ADVANCED INTELLIGENT SYSTEMS Open Access
www.advintellsyst.com

Bridging High-Fidelity Simulations and Physics-Based Learning using a Surrogate Model for Soft Robot Control

*Taehwa Hong, Jungjae Lee, Byung-Hyun Song, and Yong-Lae Park**

Soft robotics holds immense promise for applications requiring adaptability and compliant interactions. However, the lack of sufficiently fast and accurate simulation environments for soft robots has hindered progress, particularly in linking with reinforcement learning (RL) applications. Traditional finite element method (FEM) models provide precise insights into soft robot dynamics but are computationally intensive and impractical for accelerated simulation. This work introduces a novel framework that integrates high-fidelity FEM simulations with computationally efficient physics-based simulations through a surrogate model tailored for RL. The surrogate model, trained on real-world and FEM-generated datasets, captures complex dynamics while maintaining efficiency. Sim2real experiments validate the framework, implementing the trajectory tracking and the force control tasks with high accuracy. These results demonstrate the framework's ability to bridge the simulation gap, enabling its application to advanced tasks, such as manipulation and interaction in unstructured environments.

1. Introduction

Advances in soft robotics have enabled the development of adaptable and compliant systems that interact safely with unstructured environments.[1–3] Made from flexible, deformable materials, soft robots excel in tasks requiring delicate manipulation and human–robot interaction. As the demand for precise control grows for soft robots, the integration of external sensors and feedback control has become standard to improve performance and safety.[4–6] However, relying solely on sensory feedback is often insufficient, necessitating robust modeling techniques to predict and control the nonlinear behaviors of soft robotic systems.[7–9]

Typical modeling methods, such as lumped parameter models,[10] Cosserat rod theories,[11] and voxelization,[12] often fail to capture the complex, hyperelastic properties of soft materials. This limitation highlights the need for accurate and accessible modeling frameworks to harness the potential of soft robots in applications such as biomedical devices,[13] human–robot interaction,[14] and precision manipulation.[15]

Models based on the finite element method (FEM) are widely used to analyze soft materials, with tools that provide multiphysics simulation and high-fidelity models.[16,17] Although effective in incorporating material properties into simulations, FEM models are computationally intensive, making them unsuitable for dynamic and real-time applications.[18,19] Their computational complexity and high resource demands further limit their practicality for modeling soft robot control.[20,21]

Recent data-driven approaches, such as deep learning[22–24] and reinforcement learning (RL),[25–28] offer promising alternatives to the learning dynamics of soft robots. RL enables agents to learn optimal policies through interactions,[29,30] and has been applied to soft robotics for tasks requiring dynamic control.[31–33] However, end-to-end methods face challenges, including damage during exploration, susceptibility to noise, and the difficulty of collecting large datasets due to the mechanical fragility of soft robots.[13,34] These limitations emphasize the need for risk-free data acquisition of soft robots to facilitate RL-based control.

Various platforms have been developed in recent years to address the challenges of modeling and controlling soft robots, particularly with respect to real-time interaction and physical realism. Among them, dynamic simulation frameworks, such as the simulation open framework architecture (SOFA), have enabled high-fidelity modeling of soft robot behaviors.[35–37] FEM-based simulations provided in the SOFA allowed physical interactions with the environment,[38,39] but they face limitations in scalability and computational efficiency. Although model order reduction (MOR) techniques improve runtime performance, they are often task-specific and computationally intensive to implement.[40,41]

In parallel, several RL environments tailored to soft robotics have been introduced[42–44] with the aim of learning control policies directly through interaction. However, achieving high-precision control, particularly in both position and force domains, remains difficult due to the complexity of soft body dynamics and limited simulation speed or accuracy. These efforts collectively

T. Hong, B.-H. Song, Y.-L. Park
Department of Mechanical Engineering, Institute of Advanced Machines and Design
Seoul National University
Seoul 08826, Republic of Korea
E-mail: ylpark@snu.ac.kr

J. Lee
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215, USA

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202500696.

DOI: 10.1002/aisy.202500696

underscore the need for a unified solution that combines efficient simulation fidelity with learning-based control capabilities.

This work introduces a framework that embeds soft robotic models in a physics-based simulation environment compatible with RL. The framework bridges FEM models and efficient physics-based simulations with a computationally efficient alternative model. It addresses three key objectives: 1) transferring high-fidelity FEM data into fast, physics-based simulations while preserving essential dynamics, 2) enabling accelerated simulations and data acquisitions for the RL environment, and 3) validating the framework by analyzing the model accuracy in each simulation domain and sim2real transfer tasks. Beyond improving control and simulation efficiency, this framework lays the foundation for data-driven learning in soft robotics by enabling rapid iteration and safe policy exploration. It opens new possibilities for using high-resolution physics-informed simulations to accelerate RL across a range of tasks, including manipulation, interaction, and compliant control in real-world environments.

2. Framework Overview

A three-domain framework was developed to bridge real hardware, high-fidelity FEM simulation, and accelerated physics-based RL environments. An overview of the simulation-to-learning pipeline is presented in **Figure 1**. A pneumatic soft manipulator[45] was selected as the representative system due to its high degrees of freedom and nonlinear dynamic characteristics.[46,47] This platform provides a structured benchmark for evaluating control-oriented simulation strategies.

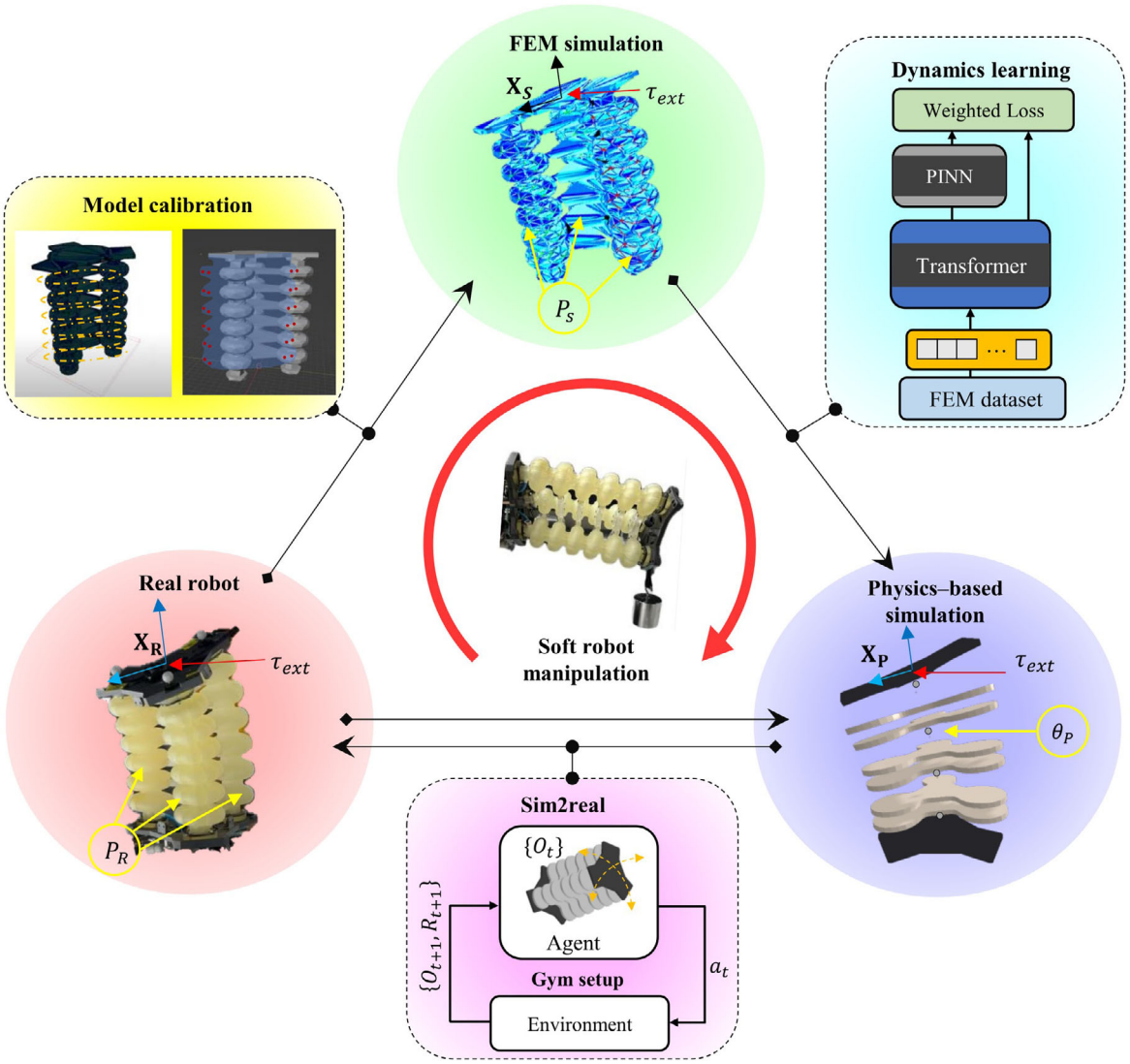

Figure 1. Overview of the proposed framework connecting three different domains ("Real robot", "FEM simulation", and "Physics-based simulation") representing a common soft manipulator. The soft manipulator is actuated by the pressure input (P), and the output includes the state positions ($\mathbf{X}$) and the velocities ($\dot{\mathbf{X}}$). The input and the output are marked with the subscripts R, S, and P, respectively. Between each domain, connections were developed utilizing simulations and methodologies, such as the model calibration, the feature mapping, and the sim2real transfer learning.

The high-fidelity model was constructed in SOFA, incorporating nonlinear material behavior and pressure-driven actuation. This model enabled accurate prediction of the manipulator's response under varying pressure inputs while avoiding hardware degradation. To reduce computational overhead, MOR techniques[40,41,48] were applied, enabling efficient generation of time-series data for forward and inverse kinematics learning.

Despite its precision, SOFA-based simulation remains computationally intensive and therefore unsuitable for RL, which demands large volumes of agent–environment interactions.[42,43] To address this, a surrogate model was implemented in PyBullet, selected for its real-time performance, robust collision handling, and integration with established learning libraries. The surrogate model comprises a combination of revolute and prismatic joints arranged to approximate the deformation patterns observed in the FEM simulation, allowing for efficient policy training and transfer.

The primary contribution lies in the integration of three distinct domains: real hardware, FEM-based simulation, and physics-driven surrogate environments, through a set of data-driven mapping functions. The mappings between each domain ensure dynamic consistency and enable bidirectional transfer between domains.

