Ph.D. Dissertation in Mechanical Engineering

Accelerated, High-Fidelity Simulation of Deformable Bodies for Physics Abstraction in Robot Learning

물리 추상화 기반의 로봇 학습을 위한 가속화된 고충실도 변형체 시뮬레이션

February 2026

Department of Mechanical Engineering
Graduate School of Seoul National University

Taehwa Hong

Accelerated, High-Fidelity Simulation of Deformable Bodies for Physics Abstraction in Robot Learning

Thesis Advisor: Prof. Yong-Lae Park

A dissertation submitted to the
Department of Mechanical Engineering
for the degree of
Doctor of Philosophy

October 2025

Graduate School of Seoul National University

The dissertation of Taehwa Hong
has been approved

December 2025

Chair	Frank Chongwoo Park	(Signature)
Vice Chair	Yong-Lae Park	(Signature)
Examiner	Jinkyu Yang	(Signature)
Examiner	Ayoung Kim	(Signature)
Examiner	Daekyum Kim	(Signature)

Abstract

The advancement of learning-based control in soft robotics is fundamentally constrained by the simulation to real gap, a critical divergence between computational speed and physical fidelity. This thesis provides a rigorous analysis of existing modeling paradigms to define this gap mathematically and justify the necessity for Data-Driven Abstraction. We demonstrate that analytical approximations fail to capture essential non-linear behaviors under external loads, while ground-truth numerical methods, like the finite element method, scale with cubic complexity, rendering them intractable for the massive data throughput required by reinforcement learning. To resolve this discrepancy, this dissertation proposes a unified pipeline that decouples physical ground-truth generation from runtime execution. A series of research studies established the theoretical foundations for two targeted abstraction strategies: (1) Surrogate modeling via model order reduction to compress macroscopic body dynamics into fast, rigid-body-compatible formats, and (2) Neural physics engines utilizing data-driven contact model approximations to capture the complex, high-dimensional mechanics of the contact interface. This methodology aims to bridge the reality gap by achieving the computational efficiency of rigid-body engines without sacrificing the continuum physics essential for robust sim-to-real transfer. Ultimately, this work contributes a generalizable pipeline to abstract high-dimensional Finite Element data into low-dimensional but essential features, applying this strategy to the two major domains of deformation-incurring systems in robotics to enable scalable and physically faithful learning.

Keyword : Physics-based Simulation, Sim-to-Real Transfer, Data-driven Abstraction, Reinforcement Learning, Soft Robotics, Vision-Based Tactile Sensor

Student Number : 2019-20340

Table of Contents

List of Figures

List of Tables

Chapter 1.
Introduction

1.1. Deformable Systems in Physical Interaction

The progression of robotics toward general-purpose physical intelligence is fundamentally constrained by the ability to master interaction with unstructured environments [1–3]. Unlike the idealized rigid-body dynamics often assumed in classical control theory [4–6], the physical world is characterized by continuous compliance and complex deformation. This phenomenon is not merely a disturbance but the fundamental mechanism of interaction inherent to any physical agent. Regardless of the specific hardware implementation, the mechanics of continuum deformation govern the fundamental capabilities of robotic agents to move [7, 8], manipulate [9, 10], and perceive [11, 12]. Consequently, the mastery of these systems requires a fundamental shift from rigid-body assumptions to models that can accurately represent and exploit continuum behavior.

A grand challenge in mastering these systems lies in the representation of deformation. Conventional approaches utilizing lumped parameter models [13], voxelization [14], and rigid body approximations [15] inherently discard the high-dimensional nature of continuum behavior. These methods reduce a distributed spatial phenomenon to discrete points and lose the critical stress and strain field information required for high-fidelity interaction [16, 17]. Consequently, the only valid ground truth for analyzing deformable systems is continuum mechanics, numerically realized through mesh-based Finite Element Methods (FEM) [18]. A mesh-based representation discretizes the continuous body into finite elements to capture the non-linear transmission of force and the complex evolution of shape that define the physical reality of soft interaction [19, 20].

Establishing a high-fidelity mathematical representation creates a subsequent necessity for extensive data acquisition when applied to modern learning-based control and perception frameworks. The most dominant and modern method of training robots is Reinforcement Learning (RL), which requires massive datasets involving millions of trial-and-error explorations to approximate the complex manifolds of control and perception [21–23]. The acquisition of such data in the real world is fundamentally unscalable due to physical limitations, including material fatigue, hysteresis, and the risk of catastrophic damage during the exploration phase. Therefore, the utilization of a simulation environment is not merely a convenience but a strict prerequisite for the development of learning-based controllers and perception models [24–26].

1.2. Computational Bottleneck of Simulating Deformation

The physical behavior of deformable robotic components is fundamentally governed by the laws of continuum mechanics. When a compliant material undergoes interaction, the process is described mathematically by a mapping from a reference configuration to a current configuration, where the internal stress state is determined by a constitutive law such as a hyperelastic Neo-Hookean [27] or Mooney-Rivlin model [28]. Accurately simulating this behavior requires enforcing the conservation of momentum and mass, which necessitates solving Partial Differential Equations (PDEs). The FEM addresses this by discretizing the continuous body into a mesh, where a single component requiring fine detail can easily exceed ten thousand nodes. This results in a system of equations with tens of thousands of degrees of freedom.

This high dimensionality creates a computational bottleneck often referred to as the curse of dimensionality. Solving the resulting linear system involving the stiffness matrix at every time step is an operation with cubic complexity, depending on the solver. While this process yields high-fidelity physics capable of resolving distributed pressure maps and tangential shear

fields, the computational load is prohibitive for dynamic simulations running at the high frequencies required for robot control. This inherent slowness of high-fidelity physics stands in sharp contrast to standard robotics simulators, which typically utilize rigid body dynamics [29–31]. These engines treat links as rigid components and model contact as a set of constraints preventing inter-penetration [32–34]. While this approximation allows for computation in microseconds, it fundamentally discards the essential information required for deformation-based interactions and yields a substantial reality gap for soft systems.

