Gan, X. Chen, J. Ji, in *2021 IEEE Int. Conf. Robotics and Biomimetics (ROBIO)*, IEEE, Piscataway, NJ **2021**, pp. 839–845.
- [68] X. You, Y. Zhang, X. Chen, X. Liu, Z. Wang, H. Jiang, X. Chen, in *2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ, September **2017**, pp. 2909–2915.
- [69] Y. Gan, P. Li, H. Jiang, G. Wang, Y. Jin, X. Chen, J. Ji, *IEEE Rob. Autom. Lett.* **2022**, 7, 12006.
- [70] G. Lou, C. Wang, Z. Xu, J. Liang, Y. Zhou, *IEEE Rob. Autom. Lett.* **2024**, 9, 7070.
- [71] Y. Li, X. Wang, K.-W. Kwok, in *2022 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2022**, pp. 7074–7081.
- [72] M. A. Graule, T. P. McCarthy, C. B. Teeple, J. Werfel, R. J. Wood, *IEEE Rob. Autom. Lett.* **2022**, 7, 4071.

- [73] S. Abondance, C. B. Teeple, R. J. Wood, *IEEE Rob. Autom. Lett.* **2020**, 5, 5502.
- [74] R. Jitoshio, T. G. Wei Lum, A. M. Okamura, C. K. Liu, in *Conf. Robot Learning* **2023**.
- [75] S. Satheshbabu, N. K. Uppalapati, G. Chowdhary, G. Krishnan, in *2019 Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ 2019, pp. 5133–5139.
- [76] S. Satheshbabu, N. K. Uppalapati, T. Fu, G. Krishnan, in *2020 3rd IEEE Int. Conf. Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ, May 2020, pp. 497–503.
- [77] N. K. Uppalapati, B. Walt, A. J. Havens, A. Mahdian, G. Chowdhary, G. Krishnan, *Robotics: Science and Systems*, MIT Press, Cambridge, MA **2020**, p. 95.
- [78] C.-H. Shih, N. Naughton, U. Halder, H.-S. Chang, S. H. Kim, R. Gillette, P. G. Mehta, M. Gazzola, *Adv. Intell. Syst.* **2023**, 5, 2300088.
- [79] C. Alessi, H. Hauser, A. Lucantonio, E. Falotico, in *2023 IEEE Int. Conf. Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2023**, pp. 1–7.
- [80] C. Alessi, D. Bianchi, G. Stano, M. Cianchetti, E. Falotico, *Adv. Intell. Syst.* **2024**, 6, 2300899.
- [81] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, C. Duriez, *Soft Rob.* **2022**, 10, 410.
- [82] C. Agabiti, E. Ménager, E. Falotico, in *2023 IEEE Int. Conf. Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2023**, pp. 1–6.
- [83] E. Ménager, C. Duriez, *IEEE 7th Int. Conf. on Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2024**, pp. 263–269.
- [84] E. Ménager, Q. Peyron, C. Duriez, *IEEE Rob. Autom. Lett.* **2023**, 8, 8478.
- [85] G. Tiboni, A. Protopapa, T. Tommasi, G. Averta, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* **2023**, pp. 612–619, <https://doi.org/10.1109/iros55552.2023.10342537>.
- [86] J. Zhang, X. Chen, P. Stegano, M. Zhou, Z. Yuan, *IEEE Rob. Autom. Lett.* **2023**, 8.
- [87] O. M. Andrychowicz, B. Baker, M. Chocie, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, *Int. J. Rob. Res.* **2020**, 39, 3.
- [88] M. Stölzle, C. D. Santina, *IEEE Control Syst. Lett.* **2021**, 6, 1837.
- [89] M. Azizkhani, I. S. Godage, Y. Chen, *IEEE Rob. Autom. Lett.* **2022**, 7, 3584.
- [90] J. Liu, P. Borja, C. D. Santina, *Adv. Intell. Syst.* **2023**, 2300385.
- [91] T. Yoon, Y. Chai, Y. Jang, H. Lee, J. Kim, J. Kwon, J. Kim, S. Choi, *IEEE Rob. Autom. Lett.* **2024**, 9, 3068.
- [92] C. C. Johnson, T. Quackenbush, T. Sorensen, *Front. Rob. AI* **2021**, 8, 654398.