3. Model Calibration

3.1. FEM Model Construction

The calibration process begins by replicating the real robot configuration in the SOFA FEM environment. The physical system consists of a bellow-type parallel manipulator actuated via three independent pressure regulators, mounted on a fixed frame with an actuated top plate, as shown in **Figure 2**a. This setup was reconstructed in SOFA using a volumetric tetrahedron mesh with three internal cavity meshes for actuation, as shown in Figure 2b.

To align the simulation with the hardware, the state variables of the model were set as shown in Figure 2c. The state variables consist of the position of the tool center point (TCP) ($p \in \mathbb{R}^3$), the length of the backbone of the manipulator ($l \in \mathbb{R}^1$), curvature ($\kappa \in \mathbb{R}^1$), and the relative orientation of the top plate ($q \in \mathbb{R}^3$). Together, they form a configuration state vector ($\mathbf{X} \in \mathbb{R}^{N_s}$) and its velocity counterpart ($\dot{\mathbf{X}} \in \mathbb{R}^{N_s}$), where N_s is the total number of the total dimensions of the state variables. Material properties such as Young's modulus (E) and Poisson's ratio (ν), as well as the parameters to determine the hyperelastic governing equation of the robot, were optimized to match two domains (Supporting Information Section 2).

3.2. Domain Mapping Functions

The pressurized cavities were simulated in the SOFA environment using boundary conditions and quadratic program (QP) solvers.[39] Discrepancies between real-world pressures (P_R) and simulated pressures (P_s) were corrected by using a multilayer perceptron (MLP)

$$P_S = f_{RS}(P_R) \tag{1}$$

which calibrates the input pressures required for consistent TCP positioning. The network maps measured inputs—pressures

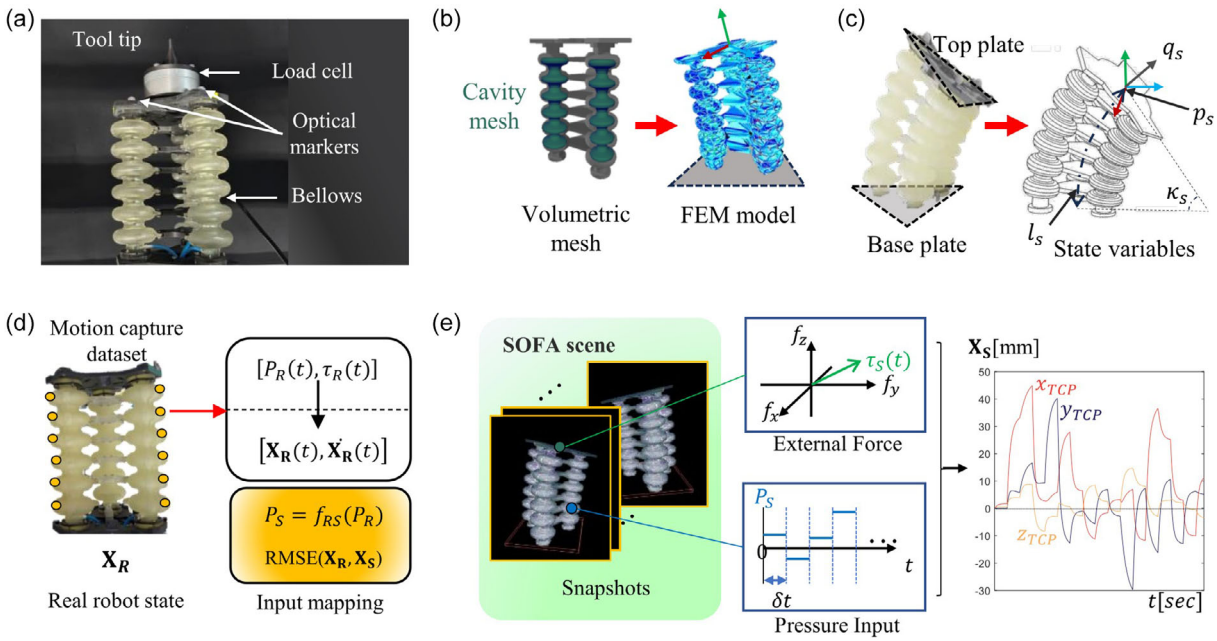

Figure 2. a) Components of the soft manipulator in the real-world setup, including the bellows-integrated manipulator. b) FEM model of the soft manipulator in the SOFA simulation scene, utilizing a mesh model that replicates the physical structure. c) Configuration state information (**X**) of the soft manipulator, which is necessary to define the behavior of the manipulator. d) Input pressure mapping and calibration process between the real robot and the FEM model, achieved using motion capture data ($\mathbf{X_R}$) to align the dynamics between the two domains. e) Snapshot collection from the SOFA simulation, where the three bellows were actuated with varied pressures (P_S) and external forces (τ_S). The right plot presents the position data ($\mathbf{X_S}$) collected over the simulations.

($P_S \in \mathbb{R}^3$) and external forces at the TCP ($\tau \in \mathbb{R}^3$) to the estimated dynamic states ($\mathbf{X}$, $\dot{\mathbf{X}}$), as shown in Figure 2d. The detailed information is provided in Supporting Information Section 4.

3.3. MOR

To enable efficient simulation while preserving the deformation characteristics of the soft manipulator, MOR was applied using proper orthogonal decomposition (POD). The high-dimensional nodal state of the FEM model at each time step was denoted as $\mathbf{p}(t) \in \mathbb{R}^{3N}$, where N is the number of mesh nodes and each entry represents 3D displacement coordinates. A snapshot matrix $Sn \in \mathbb{R}^{3N \times M}$ was constructed by collecting M time-series samples

$$Sn = [\mathbf{p}(t_1),\ \mathbf{p}(t_2),\ \ldots,\ \mathbf{p}(t_M)] \tag{2}$$

Singular value decomposition was applied to extract the dominant spatial deformation modes

$$\mathbf{S}_r = U\Sigma V^{\mathrm{T}} \tag{3}$$

where U contains orthonormal spatial modes, Σ is a diagonal matrix of singular values, and V encodes the temporal evolution. A reduced basis $\Phi \in \mathbb{R}^{3N \times r}$ was defined by selecting the first r dominant columns of U. The high-dimensional nodal state was then approximated by projecting onto this subspace

$$\mathbf{p}(t) \approx \Phi \mathbf{a}(t) \tag{4}$$

where $\mathbf{a}(t) \in \mathbb{R}^r$ denotes the generalized coordinates in the reduced space.

As shown in Figure 2e, training data were generated by applying randomized pressure inputs and external forces to the FEM model, capturing over 100 000 simulation frames. From these, key nodal regions were retained, including the TCP node ($\mathbf{X}_{\mathrm{TCP}}$) and boundary contact nodes ($\mathbf{X}_{\mathrm{b}}$), to preserve the critical interaction behavior. These reduced representations were later used to derive the low-dimensional state vector $\mathbf{X}_{\mathbf{S}}(t) \in \mathbb{R}^{N_s}$, which includes physical quantities such as position, curvature, arc length, and orientation. This reduced-order formulation significantly accelerated the simulation while maintaining sufficient accuracy for control and learning. Full implementation details are provided in Supporting Information Section 3.

4. Dynamics Modeling using Surrogate Model

4.1. Transformer-Based Physics-Informed Dynamics Modeling

The model was designed to predict the Cartesian velocity of the manipulator based on time-series joint configurations and contact forces. **Figure 3**a shows the entire structure of the dynamics model trained from the collected SOFA dataset. Each time step included an input vector

$$\mathbf{s}_t = [\mathbf{X}_{\mathbf{S}}, P_{\mathrm{S}}, \tau_{\mathrm{P}}] \in \mathbb{R}^{N_s+6} \tag{5}$$

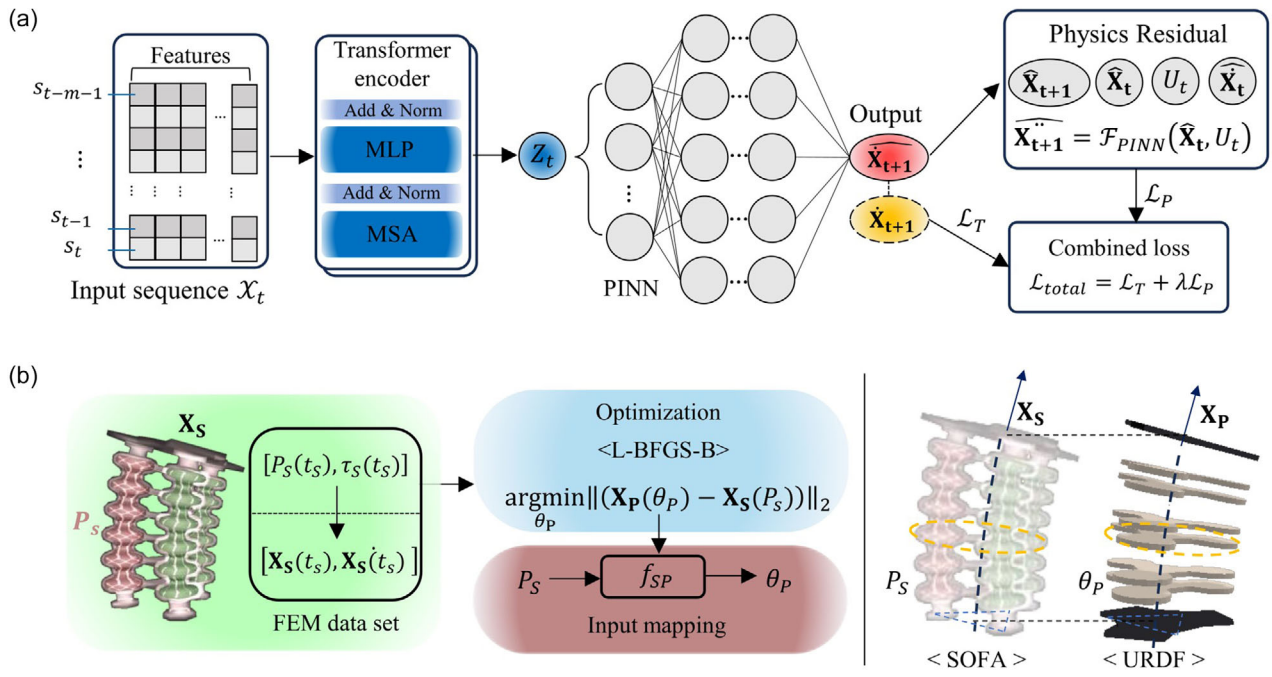

Figure 3. a) Architecture of the data-driven dynamics model combining a Transformer encoder with physics-informed residuals. The training objective integrates data loss and physical constraints into a unified loss function. b) Mapping between pressure inputs (P_S) and joint angles (θ_P) across two simulation environments. The dataset, derived from SOFA simulation snapshots, was used to optimize state alignment between the SOFA and PyBullet models. The trained forward model predicts joint angles (θ_P) from pressure inputs (P_S), minimizing the discrepancy in predicted states. The rightmost figure shows the resulting alignment of states under varied inputs in both simulation domains.