This discrepancy between physical fidelity and computational speed creates a specific crisis for robot learning, which requires simulators that are simultaneously fast and faithful. To effectively bridge this gap, a valid simulator must satisfy three distinct criteria to avoid common failure modes in learning. First, it must possess physical fidelity sufficient to capture the non-linear continuum mechanics of the real system to prevent negative transfer during reality gap crossing. Second, it must achieve computational speed comparable to rigid-body engines to enable the massive throughput required for policy convergence. Third, it must be differentiable allowing the model to be fine-tuned via real-world data to correct for inevitable parameter mismatches such as material fatigue or unmodeled friction. Current simulation paradigms force a compromise by offering either fidelity or speed, but rarely both, and almost never with the adaptability required for robust sim-to-real transfer.

1.3. Motivation and Rationale

The limitations of existing simulation paradigms present a fundamental dilemma in robotic learning, where no single approach simultaneously satisfies the requirements of physical fidelity and computational speed. Analytical models provide speed but lack generality, while numerical solvers provide fidelity but lack the necessary throughput. This dissertation proposes that the viable methodology to resolve this conflict is to decouple the generation of physical ground truth from the runtime execution of the simulation. This research

operates on the premise that the complex behavior of deformable continuum systems, while mathematically infinite-dimensional, can be effectively represented by a compressed set of latent features governed by the specific structural and material constraints of the robot. Consequently, the solution is not to simplify the physics equations *a priori* but to utilize high-fidelity FEM data as an offline teacher to learn a compressed and differentiable inference model.

The concept of this approach is shown in Figure 1.1 that constitutes a paradigm shift from numerical solving to data-driven inference. In a traditional solver, the system state is computed by iteratively resolving partial differential equations at every time step, a process constrained by the complexity of the mesh. In the proposed data-driven pipelines, the physics engine effectively becomes a function approximator. By training on valid FEM data, the model learns to predict the continuum mechanical response directly. This transformation reduces the computational complexity from a polynomial dependence on mesh size to a constant-time operation. Furthermore, unlike rigid analytical models, these learned surrogates are differentiable. This allows the simulation to be fine-tuned against sparse real-world data, thereby closing the reality gap through system identification. Therefore, data-driven abstraction serves as the essential bridge that transforms computationally intractable continuum physics into fast, faithful, and learnable simulation engines.

1.4. Methodology of Data-Driven Abstraction

Based on this rationale, this dissertation proposes a unified methodological framework for realizing accelerated high-fidelity simulation. The scope of this research encompasses the end-to-end pipeline of abstraction, which proceeds through three distinct stages. The first stage is deformation modeling, where high-fidelity FEM simulations are constructed to serve as the absolute ground truth for continuum behavior. The second stage is feature abstraction, where the high-dimensional mesh data is processed to extract the essential low-dimensional features, such as kinematic modes or topological graphs relevant to the specific robotic function. The third stage is training the inference model,

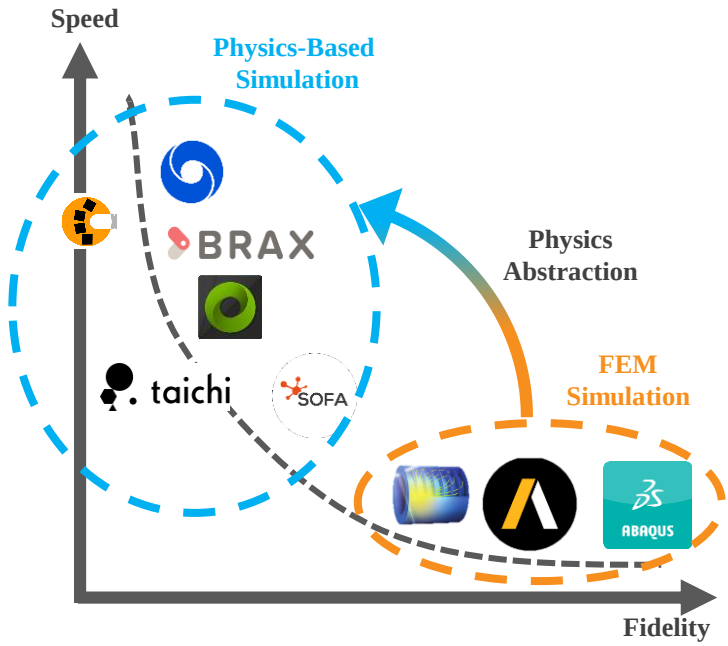


Figure 1.1: Tradeoff between the fidelity and the speed of the deformation simulators.

where deep neural networks are trained to map the system inputs to these abstracted states, effectively replacing the iterative numerical solver with a fast, differentiable function approximator.

To demonstrate the efficacy and generality of this framework, the research applies this abstraction strategy to the two topologically distinct domains of robotic interaction: the **deformable body** (action) and the **contact interface** (perception).

- **Part A: Dynamics of the Deformable Body.** The first domain is volumetric deformation which occurs within the internal structure of the robot. In the context of soft robotics, this internal deformation is coupled with actuation to define the macroscopic configuration and dynamic workspace of the system. Here, the functional goal is macroscopic control. Therefore, the abstraction strategy prioritizes global kinematic compatibility to enable the training of whole-body motion policies. This is

realized through a surrogate model which maps continuum deformation into a virtual kinematic chain allowing soft robots to be simulated within standard rigid-body physics engines with high computational efficiency.

- **Part B: Modeling the Soft Contact Interface.** The second domain is boundary deformation which occurs at the interface between the robot and the environment. Here, the localized deformation of the surface layer encodes the contact geometry and force distribution essential for exteroceptive perception. The functional goal is microscopic perception. Therefore, the abstraction strategy prioritizes local surface topology and the generalization of contact geometry to unseen objects. This is realized through a Neural Physics Engine (NPE) based on trained mapping functions. The NPE alters the solver in the physics-based simulation to provide physical information regarding contact with deformation, a capability that is absent in vanilla rigid-body versions.