- [93] M. Bartholdt, M. Wiese, M. Schappeler, S. Spindeldreier, A. Raatz, in *2021 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2021**, pp. 624–631.
- [94] R. Saegusa, G. Metta, G. Sandini, S. Sakka, in *2023 IEEE ROBIO*, IEEE, Piscataway, NJ **2008**.
- [95] R. Bajcsy, Y. Aloimonos, J. K. Tsotsos, *Auton. Rob.* **2018**, 42, 177.
- [96] A. T. Taylor, T. A. Berrueta, T. D. Murphey, *Mechatronics* **2021**, 77.
- [97] W. Zhao, J. P. Queralta, T. Westerlund, in *2020 IEEE Symp. Series on Computational Intelligence (SSCI)*, IEEE, Piscataway, NJ **2020**, pp. 737–744.
- [98] Z. Ding, N. F. Lepora, E. Johns, in *2020 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2020**, pp. 1639–1645.
- [99] R. Lal, R. Swaminathan, L. Seenivasan, L. Qiu, H. Ren, in *2021 IEEE Int. Conf. Development and Learning (ICDL)*, IEEE, Piscataway, NJ **2021**, pp. 1–6.
- [100] A. Hussein, M. M. Gaber, E. Elyan, C. Jayne, *ACM Comput. Surv.* **2017**, 50, 1.
- [101] M. S. Nazeer, C. Laschi, E. Falotico, *Sensors* **2023**, 23, 8278.
- [102] P. Oikonomou, A. Dometios, M. Khamassi, C. S. Tzafestas, in *2021 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2021**, pp. 1716–1723.
- [103] H. Zhang, R. Cao, S. Zilberstein, F. Wu, X. Chen, in *Intelligent Robotics and Applications, Lecture Notes in Computer Science* (Eds: Y. Huang, H. Wu, H. Liu, Z. Yin), Springer International Publishing, Cham **2017**, pp. 173–184.
- [104] D. Bianchi, M. Antonelli, C. Laschi, A. Sabatini, E. Falotico, in *Proc. 20th Int. Conf. Informatics in Control, Automation and Robotics - Volume 1: ICINCO, INSTICC, SciTePress, Setúbal, Portugal* **2023**, pp. 424–432.
- [105] Z. Chen, X. Ren, M. Bernabei, V. Mainardi, G. Ciuti, C. Stefanini, *IEEE Rob. Autom. Lett.* **2023**, 9, 875.
- [106] P. Oikonomou, A. Dometios, M. Khamassi, C. S. Tzafestas, in *2022 Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2022**, pp. 5212–5218.
- [107] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, C. Laschi, in *2016 6th IEEE Int. Conf. Biomedical Robotics and Biomechanics (BioRob)*, IEEE, Piscataway, NJ **2016**, pp. 833–838.
- [108] M. Cianchetti, T. Ranzani, G. Gerboni, T. Nanayakkara, K. Althoefer, P. Dasgupta, A. Mencias, *Soft Rob.* **2014**, 1, 122.
- [109] M. McCandless, F. J. Wise, S. Russo, in *2023 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2023**, pp. 573–580.
- [110] G. J. McDonald, B. Hamlen, E. Detournay, T. M. Kowalewski, *IEEE Trans. Rob.* **2023**, 39, 2400.
- [111] E. Coevoet, A. Escande, C. Duriez, *IEEE Rob. Autom. Lett.* **2017**, 2, 1413.
- [112] A. Dietrich, C. Ott, S. Stramigioli, *IEEE Rob. Autom. Lett.* **2016**, 1, 1, 184.
- [113] N. Hogan, *Annu. Rev. Control Rob. Auton. Syst.* **2022**, 5, 179.
- [114] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. Diaz-Rodriguez, *Inf. Fusion* **2020**, 58, 52.
- [115] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, *Neural Networks* **2019**, 113, 54.
- [116] V. Lomonaco, D. Maltoni, in *Proc. 1st Annual Conf. Robot Learning, Volume 78 of Proc. Machine Learning Research* (Eds: S. Levine, V. Vanhoucke, K. Goldberg), PMLR, November 2017, pp. 17–26.
- [117] U. Michieli, P. Zanuttigh, in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV) Workshops*, IEEE, Piscataway, NJ, October 2019.