A sequence of m past steps was collected as the input sequence

$$\mathcal{X}_t = [\mathbf{s}_{t-m-1}, \ \ldots, \ \mathbf{s}_t] \in \mathbb{R}^{m\times(N_s+6)} \tag{6}$$

The sequence was linearly projected onto a high-dimensional space and processed by a Transformer encoder ($\mathcal{T}_{\text{enc}}$)[49] composed of two attention layers with four heads each. The encoded representation at the final time step was used as a latent vector $\mathbf{Z}_t$ for downstream dynamics modeling, such as

$$\mathbf{Z}_t = \mathcal{T}_{\text{enc}}(\mathcal{X}_t) \tag{7}$$

The latent vector was passed to a physics-informed neural network (PINN),[50] which predicted the Cartesian velocity of the manipulator

$$\hat{\dot{\mathbf{X}}}_{\mathbf{S}}(t+1) = f_{\text{PINN}}(\mathbf{z}_t) \tag{8}$$

The physics residual was computed using a constrained multibody dynamics model[40,41] as

$$\mathfrak{R}_t = \mathbb{M}(\hat{\mathbf{X}}_{\mathbf{S}}(t))\ddot{\hat{\mathbf{X}}}_{\mathbf{S}}(t) - \mathbb{P}_S(t) + \mathbb{F}(\hat{\mathbf{X}}_{\mathbf{S}}(t), \hat{\dot{\mathbf{X}}}_{\mathbf{S}}(t)) + \mathbb{H}^{\mathrm{T}}\lambda(t) \tag{9}$$

Here, $\mathbb{M}$ denotes the mass matrix derived from the FEM model, $\mathbb{P}_S(t)$ is the pressure-induced force vector, $\mathbb{F}$ represents internal elastic and damping forces, and $\mathbb{H}^{\mathrm{T}}\lambda(t)$ captures constraint forces from fixed supports or contact boundaries. The total loss was defined as

$$\mathscr{L}_{\text{total}} = \underbrace{\|\hat{\dot{\mathbf{X}}}_{\mathbf{S}}(t+1) - \dot{\mathbf{X}}_{\mathbf{S}}(t+1)\|^2}_{\mathscr{L}_{\text{T}}} + \lambda \cdot \underbrace{\|\mathfrak{R}_t\|^2}_{\mathscr{L}_{\text{P}}} \tag{10}$$

The loss weight λ was manually selected to balance accuracy and physical consistency.[51] All ground truth velocities $\dot{\mathbf{X}}_{\mathbf{S}}(t+1)$ were obtained from the SOFA simulation dataset. This results in a learned dynamics function of the form

$$\hat{\dot{\mathbf{X}}}_{\mathbf{S}}(t+1) = f_{\text{S}}(\chi_t) \tag{11}$$

where f_s denotes the combined Transformer and PINN model that maps the input sequence $\mathcal{X}_t$ to the predicted Cartesian velocity while preserving physical consistency. Futher derivations are in Supporting Information section 6.

4.2. Surrogate Model for Physics-Based Simulation

A data-driven surrogate model was developed to approximate the deformation behavior of the soft manipulator and enable accelerated simulation. The model replicates the evolution of the configuration states $\mathbf{X}_{\mathbf{S}}$ under pneumatic actuation, including the motion of the TCP, and was calibrated to match the range of motion (ROM) observed in the high-fidelity SOFA simulation.

The surrogate design was guided by node-level analysis of the reduced-order FEM mesh. Boundary nodes ($\mathbf{X}_{\text{b}}$) were extracted from six bellows segments, each sampled from the outermost regions of six longitudinal layers to define the radial boundaries. As shown in **Figure 4**a, these nodes form the collision boundary

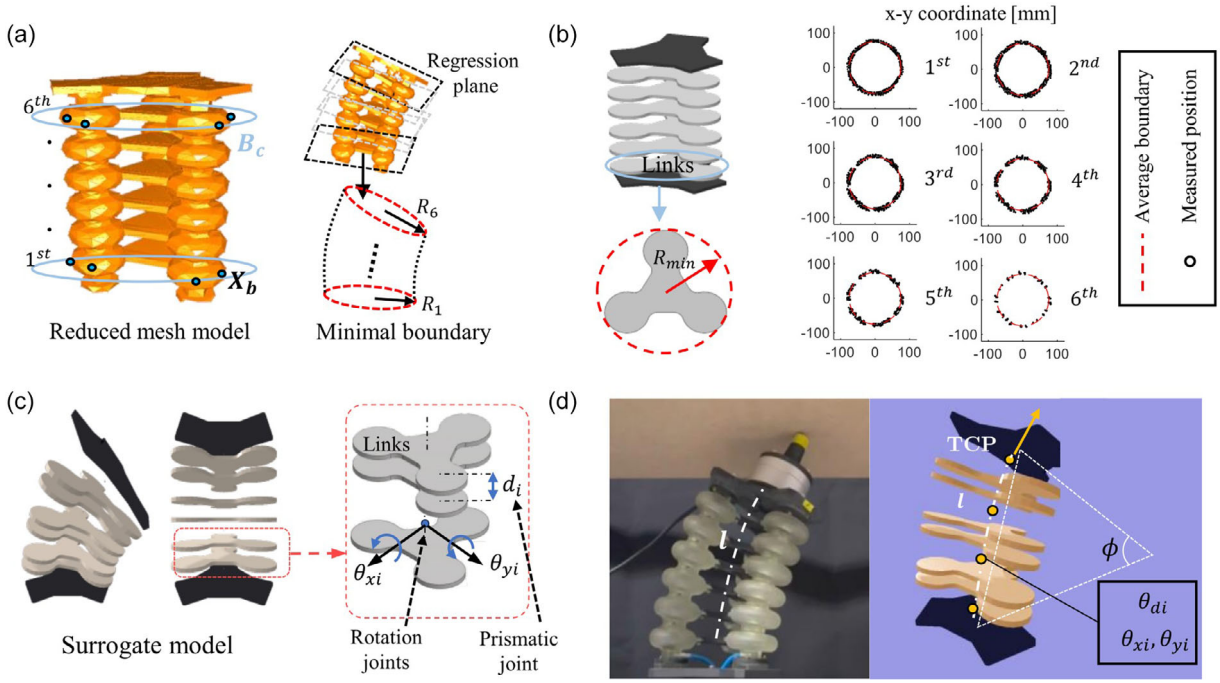

Figure 4. a) Reduced FEM mesh with layer-wise boundary nodes ($\mathbf{X}_{\text{b}}$) marked in blue. These nodes define the potential collision boundary (B_c) of each segment and are preserved during MOR to ensure contact fidelity. b) Contact boundaries computed by projecting $\mathbf{X}_{\text{b}}$ onto regression planes fitted to each layer. Red dashed circles denote the average radial boundary R_i, and the minimal boundary radius R_{min} was selected as a design constraint for the surrogate model. c) Structure of the surrogate model composed of planar links connected by rotational joints ($\theta_{x_i}, \theta_{y_i}$) and prismatic joints ($\theta_{d_i}$). d) Configuration matching between the surrogate model and the real robot, illustrating consistent deformation behavior and parameter correspondence between the two domains.

(B_c) of the deformable body. Their positions were projected onto regression planes to characterize radial deformation, yielding average contact radii by layer (R_i), where $i = 1, \ldots, 6$. The smallest of these values, $R_{\min}$, was used as a conservative constraint in the surrogate geometry, illustrated by the red dashed circles in Figure 4b.

Based on this geometric characterization, the surrogate model comprises six identical planar links stacked vertically, as shown in Figure 4c. Each pair of adjacent links is connected via two rotational joints ($\theta_{x_i}, \theta_{y_i}$) and one prismatic joint (θ_{d_i}), enabling omnidirectional bending and extension. The overall height, diameter, and thickness of the link were derived from the length of the arc (l) and the contact boundaries. The detailed design parameters are provided in Supporting Information section 7 and section 8.

The model was implemented in universal robot description format with explicit definitions of mass distribution, joint limits, and link dimensions, reflecting both ROM and self-contact constraints from FEM. As illustrated in Figure 4d, the surrogate demonstrates dynamic consistency with the reference configuration space of the real robot. This formulation preserves essential deformation and contact characteristics while offering a reduced computational cost, making it well-suited for RL and online control tasks.

4.3. Mapping between Actuation Domains

To enable simulation-to-simulation transfer between the FEM model and the surrogate model, a data-driven mapping was developed to align their respective actuation spaces. In the FEM domain, the robot is actuated by pressures ($P_S \in \mathbb{R}^3$), while the surrogate model operates in a joint space using grouped joint commands ($\boldsymbol{\theta}_P \in \mathbb{R}^{N_a}$), where N_a is the DOF of the surrogate model. A forward mapping from pressure to joint configuration was established to ensure that both models generate consistent motions ($\mathbf{X_S} \approx \mathbf{X_P}$), as illustrated in Figure 3b.

The mapping process first involved a constrained optimization, in which the joint angles θ_P were iteratively adjusted to minimize the Euclidean distance between the TCP position of the surrogate model $\mathbf{X_P}(\boldsymbol{\theta}_P)$ and the reference TCP position $\mathbf{X_s}(P_S)$ obtained from the FEM simulation. The optimization problem was formulated as

$$\boldsymbol{\theta}_P^* = \arg\min_{\boldsymbol{\theta}_P} \|\mathbf{X}_P(\boldsymbol{\theta}_P) - \mathbf{X}_S(P_S)\|_2 \tag{12}$$

subject to joint limits

$$L_\theta \leq \boldsymbol{\theta}_P \leq U_\theta \tag{13}$$

This problem was solved using the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with box constraints (L-BFGS-B),[52] which ensures physically feasible configurations despite the redundant DOFs of the surrogate model. The optimization procedure is detailed in Algorithm 1 and in Supporting Information section 5.