This division ensures that the abstraction is specifically tailored to the distinct physical requirements of action and perception for the robotic applications, rather than applying a generic reduction.

1.5. Dissertation Contributions and Organization

This dissertation is organized to systematically validate the proposed abstraction framework through a logical progression from theoretical foundations to specific methodological applications. Chapter 2 establishes the essential background by analyzing the structural limitations of current robotic simulation paradigms. It contrasts the computational efficiency of rigid-body dynamics engines with the physical fidelity of FEM to quantitatively define the simulation gap. Furthermore, this chapter reviews the theoretical foundations of model order reduction and data-driven physics learning to set the stage for the specific abstraction strategies employed in subsequent chapters.

The methodological contributions are divided into two primary parts. Part A focuses on the dynamics of the deformable body and encompasses

Chapters 3 and 4. Chapter 3 establishes the theoretical baseline by deriving an analytical kinematic model for a pneumatic origami manipulator [35]. This study demonstrates the utility of geometric abstraction for real-time control while identifying the inherent limitations regarding generality that motivate the subsequent data-driven approach. Chapter 4 introduces the surrogate modeling framework for deformable body dynamics. This work abstracts high-fidelity finite element analysis into a fast rigid-body simulation to enable reinforcement learning for soft robot control [36].

Part B focuses on the modeling of the soft contact interface and encompasses Chapters 5 and 6. Chapter 5 develops a bidirectional data pipeline for vision-based tactile sensors that utilizes calibrated simulations to bridge the visual and physical domains of contact [37]. Chapter 6 culminates in the development of the NPE. This geometry-aware graph neural network (GNN) replaces standard contact solvers to estimate high-dimensional finite element information in real-time. This integration enables the simulation of vision-based tactile sensors within rigid-body environments with physically accurate deformation and force. The utility of this framework is validated by extracting latent physical vectors to train a behavioral cloning policy for a real-world, contact-rich peg-in-hole task.

Finally, Chapter 7 synthesizes the contributions of the dissertation and discusses their broader impact on bridging the simulation gap. It argues that the proposed data-driven abstraction framework effectively resolves the conflict between computational intractability and physical fidelity, thereby removing the primary bottleneck of data scarcity for deformable systems. The chapter provides a critical analysis of current limitations and outlines future research directions, including potential applications. Ultimately, this dissertation establishes a unified methodology for transforming high-fidelity continuum mechanics into fast, tractable data generation engines, providing the essential pipeline required to train the next generation of contact-aware and physically intelligent robots.

Chapter 2.
Backgrounds and Related Work

2.1. Simulation Gap in Physical Intelligence

The core challenge within the classical domain of rigid-body robotics is considered substantially resolved. This status is attributed to the equations of motion for linked rigid segments, which are governed by Ordinary Differential Equations (ODEs). These equations are inherently well-conditioned and computationally tractable, facilitating their efficient solution using recursive algorithms that operate at super-real-time frequencies [18,32,33,38,39]. However, the transition to soft, deformable systems shows a significant divergence between physical fidelity and computational tractability. Soft systems, characterized by continuous compliance, infinite degrees of freedom, and non-linear material properties, go against the simplifying assumptions of rigid-body dynamics. This *sim-to-real gap* represents the primary bottleneck hindering the application of modern data-driven learning techniques to soft robotics [40,41].

The simulation gap establishes a binary tradeoff in current methodologies [42,43]. A simulation achieves either physical fidelity to real-world continuum mechanics or the computational speed required for the massive data throughput of Reinforcement Learning (RL), but rarely both concurrently. High-fidelity numerical methods, such as the Finite Element Method (FEM), accurately capture the ground truth of deformation. Nevertheless, their cubic complexity ($O(N^3)$) with respect to the number of mesh nodes renders them prohibitive for the millions of iterative steps essential for policy optimization [44, 45]. Conversely, analytical approximations and rigid-body physics engines [29–31,46,47] furnish the requisite speed (> 1000 Hz). However, this performance is attained by omitting essential soft interaction phenomena, including distributed contact, viscoelasticity, and volumetric deformation. This

omission introduces a reality gap that leads directly to policy failure upon transfer to physical hardware [45, 48].

This chapter provides a rigorous analysis of the existing theoretical and methodological landscape to justify the specific architectural choices proposed in this dissertation: the use of surrogate modeling for efficient whole-body dynamics (**Part A**) and a Neural Physics Engine (NPE) for high-fidelity contact perception (**Part B**). By breaking down the limitations of state-of-the-art approaches, ranging from classical modeling theories to modern differentiable simulators, this study establishes that the proposed data-driven abstraction strategies are not merely convenient optimizations but necessary structural interventions required to solve the dual problems of action and perception in deformable robots.

2.1.1 Data Scarcity Crisis in Deformable Interaction

The requirement for advanced simulation in soft robotics is directly driven by the data scarcity crisis [22, 25, 49, 50]. Unlike rigid robots, which permit direct control or moderate sim-to-real adaptation through precise modeling, soft robots exhibit complex, history-dependent behaviors [10, 51]. Capturing phenomena such as hysteresis [48], the Mullins effect [52], and viscoelastic creep [53] necessitates extensive interaction data. In a rigid system, identifying the inertial parameters of a link is a bounded problem. In a soft system, the body itself changes shape, stiffness, and dynamic response based on its current configuration and loading history.