- [118] K. Shmelkov, C. Schmid, K. Alahari, in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, IEEE, Piscataway, NJ, October 2017.
- [119] R. T. Kalifou, H. Caselles-Dupré, T. Lesort, T. Sun, N. Diaz-Rodriguez, D. Filliat, in *ICML Workshop on Multi-Task and Lifelong Learning*, Vol. 4, ICML, San Diego, CA **2019**.
- [120] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez, D. Filliat, in *Int. Conf. Learning Representations* **2019**.
- [121] L. Rupert, T. Duggan, M. D. Killpack, *Front. Rob. AI* **2021**, 8, 637301.
- [122] B. Wee Keong Ang, C. H. Yeow, *Soft Rob.* **2022**, 9, 1144.
- [123] R. L. Truby, C. D. Santina, D. Rus, *IEEE Rob. Autom. Lett.* **2020**, 5, 3299.
- [124] E. Donato, Y. Ansari, C. Laschi, E. Falotico, in *IEEE RoboSoft 2023*, IEEE, Piscataway, NJ **2023**.
- [125] Y. H. Tsai, P. P. Liang, A. Zadeh, L.-P. Morency, R. Salakhutdinov, in *Int. Conf. Representation Learning* **2019**.
- [126] Y. Li, J. Y. Zhu, R. Tedrake, A. Torralba, in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2019**.
- [127] R. P. Babadian, K. Faez, M. Amiri, E. Falotico, *Inf. Fusion* **2023**, 92, 313.
- [128] S. Duan, Q. Shi, J. Wu, *Adv. Intell. Syst.* **2022**, 4, 2200213.
- [129] R. J. Piechocki, X. Wang, M. J. Bocus, *Sci. Rep.* **2023**, 13.
- [130] R. Martin-Martin, O. Brock, *Int. J. Rob. Res.* **2022**, 41, 741.
- [131] F. Chen, F. Wang, X. Yang, Y. Dong, Q. Yong, L. Zheng, Y. Gao, H. Su, *MDPI Bioeng.* **2023**, 10, 1243.

- [132] W. Lu, J. Zhang, X. Zhao, J. Wang, J. Dang, J. *Ambient Intell. Hum. Comput.* **2017**, 8, 885.
- [133] X. Chen, Y. Li, *Found of Data*, AIMS, Springfield, MO **2023**.
- [134] M. A. Lee, B. Yi, R. Martin-Martin, S. Savarese, J. Bohg, in *IEEE IROS*, IEEE, Piscataway, NJ **2020**.
- [135] Q. Tang, J. Liang, F. Zhu, *Signal Process.* **2023**, 213.
- [136] E. Donato, E. Falotico, T. G. Thuruthel, in *IEEE RoboSoft 2024*, IEEE, Piscataway, NJ **2024**.
- [137] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, *Entropy* **2020**, 23, 18.
- [138] S. Roque, S. K. Damodaran, *Int. J. Hum.-Comput. Interact.* **2022**, 38, 1789.
- [139] X. Li, Z. Serlin, G. Yang, C. Belta, *Sci. Rob.* **2019**, 4, 2158.
- [140] S. T. Kanneganti, J. S. Pei, D. F. Hougen, in *2021 IEEE Symp. Series on Computational Intelligence (SSCI)*, IEEE, Piscataway, NJ **2021**.
- [141] F. Sado, C. K. Loo, W. S. Liew, M. Kerzel, *ACM Comput. Surv.* **2023**, 55.
- [142] E. Donato, T. G. Thuruthel, E. Falotico, in *IEEE BioRob 2024*, IEEE, Piscataway, NJ **2024**.
- [143] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, S. I. Lee, *Nat. Mach. Intell.* **2020**, 2, 56.
- [144] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA **2016**.
- [145] M. A. G. Santamarta, L. Fernandez-Becerra, D. Sobrin-Hidalgo, A. M. Guerrero-Higueras, I. Gonzalez, F. J. R. Lera, in *Hybrid Artificial Intelligent Systems*, Springer, Cham, Switzerland **2023**.
- [146] L. Li, E. Donato, V. Lomonaco, E. Falotico, in *IEEE-RAS RoboSoft 2024*, IEEE, Piscataway, NJ **2024**, <https://doi.org/10.1109/RoboSoft60065.2024.10522027>.