Once optimized joint configurations $\boldsymbol{\theta}_P^*$ were collected across a range of pressure inputs P_S, another fully connected MLP was trained to approximate the mapping

$$\boldsymbol{\theta}_P(t) = f_{SP}(P_S(t)) \tag{14}$$

Algorithm 1. 1 L-BFGS-B Optimization Pseudo code for Actuation Space Mapping

Input:
- Target TCP position $\mathbf{X}_{\text{target}} \in \mathbb{R}^3$
- Initial guess $\boldsymbol{\theta}_0 \in \mathbb{R}^{N_a}$
- Joint limits $L_\theta, U_\theta \in \mathbb{R}^{N_a}$
- Maximum iterations $N_{\max}$
- Tolerance $\varepsilon > 0$

Output: Optimized joint angles $\boldsymbol{\theta}_P^*$

Define objective function:

$$f(\boldsymbol{\theta}) = \|\mathbf{X}_{\text{actual}}(\boldsymbol{\theta}) - \mathbf{X}_{\text{target}}(P_S)\|_2$$

Solve optimization:

$$\boldsymbol{\theta}_P^* = \arg\min_{\boldsymbol{\theta}_P} f(\boldsymbol{\theta}_P) \quad \text{subject to} \quad L_\theta \leq \boldsymbol{\theta}_P \leq U_\theta$$

Initialize: $\boldsymbol{\theta}_P \leftarrow \boldsymbol{\theta}_0$

Set iteration counter $k \leftarrow 0$

while not converged and $k < N_{\max}$ **do**

  Compute current TCP position: $\mathbf{X}_{\text{actual}} = \mathbf{X}_{\text{actual}}(\boldsymbol{\theta}_P^k)$

  Compute gradient $\nabla f(\boldsymbol{\theta}_P^k)$

  Update joint angles using L-BFGS-B:

$$\boldsymbol{\theta}_P^{k+1} = \boldsymbol{\theta}_P^k - \alpha^k \nabla f(\boldsymbol{\theta}_P^k)$$

  (where α^k is determined by line search)

  Apply joint limits:

$$\boldsymbol{\theta}_P^{k+1} = \min(\max(\boldsymbol{\theta}_P^{k+1}, L_\theta), U_\theta)$$

  if $\|\nabla f(\boldsymbol{\theta}_P^{k+1})\| < \varepsilon$ **or** $\|\boldsymbol{\theta}_P^{k+1} - \boldsymbol{\theta}_P^k\| < \varepsilon$ **then**

    break

  end if

  $k \leftarrow k + 1$

end while

return $\boldsymbol{\theta}_P^*$ and $\mathbf{X}_{\text{actual}}(\boldsymbol{\theta}_P^*)$

where F_{SP} denotes the learned mapping function. This network enables the continuous and efficient translation of pressure commands in the FEM domain into joint space control signals, enabling the surrogate model to follow the dynamics learned in Equation (9).

Figure 3b shows the mapping pipeline, including the optimization-based correspondence between models and the learned regression function. This cross-domain actuation alignment serves as a critical interface for sim-to-sim transfer and supports RL in the surrogate domain while retaining the physical fidelity of the FEM reference.

4.4. Inverse Dynamics Modeling

In contrast to forward dynamics prediction, which estimates future states from given control inputs, inverse dynamics modeling aims to infer the actuation inputs required to achieve a desired state transition. Specifically, the objective was to predict the pressure input vector $P_S(t)$ that yields a target Cartesian velocity $\dot{\mathbf{X}}_S(t+1)$, given the current state information ($\mathbf{X_S}(t)$) and the external contact force ($\tau_{\text{ext}}(t)$).

This setting is particularly relevant in trajectory tracking or position control, where the desired motion is known a priori and the corresponding actuation must be inferred. For example, given a reference trajectory $\mathbf{X}_{\text{ref}}(t)$, the system must determine the pressure inputs that produce the velocity required to follow this path. The inverse model thus functions as a control-oriented

module that maps motion intent into executable actuation commands.

Due to redundancy in actuation and the underdetermined nature of the inverse mapping, a separate neural network was trained to approximate the function

$$\hat{P}_S(t) = f_S^{\dagger}(\mathbf{X}_S(t),\ \boldsymbol{\tau}_{ext}(t),\ \dot{\mathbf{X}}_S(t\ +\ 1)) \tag{15}$$

where $f_S^{\dagger}$ is a MLP trained using data from the FEM simulation. The training objective minimizes the supervised prediction error as

$$\mathscr{L}_S^{\dagger} = \|\hat{P}_S(t) - P_S(t)\|^2 \tag{16}$$

An auxiliary loss was used by passing the predicted pressure $\hat{P}_S(t)$ through the pretrained forward model f_s defined in Equation (9) to promote consistency with system dynamics. A cycle-consistency objective was set as

$$\mathscr{L}_{cyc} = \|f_S(\mathcal{X}'_t) - \dot{\mathbf{X}}_S(t+1)\|^2 \tag{17}$$

where $\mathcal{X}'_t$ is a synthetic input sequence constructed using the predicted pressure. The total loss is defined as

$$\mathscr{L}_{total}^{\dagger} = \mathscr{L}_S^{\dagger} + \beta \cdot \mathscr{L}_{cyc} \tag{18}$$

with β as a weighting parameter. The joint angle θ_P for the surrogate model was obtained through the pressure to joint mapping f_{SP}.

5. Customized RL Environment

A customized RL environment was developed to evaluate the effectiveness of the surrogate model and the trained dynamics within standard learning pipelines. The environment was built to be fully compatible with the Gymnasium-based interfaces,[53] exposing conventional agent structures including observation (O_t), action (a_t), and reward (R_t). A modular class structure was implemented to manage the interaction logic, including actuation, contact detection, and state updates. The structure of the environment and the interaction of the agent are illustrated in **Figure 5**a.

5.1. Task Definition

Three distinct control tasks were defined to evaluate different aspects of the agent's capability using the surrogate model. Task 1 targets pure trajectory tracking in Cartesian space. The agent is required to follow a predefined reference trajectory $\mathbf{X}_{ref} \in \mathbb{R}^{N_t \times 3}$, where N_t is the number of points. Two representative shapes were used: a circular path and a star-shaped path, as shown in Figure 5d. Task 2 focuses solely on force control. The agent must apply internal joint commands to generate the desired contact force in the TCP, aligned with a target external force vector $\boldsymbol{\tau}_{ext}^*$. Last, Task 3 combines position and force objectives into a multiobjective control scenario. The agent follows a tilted elliptical trajectory while maintaining a constant normal force on a sloped contact surface $\boldsymbol{S}_{ref}$.

These task definitions were selected to evaluate the feasibility of the surrogate model under varying physical objectives, ranging from geometric accuracy to compliant contact behavior. An example of hybrid trajectory-force control on a ramp stage is visualized in Figure 5e.

5.2. Observation and Action Space

The observation vector provided to the agent includes joint and TCP-level information, summarized as

$$O_t = [\boldsymbol{\theta}_P,\ \dot{\boldsymbol{\theta}}_P,\ \mathbf{X}_P,\ \dot{\mathbf{X}}_P,\ \boldsymbol{\tau}_{ext}] \tag{19}$$

where $\boldsymbol{\theta}_P$ and $\dot{\boldsymbol{\theta}}_P$ denote joint angles and velocities, $\mathbf{X}_P$ and $\dot{\mathbf{X}}_P$ denote the position and velocity of TCP, and τ_{ext} is the external contact force vector measured at the TCP.

The action space consists of continuous joint-level actuation commands:

$$a_t = \boldsymbol{\theta}_P^{cmd} \in \mathbb{R}^{N_a} \tag{20}$$

where N_a is the number of actuated degrees of freedom in the surrogate model.

5.3. Reward Function Shaping

A composite reward function was employed to simultaneously address position tracking and force regulation

$$R = \alpha R_q + (1-\alpha) R_F \tag{21}$$

where R_q and R_F represent the normalized rewards for position and force, respectively. Both terms were scaled within [0,1] based on the ideal task performance, and the scalar $\alpha \in [0,1]$ was used to balance their contributions.

The position reward R_q was defined as a decreasing function of the norm-2 error $\delta_q = \|\mathbf{X}_P - \mathbf{X}_{ref}\|_2$ between the current and target TCP positions. To encourage sequential trajectory execution, rewards were conditioned by the current step index such that only proximity to the next target point contributed to the accumulated return. This mechanism is visualized in Figure 5b.

The force reward R_F was determined from the root mean square error δ_F between the measured contact force and the reference force vector $\boldsymbol{\tau}_{ext}^*$. The empirical shapes of R_q and R_F were selected to ensure smooth reward gradients and training stability, as illustrated in Figure 5c.

The resulting reward structure captures the multiobjective nature of the task space, balancing geometric accuracy with interaction forces. The final profile of the combined reward function is depicted in Figure 5e.

6. Experimental Section

6.1. Sim2Real Learning Framework

The experiments evaluate the proposed learning pipeline across three domains: the real soft manipulator, an FEM simulation in SOFA, and a physics-based surrogate model implemented in PyBullet. All experiments were conducted using a pneumatic soft