Collecting the necessary volume of data to characterize these behaviors in the physical world is often unfeasible due to several compounding factors. First, material fatigue presents a severe limitation. Soft actuators, particularly those based on pneumatic or elastomeric principles, degrade rapidly under continuous high-cycle operation [54]. A silicone chamber subjected to thousands of pressurization cycles will exhibit altered stiffness and damping properties, effectively rendering the data collected at the beginning of an experiment invalid for the model trained at the end. A second critical consideration in-

volves the compromised safety of the platform during the exploration phase of RL. The stochastic nature of RL algorithms often generates actions that can rupture soft chambers, tear fabric constraints, or exceed the elastic limit of the material. Consequently, this risk necessitates a safe, resettable simulation environment to prevent hardware loss from catastrophic failures. Finally, state estimation in the real world is extremely difficult. Obtaining ground-truth state information, such as the full curvature of a continuum arm or the distributed pressure map of a tactile sensor, requires complex external motion capture systems that are prone to occlusion and calibration errors. Simulation, by contrast, provides privileged access to the full state vector, enabling the training of state estimators and policies that would be impossible to supervise in the physical domain. Consequently, the development of a simulator that is both fast and physically grounded is not a luxury but a prerequisite for scaling robot learning to deformable systems.

2.1.2 Division of Action and Perception in Deformation System

To rigorously address the simulation gap, this dissertation divides the problem into two topologically and functionally distinct domains: the macroscopic dynamics of the body (Action) and the microscopic mechanics of the interface (Perception), as summarized in Table 2.1. While both domains are governed by the same underlying continuum mechanics, they impose fundamentally distinct requirements on simulation.

Action Domain : This domain concerns the volumetric deformation of the robot's structure. Here, the primary challenge is kinematic compatibility. To enable effective control, the simulation must bridge the gap between infinite-dimensional continuum physics and the low-dimensional state-spaces (e.g., generalized coordinates) required by modern reinforcement learning algorithms. The goal is proprioceptive fidelity in predicting how internal actuation translates to macroscopic configuration and inertial dynamics over time.

Perception Domain : This domain concerns the deformation of the boundary layer at the contact interface. Here, the primary challenge is geometric and topological fidelity. Unlike the body, which requires global motion consistency, the interface requires the resolution of high-frequency surface deformations, local stress concentrations, and complex contact topologies to generate accurate tactile signals. The goal is exteroceptive fidelity—mapping the local imprint of the environment to a high-resolution sensory output.

As the following sections will demonstrate, state-of-the-art methods cannot simultaneously satisfy both sets of requirements. A single solver optimized for body dynamics is too coarse for tactile sensing, while a solver capable of resolving tactile micro-geometry is computationally intractable for full-body motion. This necessitates the proposed dual-architecture approach: a kinematic surrogate model for the action domain and a geometry-aware Neural Physics Engine for the perception domain.

Criteria	**Part A** (Action)	**Part B** (Perception)
Target Domain	Body Dynamics	Contact Interface
Physical Phenomenon	Volumetric Deformation	Boundary Deformation
Functional Goal	Macroscopic Control	Microscopic Perception
Primary Challenge	Kinematic Compatibility	Geometric Fidelity
Inductive Bias	Kinematic Chain (Temporal)	Graph Topology (Spatial)
Abstraction Method	Surrogate Model (Rigid Links)	Neural Physics Engine (GNN)

Table 2.1: Division of the abstraction framework into two domains.

2.2. Modeling Deformable Continuum Physics

To effectively simulate a soft robot, one must first establish the mathematical ground truth of its behavior. Unlike rigid bodies defined by six degrees of freedom, a deformable body is an infinite-dimensional system governed by the partial differential equations of continuum mechanics.

The motion of a deformable body is described by a mapping ϕ from a reference configuration to a current configuration. The primary measure of local deformation is the deformation gradient $F = \nabla_X \phi$, which maps infinitesimal line segments from the reference to the current state. For the soft elastomers typically used in soft robotics and tactile sensors, the material behavior is best described as hyperelastic. In hyperelasticity, the stress is derived from a scalar strain energy density function $\Psi(F)$, which represents the potential energy stored in the material per unit reference volume [55]. The standard model for rubber-like materials is the Neo-Hookean model, where the energy density is defined as

$$\Psi = \frac{\mu}{2}(I_1 - 3) - \mu \ln(J) + \frac{\lambda}{2}(\ln(J))^2 \tag{2.1}$$

where μ and λ are Lamé parameters, I_1 is the first invariant of the deformation tensor, and J is the volume ratio. This constitutive law captures the non-linear stress-strain relationships and quasi-incompressibility that are critical for accurate force prediction but are notably absent in linear elastic models.

Since analytical solutions to these continuous equations are impossible for arbitrary geometries, numerical discretization is required. The Finite Element Method (FEM) serves as the gold standard for this spatial discretization [18]. By subdividing the continuous domain into a finite number of elements, FEM approximates the governing equations as a system of ordinary differential equations

$$M\ddot{q} + D\dot{q} + f_{int}(q) = f_{ext} \tag{2.2}$$

where q represents the nodal positions, M is the mass matrix, and $f_{int}(q)$ is the non-linear internal force vector derived from the constitutive law. While

physically faithful, solving this system requires implicit time integration to maintain stability with stiff materials. This necessitates solving a large system of non-linear algebraic equations at every time step using Newton-Raphson iteration. The computational complexity of factorizing the resulting stiffness matrix scales between $O(N^2)$ and $O(N^3)$ depending on the mesh topology. For a high-fidelity simulation requiring thousands of nodes, this computational cost renders FEM intractable for the massive throughput required by reinforcement learning algorithms [21].

Alternative modeling approaches such as Cosserat Rod theory [56] or Piecewise Constant Curvature (PCC) models [57] offer significant speed advantages by simplifying the robot geometry to a 1D curve or a series of arcs. However, these methods suffer from a dimensionality mismatch. They assume a rigid cross-section and cannot capture the volumetric deformation, bulging, or complex contact patches that are essential for the interaction tasks considered in this work. Consequently, while useful for kinematic planning, they lack the physical fidelity required for contact-rich policy learning.

2.3. The Fidelity-Efficiency Tradeoff in Physics-Based Simulation

The inherent computational cost of high-fidelity continuum modeling has created a bifurcation in robotic simulation paradigms. Existing simulators generally fall into two categories that force a compromise between speed and accuracy.