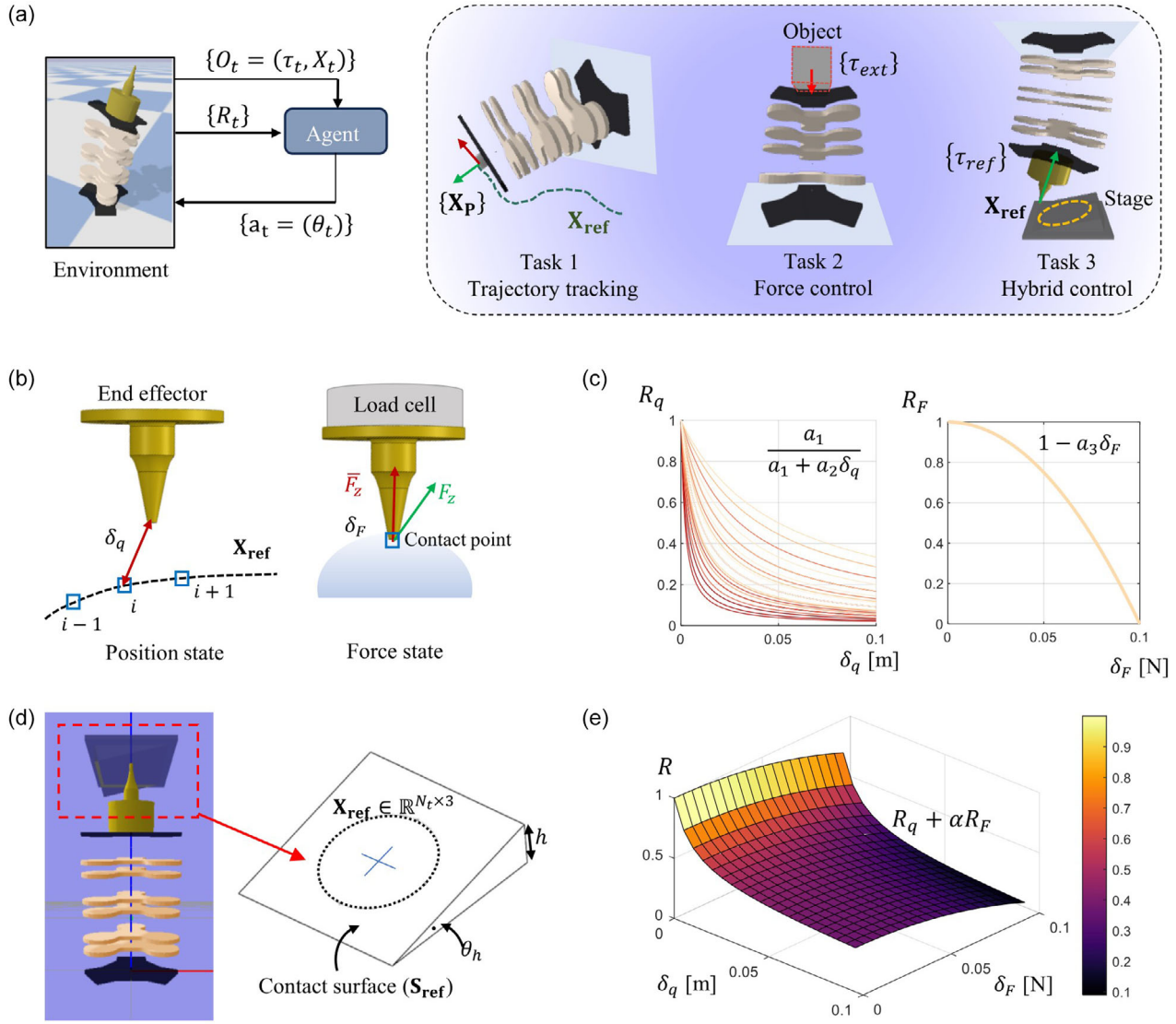

Figure 5. a) Customized RL environment setup illustrating the interaction between the agent and the environment across three tasks involving position and force controls. b) (left) Sequential position rewards conditioned on the order of trajectory points, ensuring the agent follows the path in the correct sequence. (right) The force error (δ_F) is computed with the target force vector. c) Reward function shapes for position (R_q) and force (R_F), showing their respective contributions. d) Predefined trajectory ($\mathbf{X}_{ref}$) consisting of m coordinates on the target surface ($\mathbf{S}_{ref}$). The example shown corresponds to Task 3, where the trajectory follows an elliptical path on the ramp surface. e) Manifold of the combined reward function (R), where the weighting factor (α) determines the balance between the position and the reward of the force.

manipulator configured as a parallel mechanism, where three bellows actuators are positioned between two rigid plates. The bottom plate was rigidly mounted to the base frame, and the top plate was free to move under internal pressure. This configuration was used consistently across simulation and real-world experiments. For interaction tasks, a load cell and custom tool tips were attached to the top plate to enable external force application and measurement, as shown in Figure 2a. The full experimental setup, including the optical marker placement and motion capture system, is shown in **Figure 6**a.

The actuator in the real and simulated systems was defined within a pressure range of −20 to 35 kPa, selected to ensure structural safety and consistency in the ROM. The surrogate model replicates the ROM using joint angle actuation derived from learned mappings.

The pipeline of deployment for the sim2real transfer is shown in Figure 6b. A trained policy $\pi(\theta_P(t)|O_t)$ produces joint commands, which are converted into simulated pressures using the mapping function f_{SP}, and subsequently into real robot pressures using f_{PR}. A moving average filter smooths the resulting pressure signals $P_R(t)$ before execution. The inference loop operates at over 300 Hz, while the actuation of the real system was limited to 2 Hz due to hardware constraints. The quantitative analysis of the hardware system bandwidth was conducted and reported in the Results section and Figure 9c.

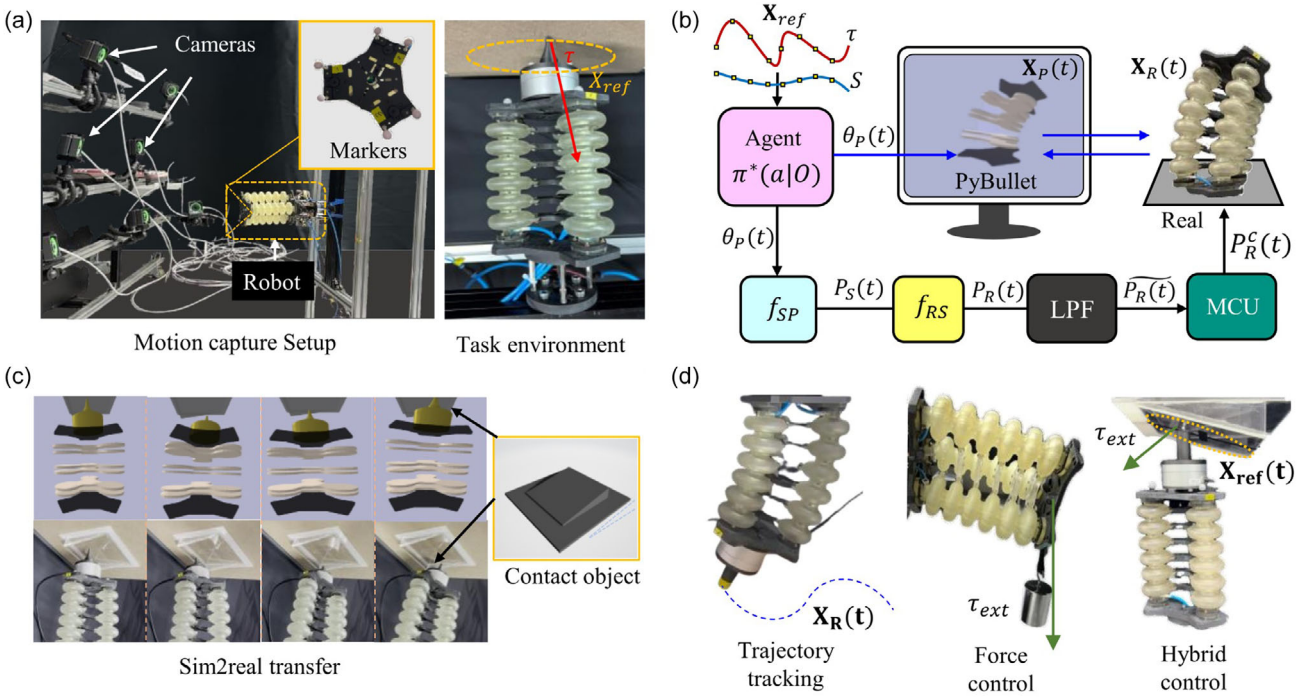

Figure 6. a) Motion capture setup used for collecting real-world robot data, incorporating the optical markers and the cameras for precise state estimation. b) Sim2real transfer pipeline illustrating a real-time mapping between the PyBullet simulation and the real robot. c) Mirrored real-world task implementation setup in the PyBullet environment, including the force measurement units and a contact object. d) Three types of Sim2real tasks: i) trajectory tracking, ii) force control, and iii) hybrid control.

6.2. Data Acquisition and Agent Training

Comprehensive data collection was conducted in each domain. In the real system, 5000 distinct pressure configurations were applied in the range −20–30 kPa per actuator (Supporting Information Section 1). Each case was held for stabilization, and the resulting motions were recorded using the motion capture system.

In the FEM domain, the pressure input (P_S) and the external contact forces (τ_{est}) were applied to a model using the same pressure range. The contact forces (τ_{est}) did not exceed 2 N, which corresponds to the maximum payload of the manipulator. The simulation was run at a time step of 0.01 s, generating outputs at 5.56 Hz. Approximately two million samples were collected and used to train the dynamics model in Equation (9).

In the PyBullet-based surrogate domain, the joint angle limits were derived from the FEM-ROM by mapping f_{SP}. The simulation of the surrogate model was sampled at over 1000 Hz, with each configuration executed over 20 control steps to efficiently generate the training dataset for RL.

A custom Gym-compatible RL environment was constructed around the surrogate model. The observation space includes joint states $(\boldsymbol{\theta}_P, \dot{\boldsymbol{\theta}}_P)$, TCP position and velocity $(\mathbf{X}_P, \dot{\mathbf{X}}_P)$, and external contact force (τ_{est}). The action space consists of continuous joint commands. Three agents were tested: soft actor-critic,[54] proximal policy optimization,[55] and behavior cloning.[56] These agents represent off-policy, on-policy, and supervised imitation learning methods, respectively, and were implemented using the stable-baselines3 library.[57]

6.3. Task Setup and Evaluation

The framework was validated on three distinct control tasks of increasing complexity, summarized in Figure 6d. All tasks were implemented in both the simulation and the hardware.

The trajectory tracking task involved following predefined Cartesian paths, including circular and star-shaped trajectories. For example, a circular trajectory was defined as $\mathbf{X}_{ref}(t) = [r\cos t, r\sin t, z_0]$. The trained policy reproduced these paths on the real robot with high spatial accuracy, as confirmed by comparing the simulated trajectory $\mathbf{X}_p(t)$ and the real measurement $\mathbf{X}_R(t)$.

The force-induced deformation estimation task evaluated the agent's ability to predict the resulting displacement caused by external loading. A weight was applied to the TCP, shifting the configuration from the unloaded state $\mathbf{X}_{w/o}$ to the deformed configuration $\mathbf{X}_w$. The agent inferred the deformed position under load, predicting $\mathbf{X}_w$ based on the applied force (τ_{est}) and the current state of the manipulator.

The hybrid control task combined trajectory tracking with force regulation. The agent was required to follow a tilted elliptical path while maintaining a constant normal contact force. The agent achieved accurate performance in both objectives, with contact force vectors that closely matched target values throughout the task.

All tasks were implemented in a mirrored PyBullet simulation that included identical geometry, contact surfaces, and force measurement instrumentation, as shown in Figure 6c. This ensured consistency between simulation and real-world deployment for effective policy transfer.

7. Result

7.1. Calibration Results and Validation

The reachable configuration spaces of the real robot, the FEM simulation, and the surrogate model were collected under identical actuation constraints and compared. **Figure 7**a shows the resulting trajectories in the Cartesian plane. The surrogate model covered the entire reachable space observed in both the FEM simulation and the real robot, including all positions recorded in physical experiments. The FEM and the real robot exhibited close agreement in their ROM.

The differences between the FEM model and the real robot were quantified in five configuration variables: tip position (p_S), tip velocity ($\dot{p}_S$), arc length (l_S), orientation (q_S), and curvature (κ_S). Configuration errors were normalized using min–max scaling within each variable's domain-specific range. The normalized mean errors and standard deviations for each variable are shown in Figure 7b and summarized in **Table 1**.

Two neural network mappings were used to translate between the domains: f_{RS} from real to simulated pressures, and f_{SP} from simulated pressures to the surrogate model's joint angles. The average prediction error for f_{RS} was 0.13 kPa. For the f_{SP} mapping, the average prediction error was recorded as 0.32 deg for the rotational joints ($\theta_{x,y}$) and 4.23 mm for the prismatic joints (θ_d), as detailed in Table 1. It is important to note that these joint-level errors do not directly propagate to the end-effector's

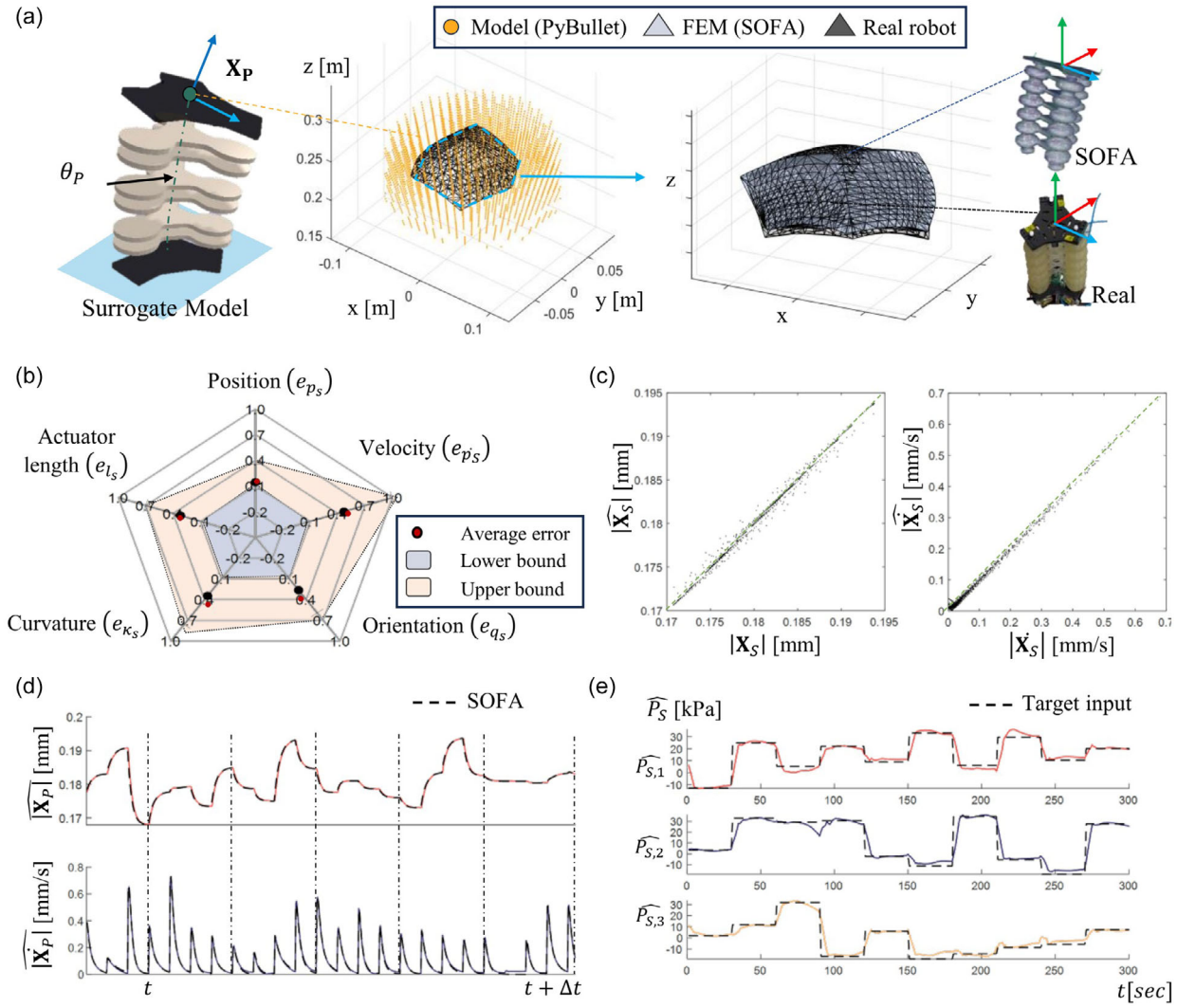

Figure 7. a) Comparison of the reachable configuration space across the three domains: real robot, FEM simulation, and surrogate model. Each boundary represents the ROM achieved under actuation constraints. b) Normalized average errors and standard deviations for key configuration variables, comparing FEM simulation and real-world measurements. Metrics include position, velocity, orientation, curvature, and actuator length. c) Prediction accuracy of the learned forward dynamics model f_S, showing the absolute values of the predicted state $\widehat{\mathbf{X}}_S$ and velocity $\dot{\widehat{\mathbf{X}}}_S$ compared to ground truth from the FEM simulation. d) Optimized mapping from FEM dynamics to the joint space of the surrogate model, aligning the predicted states $\widehat{\mathbf{X}}_P$ and $\dot{\widehat{\mathbf{X}}}_P$ over time $[t, t + \Delta t]$. e) Inverse dynamics results predicting actuator pressures $\hat{P}_S$ from target motion states, demonstrating the capability to reconstruct input pressures from desired trajectories.

Table 1. Mapping errors between domains.

Mapping	Variable	Value (Mean ± Std)	Unit
MLP functions	f_{RS}	0.13 ± 0.02	kPa
	f_{SP}	$\theta_{x,y}$: 0.32 ± 0.02	deg
		θ_d : 4.23 ± 0.06	mm
Model calibration (Real–SOFA)	l_S	0.30 ± 0.06	mm
	q_S	0.35 ± 0.08	deg
	κ_S	0.21 ± 0.04	mm^{-1}
	p_S	1.40 ± 0.62	mm
	p_S	12.69 ± 4.29	mm s^{-1}
Dynamics learning (SOFA–PyBullet)	$\hat{\mathbf{x}}_p$	3.35 ± 0.43	mm
	$\hat{\dot{\mathbf{x}}}_p$	14.56 ± 4.38	mm s^{-1}
Inverse dynamics	$\hat{p}_S$	0.38 ± 0.05	kPa
	$\hat{\theta}_p$	0.17 ± 0.04	deg

Cartesian position. This effect is attributed to the kinematic redundancy of the soft manipulator, where the mapping from a three-pressure input to a multiple joint space allows for multiple joint configurations to achieve a similar end-effector pose.

7.2. Results of Learned Dynamics Models

The performance of the learned forward dynamics model (f_S) was evaluated using validation sequences collected from the FEM simulation. Figure 7c presents the predicted states ($\widehat{\mathbf{X}}_\mathbf{S}$) and velocities ($\widehat{\dot{\mathbf{X}}}_\mathbf{S}$) over time. Quantitative prediction errors are reported in Table 1, with mean errors of 3.35 mm for position and 14.56 mm s^{-1} for velocity.

Figure 7d shows the results of transferring the learned forward dynamics model f_S to the joint space of the surrogate model using the mapping function f_{SP}. Given a target state ($\mathbf{X}_\mathbf{S}$) and a velocity ($\dot{\mathbf{X}}_\mathbf{S}$), the corresponding joint configuration ($\boldsymbol{\theta}_P$) was estimated and applied to the surrogate model.

The inverse dynamics model estimated the actuator pressures of desired motion targets. Given the target state ($\mathbf{X}_\mathbf{S}$) and the velocity ($\dot{\mathbf{X}}_\mathbf{S}$), the model predicted the required input pressures ($\hat{P}_{S,i}$) to drive the system. Figure 7e illustrates the estimated pressures for multiple samples. Table 1 reports the mean pressure prediction error as 0.38 kPa and the corresponding joint angle reconstruction error as 0.17 deg.

7.3. Policy Evaluation

The trained RL agents were evaluated on three control tasks involving position, force, and hybrid objectives. Each policy was trained in the customized gym environment using joint-level actuation of the surrogate model and then transferred to the real robot using the mapping functions f_{SP} and f_{PR}, as shown in Figure 6b. The policies were trained using three different RL agents, and the detailed reward progression over training episodes is provided in the Supporting Information Section 9. For each task, the policy with the highest final reward among the three algorithms was selected for evaluation.

In Task 1, the agent executed predefined Cartesian trajectories, including circular and star-shaped paths. **Figure 8**a presents the trajectory tracking results in both the simulation ($\mathbf{X}_p$) and the real system ($\mathbf{X}_\mathbf{R}$). The simulated execution by the surrogate model is shown in blue, while the sim2real execution is shown in red. **Table 2** reports the average tracking errors for the circular and star trajectories, with surrogate-level errors of 1.02 and 1.24 mm, and corresponding sim2real errors of 3.38 and 6.79 mm.

In Task 2, the agent was given a target state representing the deformation under external loading (τ_{ext}) and was required to estimate the resulting position. Figure 8b shows the unloaded state ($\mathbf{X}_{w/o}$) and the target deformed position obtained from the simulation, marked as red crosses ($\overline{\mathbf{X}}_w$). The agent's predictions from multiple trials are shown as blue circles ($\mathbf{X}_w$), representing the estimated deformed positions under load. Table 2 reports the average position errors with respect to the simulation target, yielding 3.20 mm in simulation and 4.68 mm in sim2real.

In Task 3, the agent followed a tilted elliptical trajectory ($\mathbf{X}_{ref}$) while regulating contact force (F_n). Figure 8c shows the executed trajectory on the reference surface (S_{ref}). The norm-2 force tracking error is shown in Figure 8d, and the component-wise force profiles (F_x, F_y, F_z) are shown in Figure 8e. As reported in Table 2, the simulation errors were 1.35 mm in position and 0.55 N in force. The corresponding sim2real errors were 4.20 mm and 0.70 N.

7.4. Additional Experimental Validation for Performance and Robustness

For further validation of the proposed sim2real agent and to analyze its performance in more challenging scenarios, three additional experiments were conducted. These experiments focused on policy generalization, robustness to external disturbances, and a quantitative characterization of the system's physical limitations. They collectively demonstrate the practical viability and resilience of the sim2real pipeline.

7.4.1. Generalization to Complex 3D Trajectories

The generalization capability beyond the training tasks was evaluated by implementing the learned policy with a complex, nonplanar 3D trajectory not seen during training. This task required the soft manipulator to trace a figure-eight path on a curved surface. As depicted in **Figure 9**a, the real robot successfully tracked this intricate 3D path with high precision comparable to the result of Task 1. The average tracking error was recorded as 5.34 mm with a standard deviation of 0.38 mm. The close alignment between the reference trajectory ($\mathbf{X}_{ref}$), the surrogate model's prediction ($\mathbf{X}_p$), and the real robot's execution ($\mathbf{X}_\mathbf{R}$) confirms that the framework can generalize effectively to complex, 3D trajectories.