The first category consists of Rigid Body Dynamics (RBD) engines such as MuJoCo [29], PyBullet [30], Drake [47] and IsaacSim [58]. These simulators model contact as a kinematic constraint preventing inter-penetration, typically formulated as a Linear Complementarity Problem (LCP). The LCP solvers are highly optimized for articulated systems, capable of resolving contact forces for complex chains in microseconds ($O(K)$ complexity where K is the number of constraints). This speed has made them the default choice for reinforcement learning. However, the rigid-body assumption fundamentally

discards the mechanics of deformation. Contact is treated as a discrete point rather than a distributed patch, and the compliance of the robot body is ignored. This leads to a significant reality gap where policies trained in simulation fail to transfer to the compliant real world.

The second category consists of high-fidelity soft body simulators such as SOFA [40], NVIDIA Flex [59], or commercial FEM packages. These tools use penalty-based methods or constraint solvers to resolve contact between deformable meshes. While they capture the rich physics of deformation, they are bound by the computational bottleneck of the underlying FEM solver. Even with recent advances in GPU acceleration such as Isaac Sim [58], these methods remain iterative solvers that must converge at every time step. This iterative nature makes them orders of magnitude slower than rigid-body engines and difficult to integrate into the massive parallel training pipelines required for modern RL. Furthermore, standard FEM solvers are often non-differentiable, making it difficult to automatically tune simulation parameters to match real-world data.

2.4. Data-Driven Abstraction and Surrogate Modeling

To bridge the gap between the high fidelity of FEM and the high speed of rigid-body simulation, this dissertation adopts a strategy of data-driven abstraction. This approach posits that the essential dynamics of a deformable system can be represented in a lower-dimensional space.

Mathematical Model Order Reduction (MOR) techniques, such as Proper Orthogonal Decomposition (POD) [44], provide a formal basis for this abstraction. By performing Singular Value Decomposition (SVD) on a snapshot matrix of FEM deformation data, one can identify a subspace of dominant deformation modes. Projecting the full-order dynamics onto this subspace significantly reduces the number of degrees of freedom. However, standard MOR methods still require evaluating non-linear forces in the high-dimensional space or using complex hyper-reduction techniques, and the resulting reduced

models are not natively compatible with the standard rigid-body control stacks used in robotics.

This motivates the use of learned Surrogate Models. Instead of mathematically projecting the equations, a surrogate model uses machine learning to approximate the function mapping from the current state and action to the next state. By training on high-fidelity FEM data offline, the surrogate learns to emulate the physics of the continuum body. For the specific domain of body dynamics (Part A), this dissertation proposes mapping the continuum deformation to a Virtual Kinematic Chain. This involves training a network to predict the configuration of a pseudo-rigid body model that matches the FEM deformation. This specific abstraction preserves the kinematic structure required by rigid-body physics engines, allowing the soft robot to be simulated with the stability and speed of an articulated rigid system while retaining the deformation characteristics learned from the ground truth.

2.5. Simulation for Vision-Based Tactile Sensing

While surrogate models address the macroscopic dynamics of the body, the simulation of physical interaction also requires modeling the microscopic mechanics of the contact interface. This challenge is epitomized by Vision-Based Tactile (ViTac) sensors such as GelSight [12] and DIGIT [60]. Simulating such sensors presents a unique dual challenge which requires solving both the mechanical contact problem to determine the elastomer shape and the optical rendering problem to generate the camera image. Existing simulators can be categorized into four paradigms based on their approach to this problem.

Graphics-based simulators such as TACTO [61] and Tactile Gym [62] prioritize speed by abstracting mechanics into geometric penetration depth. They utilize standard rendering pipelines to generate depth maps that mimic tactile readings. While highly efficient (> 100 Hz) and suitable for large-scale reinforcement learning, they lack a true force-deformation constitutive model, failing to capture complex mechanical effects like shear or friction-induced bulging. Similarly, FOTS [63] optimizes the rendering pipeline for speed but

relies on simplified shading approximations.

Data-driven and example-based approaches such as Taxim [64] avoid explicit physics solving by leveraging calibration data. Taxim constructs polynomial lookup tables mapping surface gradients to pixel intensities based on real-world examples. TacSL [65] provides a library for learning visuotactile skills, often using data-driven approximations to bridge the gap between simulation and reality. These methods achieve high perceptual fidelity for specific sensor configurations but often lack the generalizability to model novel interactions or unseen object geometries outside their calibration distribution.

Physics-based FEM approaches aim for mechanical fidelity. DiffTactile [66] integrates a differentiable FEM solver with a Neo-Hookean constitutive model, allowing for system identification via gradient descent. TwinTac [67] adopts a digital twin approach, synchronizing real-world sensor data with FEM simulations to train a high-fidelity virtual counterpart. While these methods provide the most accurate representation of contact mechanics, they are bound by the computational latency of iterative solvers ($O(N^2)$), limiting their throughput for the massive-scale data generation required by modern robot learning.

Finally, neural and generative approaches explore bypassing explicit simulation pipelines. NeuralFeels [68] utilizes neural fields to encode contact geometry for online perception and SLAM, while generative methods like Sim2Surf [69] use diffusion models or GANs to hallucinate tactile images from visual cues. While promising for perception tasks, these end-to-end generative models often lack the explicit physical constraints required to serve as a robust ground truth for control policy training.

2.6. Rationale for Method Selection

The selection of specific abstraction architectures for Part A and Part B is driven by the distinct computational constraints and inductive biases required for their respective tasks.

For Part A, the primary constraint is solver compatibility. Modern rein-

forcement learning environments and control stacks rely on rigid-body physics engines that optimize collision detection for articulated kinematic chains. A generic dimensionality reduction would destroy this kinematic structure, rendering the model incompatible with standard contact solvers. Therefore, we implement the Surrogate Model, which explicitly maps continuum deformation into a Virtual Kinematic Chain. This preserves compatibility with fast LCP solvers while capturing the compliance of the soft body through learned joint stiffness parameters.