7.4.2. Robustness to External Disturbances

In order to assess the robustness of the agent as a closed-loop controller, a physical perturbation experiment was performed

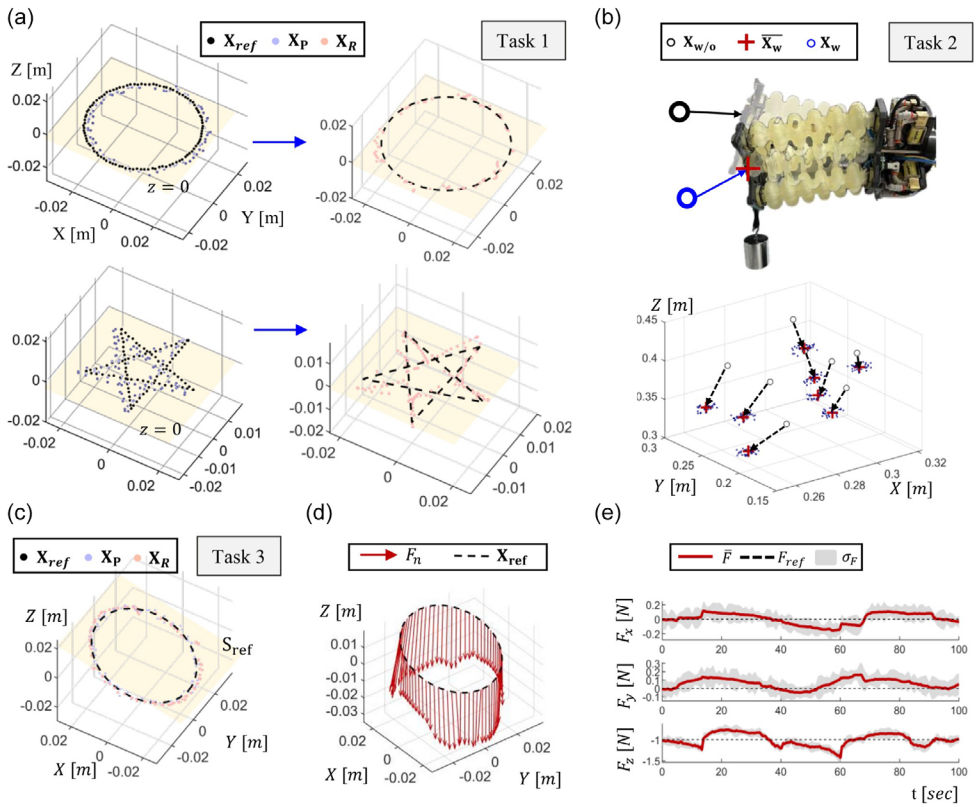

Figure 8. a) Trajectory tracking results for Task 1. The plot on the left shows the prediction of the surrogate model in simulation ($\mathbf{X_P}$, blue), while the plot on the right shows the sim2real results using the real robot ($\mathbf{X_R}$, red). Both results follow the predefined target trajectory $\mathbf{X}_{\text{ref}}$. b) Task 2: deformation compensation under external load. The unloaded position $\mathbf{X}_{w/o}$ and the externally deformed position $\overline{\mathbf{X}_w}$ are shown as red crosses, while the corrected position predicted by the agent $\mathbf{X}_w$ is shown as blue circles. c) Task 3: hybrid control results in which the agent follows a tilted elliptical trajectory $\mathbf{X}_{\text{ref}}$ on the sloped reference surface $\mathbf{S}_{\text{ref}}$, while maintaining contact. d) Norm-2 force tracking error $\|\mathbf{F} - \mathbf{F}_{\text{ref}}\|$ during Task 3, showing temporal accuracy of the contact force. e) Component-wise tracking of the target contact force (F_x, F_y, F_z) over time, confirming stable force regulation.

Table 2. Task implementation errors.

System	Task	Target	Value (Mean ± Std)	Unit
RL agent ($\widehat{\mathbf{X}}_P$)	Task 1	$\mathbf{X}_{\text{ref}}$: Circular	1.02 ± 0.61	mm
		$\mathbf{X}_{\text{ref}}$: Star	1.24 ± 0.50	mm
	Task 2	$\mathbf{X}_w$	3.20 ± 0.52	mm
	Task 3	$\mathbf{X}_{\text{ref}}$	1.35 ± 0.43	mm
		F_{ref}	0.55 ± 0.19	N
Sim2real ($\widehat{\mathbf{X}}_R$)	Task 1	$\mathbf{X}_{\text{ref}}$: Circular	3.38 ± 1.49	mm
		$\mathbf{X}_{\text{ref}}$: Star	6.79 ± 1.56	mm
	Task 2	$\overline{\mathbf{X}}_w$	4.68 ± 1.77	mm
	Task 3	$\mathbf{X}_{\text{ref}}$	4.20 ± 1.22	mm
		F_{ref}	0.70 ± 0.16	N

on the real robot. During the planar circular trajectory tracking task from Task 1, a disturbance was induced via an intentional air supply disconnect to one of the three pneumatic chambers during the actuation. The supply was then reconnected five seconds later to confirm how the agent recovers from the external disturbance. As shown in Figure 9b, a sharp deviation from the trajectory, followed by a rapid and stable recovery. The policy compensated for the error and converged back to the circular reference path within five seconds of the line being reconnected.

7.4.3. Bandwidth Limitation in Policy Rollout

When validating the learned rollout policy on the physical hardware, the system's operational speed is a critical factor. To investigate the performance limitations, the bandwidth of the hardware was quantitatively analyzed. The mean position error was measured for a point-to-point positioning task while systematically increasing the input command frequency. The results, shown in Figure 9c, reveal that a critical bottleneck exists within the hardware itself. A sharp increase in tracking error was observed as the frequency exceeded ≈2 Hz. The measured position errors increased from 13.8 to 63.9 mm after this frequency. This analysis demonstrates that the control speed during policy rollout is constrained by the physical response of the pneumatic hardware, not the inference rate of the policy. The detailed error and standard deviation values for each frequency are provided in the Supporting Information Section 1.

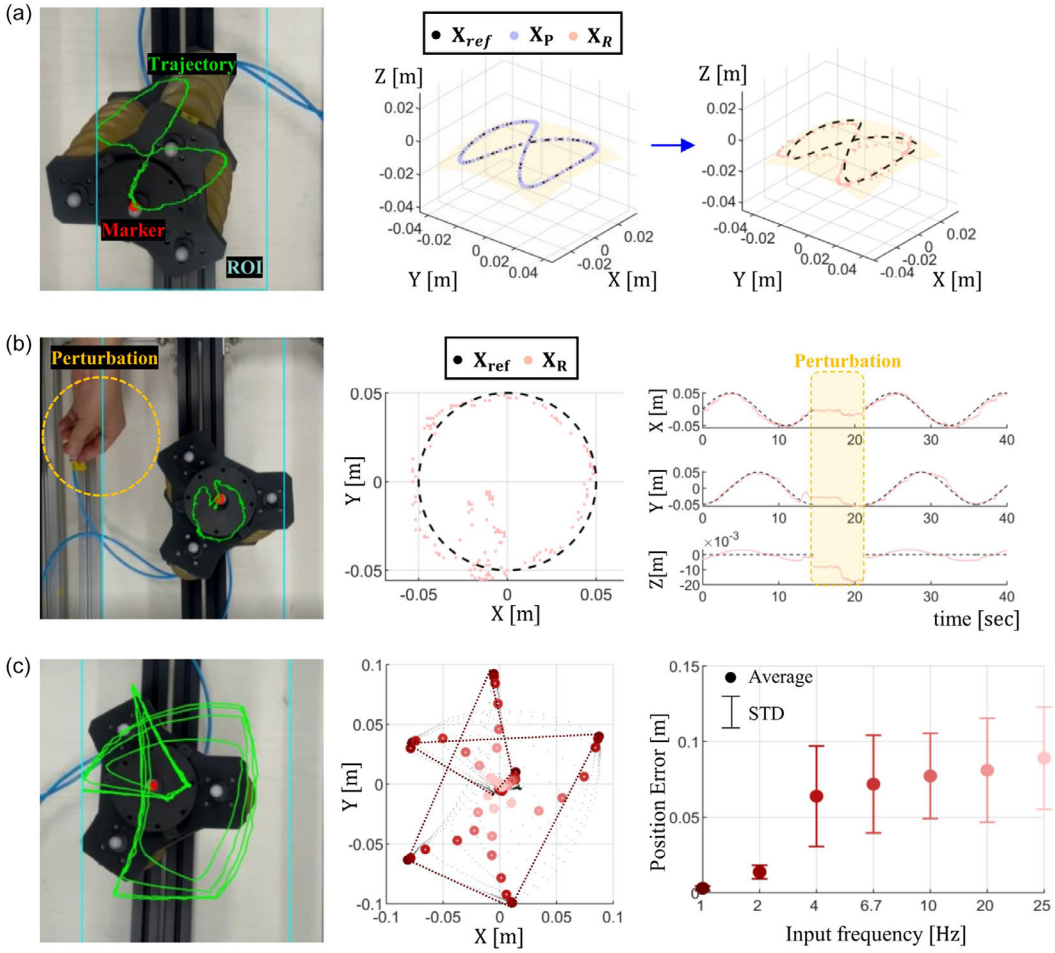

Figure 9. a) Generalizability of sim2real transfer is demonstrated by tracking a complex 3D nonplanar figure-eight trajectory not seen during training. b) Robustness of the RL policy is evaluated by showing successful recovery from an unexpected physical disturbance (temporary pneumatic line disconnection) during a circular trajectory task. c) Quantitative characterization of the hardware-imposed control bandwidth is presented by plotting the mean position error versus input pressure command frequency.

8. Discussion

This study introduced an integrated sim2real framework for soft robot control that connects high-fidelity FEM simulation, dynamics modeling, surrogate abstraction, and RL. The framework was designed to address key challenges in soft robot learning, including nonlinear deformation, high compliance, and actuation redundancy, while maintaining compatibility with standard RL toolkits.

The FEM-based model, constructed in SOFA, was calibrated against real robot measurements using dense pressure sweeps and motion capture data. The MOR formulation using POD enabled efficient simulation while preserving key deformation modes. The resulting dataset supported the training of forward and inverse dynamics models using a Transformer-PINN architecture.

A data-driven surrogate model was developed to enable real-time simulation and policy training. The model replicated the ROM and actuation responses of the FEM simulation using a joint link structure implemented in PyBullet. The learned mappings between FEM pressure inputs and surrogate joint angles preserved physical consistency. This allowed the surrogate to be used interchangeably during training while maintaining alignment with FEM-based predictions.

The forward and inverse dynamics models enabled accurate motion prediction and actuation inference within the FEM and surrogate domains. The learned dynamics generalized across unseen trajectories and supported stable behavior generation when transferred to the surrogate model. The use of actuation-space mappings ensured that predictions from the forward model could be faithfully reproduced through the joint-level abstraction, preserving temporal structure and physical consistency.