For Part B, the primary constraint is geometric generalization. A tactile sensor must accurately predict deformation for arbitrary, unseen object geometries that were not present in the training set. Standard deep learning models like CNNs or MLPs tend to overfit to the global object shapes seen during training. To address this, we implement the Neural Physics Engine (NPE) using Graph Neural Networks (GNNs). This architecture imposes a topological inductive bias, forcing the model to learn the local, node-to-node rules of force propagation. By learning the local physics rather than global shapes, the GNN-based NPE ensures that the simulation generalizes effectively to novel contact interactions.

: Modeling Deformable Body Dynamics

Chapter 3.
Analytic Model of Deformable Robot

3.1. Motivation

This chapter established an analytical modeling representation for modeling deformable systems. This research selected a deformable manipulator as its testbed, specifically focusing on an even more difficult class of these systems: a collapsible manipulator based on origami principles. The unique, nonlinear folding mechanics of origami structures present a significant modeling challenge, making them an ideal subject for investigating the limits of analytical tractability and motivating the development of a simplified, yet effective, kinematic model for real-time control. This work is based on the previous publication, *"Model-based control of proprioceptive origami actuators for pneumatic manipulation"* published in the International Journal of Robotics Research in 2025 [35].

Origami-inspired robots have emerged as a novel and versatile category within the field of soft robotics, leveraging the principles of traditional Japanese paper folding to create simple and lightweight structures capable of complex three-dimensional movements [70–72]. This innovative approach has found applications across diverse sectors, including manufacturing [73], biomedical devices [74], and space exploration [75], where their unique properties are particularly beneficial. The inherent compliance and adaptability of origami-utilized soft robots also make them ideal for safe interaction with humans and delicate surroundings [76].

Despite these advantages, the control of origami robots poses significant challenges due to the complexities involved in accurate modeling of their kinematics and dynamics [51, 77]. The nonlinear nature of folding mechanisms, coupled with variations in material properties, complicates the prediction and

control of these systems [78, 79]. Researchers have attempted to overcome these hurdles through various model-based control strategies, including finite element analysis (FEA) [80], lumped parameter models [81], and machine learning techniques [82]. Furthermore, geometric and strain energy-based modeling approaches have been widely employed to describe origami structures [83, 84]. These methods have been particularly effective in capturing the deformation characteristics of Yoshimura-patterned actuators, as seen in prior studies [85]. However, these approaches often assume simplified mechanical approximations that limit their applicability to real-time control. Moreover, inverse kinematics (IK) solutions in origami-inspired manipulators are typically formulated either through rigid-link approximations [85] or iterative numerical optimization [86], which increase the computational complexity.

To address issues in heavy computation and real-time state estimation, researchers have explored the use of proprioceptive soft actuators capable of detecting their own internal states without adding external sensors. Specifically in soft robots, these systems are designed to detect the deformation or motion states of the actuators directly, providing an alternative to relying solely on modeling [87–89]. Such proprioceptive actuators have self-sensing mechanisms, which typically employ resistive [90–93], capacitive [94, 95], or optical [96–98] sensors to measure strain, orientation, or bending curvature. Although this self-sensing capability offers unique advantages, fabrication can be challenging. Accurate measurement of the configuration of a proprioceptive soft manipulator often requires multiple sensors, leading to difficulties in fabrication and calibration. These requirements can result in a system that is bulky and complex, with a need for a wide sensing range and robustness against noise [76, 99].

This study aims to tackle the dual challenges of computational complexity in model-based control methods and the difficulties in precise configuration estimation of soft origami manipulators. The approach encompasses the development of a highly functional soft manipulator utilizing solely origami structures, coupled with the derivation of a simplified kinematic model. The

work begins with the design of modularized, pneumatically driven bellow-type actuators incorporating an origami cylinder structure with an embedded self-sensing mechanism at each module. The use of cylindrical origami patterns, such as the Yoshimura or Kresling design, brings two principal benefits which are the intrinsic functionality of the structure itself and a significant simplification in the kinematics of motion [100, 101]. The origami cylinders are characterized by elastic properties that enable axial collapse along predefined folding lines, akin to a spring mechanism [102–104]. This feature allows for broad, compliant movements suited for various robotic systems, such as grippers [105, 106], manipulators [86, 107], and locomotive robots [108, 109]. Furthermore, when pneumatically powered, origami cylinders provide a large range of motion, significant force, and ease of reconfiguration, increasing their versatility [110–112]. However, despite the uniqueness of the origami cylinder, few methods are practically available to estimate its kinematics when actuated pneumatically. Much of the research analyzes the origami cylinder in a static manner, using non-linear FEA [113], post-buckling analysis [114], and numerical simulations [115, 116] that are computationally expensive for dynamic scenarios.

Therefore, a simplified kinematic model for the soft manipulator is proposed. The main idea of this model is to incorporate the essential assumptions and features from prior works to simplify the structure's analysis [83, 103, 117]. The proposed model establishes a mapping between the actuator's 3D configuration, the input signals, and any applied external loads. This allows for a reduction in the complexity of the required sensor system to a single, compact length-sensing unit. By integrating this simplified model with the actuator's proprioceptive characteristics, this approach effectively controls the manipulator in a synergistic, model-based, and sensor-informed manner.

In summary, this study presents three main contributions. First, a modularized pneumatic actuator is designed and fabricated using the Yoshimura origami cylinder, featuring proprioceptive functions to measure its configuration. These modules are integrated into an extended soft manipulator system.

Second, a simplified kinematic model for the origami cylinder is proposed, enabling accurate pose estimation in conjunction with its self-sensing mechanisms. This derived model uses a minimal proprioceptive sensing mechanism, simplifying the control strategy while maintaining high performance. Finally, a continuum robot made of multiple actuation units called origami cylinder modules (OCMs) is demonstrated in various applications, such as object picking, manipulation, and path-planning tasks, highlighting their potential in practical use cases.

3.2. Robot design

This section describes the proposed design of a soft manipulator that utilizes pneumatically actuated origami actuators, as depicted in Figure 3.1. The manipulator was constructed using multiple Origami Cylinder Modules (OCMs), each exhibiting the characteristic properties of the Yoshimura cylinder to enable both linear and bending deformation. The fabrication process for the OCM, which uses polyethylene terephthalate (PET) films and polypropylene (PP) sealing tapes, is briefly illustrated in Figure 3.2-(A).

Figure 3.2-(B) presents the design of the OCM and its integrated sensing mechanism. A Hall effect sensor (SS41, Honeywell) and a neodymium magnet were installed within the hollow air chamber of the cylinder for the purpose of detecting longitudinal displacement. This sensor module was designed to be compact and not to interfere with the existing structure, providing a linear response corresponding to the distance between the sensor and the magnet. Each OCM can actively contract and extend within a range of -60% to 60% of its 40 mm initial length and can bend up to 1.5 radians. A single OCM weighs approximately 25 grams, a lightness attributable primarily to its thin PET film base material, with most of the weight coming from the 3D printed components.

The complete continuum robot is composed of extended OCMs that are arranged in parallel, as shown in Figure 3.1. To facilitate a more straightforward kinematic analysis, the design adopts a piecewise constant curvature

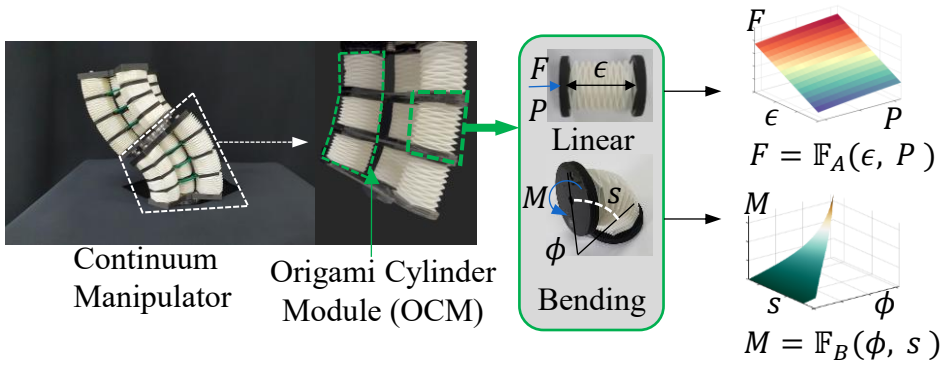


Figure 3.1: Continuum robot, comprising two segments of origami cylinder modules (OCMs). The OCM is capable of generating both linear and bending motions. The characteristics of these two distinct deformation modes decoupled and analyzed.

(PCC) approach, a common strategy for continuum robots suggested in previous studies [57, 80]. In this configuration, the OCMs were connected by a radial frame, as shown in Figure 3.1, to establish a uniform curvature across each segment of the robot. The design incorporates four extended OCMs. Three of these were strategically positioned in an equilateral arrangement, which was essential for generating the actuation motions of the module. The fourth cylinder, located at the centroid of this triangle, serves as a structural backbone that adds stiffness to the manipulator [118] and encloses connecting components like electric wires and pneumatic tubes.

3.3. Modeling the Origami Cylinder Module

The Origami Cylinder Module (OCM) presented in this work is composed of a cylindrical chamber fabricated with the Yoshimura pattern. While the Yoshimura cylinder is generally known as a volumetric origami structure that is rigid when properly folded [119], it also exhibits the capacity to deform under specific conditions if constructed from materials with low stiffness [120, 121]. The deformation of the cylinder along its crease lines and thin facets occurs as a result of axial compression, extension, and bending [122]. Furthermore,

the Yoshimura cylinder possesses considerable torsional stiffness, a property that allows its twisting motion to be neglected in the subsequent analysis [86].

The primary objective of the modeling is to analyze the kinematics of the Yoshimura cylinder based on its volumetric origami characteristics. Given its linear volume reduction, the axial strain, denoted as ϵ, can be defined as the ratio of the change in axial length ($l - l_0$) to the original length (l_0).

$$\epsilon = \frac{l - l_0}{l_0}. \tag{3.1}$$

The initial length l_0 is a design parameter determined when the origami pattern is created. The Yoshimura cylinder is also capable of bending, which results in a bending angle ϕ, a curvature κ, and a length of the neutral axis for bending s. Therefore, the overall configuration C of the actuator includes these bending variables alongside the axial strain ϵ, as illustrated in Figure 3.1. The controllable inputs to the system are the internal pressure of the compressed air P and the external load τ, where τ accounts for both force F and moment M. The aim of this chapter is to derive the relationship between these inputs and the resulting configuration, which is expressed as the kinematic function f_{OCM}.

$$\tau_{ext} = f_{OCM}(P, C), \tag{3.2}$$

In this function, C represents the geometric information of the Yoshimura cylinder that determines its configuration, as shown in Figure 3.3-(A).

3.3.1 Design Parameters of the Yoshimura Cylinder

The kinematic behavior of the Yoshimura cylinder, which is shown in Figure 3.3-(A) with its characteristic repeated patterns, is affected by several design parameters. The smallest foldable unit within these patterns is identified as a single layer, constructed by the tessellation of identical isosceles triangles. For the OCM design, it is assumed that all N layers are repeated and experience uniform deformations. This single layer therefore plays a crucial role in determining both the range of motion and the overall configuration of