Across all tasks evaluated, the learned policies demonstrated robust trajectory execution and force regulation in simulation. When transferred to the real robot, the policies maintained task objectives but exhibited increased errors due to physical discrepancies not captured in the simulation pipeline. These included unmodeled contact hysteresis, actuator latency, and material

friction. Despite these factors, the sim2real transfer was stable, indicating that the surrogate abstraction and dynamics learning components effectively supported zero-shot deployment. To further probe the real-world capabilities of the framework, additional experiments were performed. These results confirmed the policy's ability to generalize to complex, nonplanar 3D trajectories not encountered during training. Furthermore, the controller demonstrated robustness, successfully recovering from a large, intentional physical disturbance, highlighting its closed-loop nature.

Several limitations remain. The surrogate model, while effective for training, does not explicitly model elastic body interactions or distributed compliance. Incorporating pseudocontinuum elements or compliant joint structures may improve realism in contact scenarios. The inverse model also assumes deterministic mapping, which may limit robustness under uncertain loading or sensor noise.

Future work will extend the framework to contact-rich tasks such as manipulation, locomotion, or environmental interaction on deformable substrates. Enhancing the fidelity of contact simulation and tactile feedback in the surrogate domain may further improve the transfer accuracy.

Although this study focused on a pneumatic manipulator, the architecture is modular and actuator-agnostic, and could be applied to other systems such as fiber-reinforced actuators or tendon-driven arms. In contrast to task-specific pipelines[31,32,43,58] or finely-tuned classical controllers, this approach emphasizes generalization and adaptability. While a classical controller might outperform on a single, known task, the strength of the presented RL-based approach lies in its generalization capabilities, as demonstrated by its success on the unseen tasks. Furthermore, unlike other RL frameworks that often report qualitative task success, our pipeline provides a pathway for developing controllers for high-precision tasks that demand quantitative accuracy. This focus on quantitative, generalizable, and robust performance, validated through extensive real-world experiments, positions the framework as a flexible and powerful tool for developing learning-based controllers for a wide range of complex soft robotic systems.

Supporting Information

Supporting information and videos are available from the Wiley Online Library. The code used in this study is available at https://github.com/SNU-SRBL/SoftRobot_sim2real.

Acknowledgements

This work was supported in part by the National Research Foundation (RS-2023-00208052) and in part by the Industrial Strategic Technology Development Program (RS-2024-00423940), both funded by the Korean Government (MSIT and MOTIE, respectively).

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are openly available in [GitHub] at [https://github.com/SNU-SRBL/SoftRobot_sim2real], reference number [147309738].

Keywords

data-driven surrogate model, model order reduction, physics-based simulation, reinforcement learning, sim-to-real transfer

Received: June 24, 2025
Revised: September 29, 2025
Published online:

[1] C. Laschi, B. Mazzolai, M. Cianchetti, *Sci. Rob.* **2016**, *1*, eaah3690.
[2] G. M. Whitesides, *Angew. Chem. Int. Edit.* **2018**, *57*, 4258.
[3] J. Shintake, V. Cacucciolo, D. Floreano, H. Shea, *Adv. Mater.* **2018**, *30*, 1707035.
[4] D. Rus, M. T. Tolley, *Nature* **2015**, *521*, 467.
[5] C. Majidi, *Soft Rob.* **2014**, *1*, 5.
[6] S. Kim, C. Laschi, B. Trimmer, *Trends Biotechnol.* **2013**, *31*, 287.
[7] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, P. Dario, *Adv. Rob.* **2012**, *26*, 709.
[8] C. Della Santina, C. Duriez, D. Rus, *IEEE Control Syst. Mag.* **2023**, *43*, 30.
[9] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, F. Renda, *IEEE Trans. Rob.* **2023**, *39*, 1728.
[10] D. Marcheseandrew, D. Onalcagdas, D. Rus., *Soft Rob.* **2014**, *1*, 1.
[11] D. C. Rucker, B. A. Jones, R. J. Webster III, *IEEE Trans. Rob.* **2010**, *26*, 769.
[12] F. Corucci, N. Cheney, S. Kriegman, J. Bongard, C. Laschi, *Front. Rob. AI* **2017**, *4*, 34.
[13] M. Cianchetti, C. Laschi, A. Menciassi, P. Dario, *Nat. Rev. Mater.* **2018**, *3*, 143.
[14] 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.
[15] M. S. Nazeer, C. Laschi, E. Falotico, *IEEE Trans. Rob.* **2024**, *40*, 2498.
[16] Q. Wang, Z. Wu, J. Huang, Z. Du, Y. Yue, D. Chen, D. Li, B. Su, *Compos. Part B: Eng.* **2021**, *223*, 109116.
[17] Y. Choi, G. Shin, S. J. Yoon, Y.-L. Park, *Soft Rob.* **2024**, *12*, 1.
[18] E. Coevoet, A. Escande, C. Duriez, *IEEE Robot. Autom. Lett.* **2017**, *2*, 1413.
[19] L. Ding, L. Niu, Y. Su, H. Yang, G. Liu, H. Gao, Z. Deng, *Chin. J. Mech. Eng.* **2022**, *35*, 24.
[20] J. M. Bern, P. Banzet, R. Poranne, S. Coros, *Robotics: Science and Systems*, Vol. *1*, **2019**.
[21] C. Duriez, in *2013 IEEE Inter. Conf. on Robotics and Automation*, IEEE, Karlsruhe, Germany **2013**, pp. 3982–3987.
[22] 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*, e0246102.
[23] K. Chin, T. Hellebrekers, C. Majidi, *Adv. Int. Syst.* **2020**, *2*, 1900171.
[24] B. Jumet, M. D. Bell, V. Sanchez, D. J. Preston, *Adv. Int. Syst.* **2022**, *4*, 2100163.
[25] H. Zhang, R. Cao, S. Zilberstein, F. Wu, X. Chen, in *Intelligent Robotics and Applications: 10th Inter. Conf., ICIRA 2017, Wuhan, China, August 16–18, 2017, Proceedings, Part I 10*, Springer, Wuhan, China **2017**, pp. 173–184.

[26] M. Raeisinezhad, N. Pagliocca, B. Koohbor, M. Trkov, *Front. Rob. AI* **2021**, *8*, 639102.
[27] D. Kim, S. Lee, T. H. Hong, Y.-L. Park, *npj Rob.* **2023**, *1*, 7.
[28] S. Bhagat, H. Banerjee, Z. T. Ho Tse, H. Ren, *Robotics* **2019**, *8*, 4.
[29] J. Kober, J. A. Bagnell, J. Peters, *Int. J. Rob. Res.* **2013**, *32*, 1238.
[30] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, *IEEE Signal Process. Mag.* **2017**, *34*, 26.
[31] T. G. Thuruthel, E. Falotico, F. Renda, C. Laschi, *IEEE Trans. Rob.* **2018**, *35*, 124.
[32] T. Yang, Y. Xiao, Z. Zhang, Y. Liang, G. Li, M. Zhang, S. Li, T.-W. Wong, Y. Wang, T. Li, Z. Huang, *Sci. Rep.* **2018**, *8*, 14518.
[33] C. Della Santina, R. K. Katzschmann, A. Biechi, D. Rus, in *2018 IEEE Int. Conf. on Soft Robotics (RoboSoft)*, IEEE, Livorno, Italy **2018**, pp. 46–53.
[34] T. G. Thuruthel, B. Shih, C. Laschi, M. T. Tolley, *Sci. Rep.* **2019**, *4*, eaav1488.
[35] W. Huang, X. Huang, C. Majidi, M. K. Jawed, *Nat. Commun.* **2020**, *11*, 2233.
[36] Z. Wan, Y. Sun, Y. Qin, E. H. Skorina, R. Gasoto, M. Luo, J. Fu, C. D. Onal, *Soft Rob.* **2023**, *10*, 258.
[37] D. A. Haggerty, M. J. Banks, E. Kamenar, A. B. Cao, P. C. Curtis, I. Mezić, E. W. Hawkes, *Sci. Rob.* **2023**, *8*, eadd6864.
[38] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, C. Duriez, in *2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, Seattle, WA, USA **2015**, pp. 2550–2555.
[39] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, C. Duriez, *Adv. Robot.* **2017**, *31*, 1208.
[40] O. Goury, C. Duriez, *IEEE Trans. Rob.* **2018**, *34*, 1565.
[41] O. Goury, B. Carrez, C. Duriez, *IEEE Rob. Autom. Lett.* **2021**, *6*, 3752.
[42] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, C. Duriez, *Soft Rob.* **2023**, *10*, 410.
[43] M. A. Graule, T. P. McCarthy, C. B. Teeple, J. Werfel, R. J. Wood, *IEEE Rob. Autom. Lett.* **2022**, *7*, 4071.
[44] L. Qin, H. Peng, X. Huang, M. Liu, W. Huang, *Curr. Rob. Rep.* **2024**, *5*, 1.
[45] S. Ku, B.-H. Song, T. Park, Y. Lee, Y.-L. Park, *Int. J. Rob. Res.* **2024**, *43*, 7.
[46] W. Dou, G. Zhong, J. Cao, Z. Shi, B. Peng, L. Jiang, *Adv. Mater. Technol.* **2021**, *6*, 2100018.
[47] Z. Gong, B. Chen, J. Liu, X. Fang, Z. Liu, T. Wang, L. Wen, *Front. Rob. AI* **2019**, *6*, 26.
[48] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, D. Rus, in *2019 2nd IEEE Int. Conf. Soft Robotics (RoboSoft)*. IEEE, Philadelphia, PA, USA **2019**, 717–724.
[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Long Beach, CA, USA **2017**.
[50] M. Raissi, P. Perdikaris, G. E. Karniadakis, *J. Comput. phys.* **2019**, *378*, 686.
[51] Z. Zhao, X. Ding, B. A. Prakash, arXiv preprint arXiv:2307.11833 **2023**.
[52] C. Zhu, R. H. Byrd, P. Lu, J. Nocedal, *ACM Trans. Math. Soft. TOMS* **1997**, *23*, 550.
[53] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W.Zaremba **2016**.
[54] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, S. Levine, arXiv preprint arXiv:1812.05905 **2018**.
[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, arXiv preprint arXiv:1707.06347 **2017**.
[56] F. Torabi, G. Warnell, P. Stone, arXiv preprint arXiv:1805.01954 **2018**.
[57] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, *J. Mach. Learn. Res.* **2021**, *22*, 1.
[58] X. Liu, C. D. Onal, J. Fu, *IEEE Transactions on Robotics* **2023**, *39*, 3382.