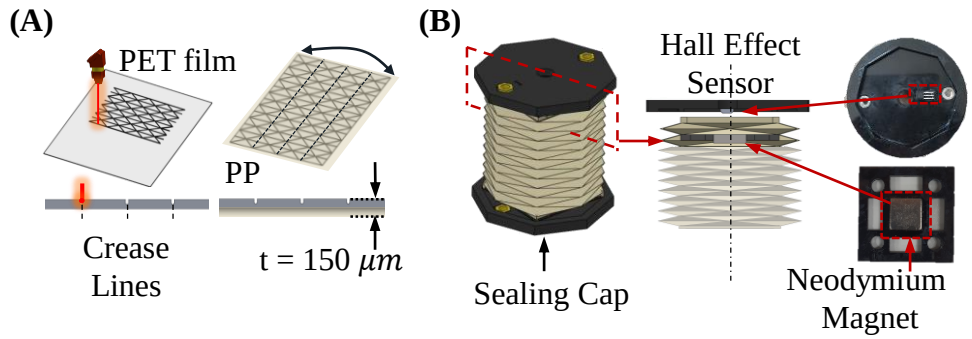


Figure 3.2: (A) Fabrication of a pneumatic chamber based on the Yoshimura cylinder pattern. This pattern was engraved on a 100 μm polyethylene terephthalate (PET) film using the laser CNC machine (Trotec Speedy 300, Trotec) and sealed with 50 μm polypropylene (PP) tape to create an airtight chamber. The cylindrical shape was formed, and then the Yoshimura origami pattern was folded. (B) 3D-printed air sealing caps and neodymium magnet located inside the folded Yoshimura pattern. A hall-effect sensor and a neodymium magnet were used to measure the displacement of a single layer of the actuator.

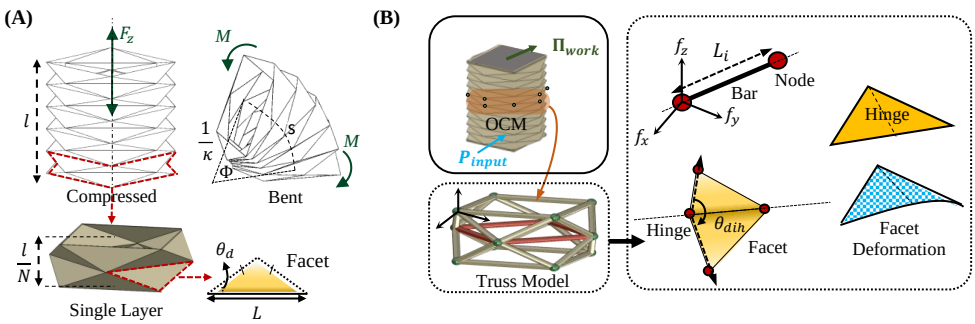


Figure 3.3: (A) Configuration of the Yoshimura origami cylinder and the definition of a single-layer. (B) Yoshimura origami cylinder treated as energy-conserving, with each vertex represented as a node and each edge as an elastic bar element with a single DOF. Rotational springs simulate hinge deformations, accounting for the nonlinear facet behavior.

the actuator. Each single layer is composed of an isosceles triangle defined by

two parameters which are the length of its base side (L) and the angle of its side (θ_d), as depicted in Figure 3.3-(A). Table 3.1 provides a summary of the design parameters that are considered in the analysis.

Design Parameters	L	30 mm
	l_0	40 mm
	θ_d	$\pi/8$ rad
	N	10
	n_b	40
	n_h	32
	w	13 g
Fitting Coefficients	k_b	1.3 N/m
	k_h	0.1 N/m
Manipulator	w_M	238 g
	d	60 mm
	s_c	210 mm

Table 3.1: Design parameters and specifications.

3.3.2 Assumptions for Pattern Analysis on the Yoshimura Origami Cylinder

Three assumptions are made to simplify the characterization of the actuator. First, the actuator is considered to be a conservative system, which allows the application of the virtual work principle to relate the external force to the configuration of the actuator [123]. This approach facilitates a straightforward relationship between the applied forces and the resulting deformations. Second, a uniform stress distribution across the entire cylindrical structure is assumed, which leads to consistent deformation across all layers, whether through linear compression, extension, or bending. Given that the cylinder is composed of repeated single layers, the entire structure can be conceptualized as a stack of multiple layers, as shown in Figure 3.3-(A). Consequently, the kinematics of this single layer can be uniformly applied to the entire structure. Lastly, for analytical simplicity, the cylinder is simplified as a truss structure, serving as an equivalent model.

As illustrated in Figure 3.4-(A), a single layer forms a polyhedron consisting of 16 isosceles triangles, which creates a closed structure with 12 vertices

and 40 edges. Figure 3.3-(B) depicts this abstraction of the Yoshimura cylinder model, where the vertices and edges are represented as nodes and elastic bar elements, respectively. Each edge is modeled as an elastic bar with a single degree of freedom (DOF) that can store strain energy through deformation. Additionally, rotational springs are used to simulate the hinges of the origami to capture the elastic deformations along the crease lines. The potential energy was calculated based on the amount of rotation between the neighboring facets that share each hinge [124]. Moreover, due to the nonlinear deformation experienced by the Yoshimura cylinder as a result of its rigidity, previous studies dealing with volumetric origami structures have incorporated additional elements to account for facet deformation [83, 103, 115]. Following these results, some basic features were also adopted in the simplified model, as shown in Figure 3.4-(A). The figure details the arrangement of 12 nodes ($a_{11} \sim a_{34}$) and an additional four nodes ($m_1 \sim m_4$) added at the midpoint of the horizontal edges to simulate facet deformation. The truss model employs the properties of virtual materials, with two defined stiffness coefficients, k_b and k_h, representing the axial and torsional resistance, respectively, to accurately reflect the kinematics of the actual structure. Based on the first assumption that the actuator behaves as a conservative system, the total potential energy stored in the origami cylinder can be expressed as follows.

$$\Pi_{total} = \Pi_{bar} + \Pi_{hinge} - \Pi_{work}, \tag{3.3}$$

Here, each term represents a different contribution to the system's energy.

- Π_{bar}: The strain energy of the bar elements accounts for the elastic deformation along the edges of the origami structure. This strain energy is calculated based on the elongation or compression of the bar elements, which is directly related to the nodal displacement and the geometric configuration of the Yoshimura cylinder.

- Π_{hinge}: The rotational energy stored in the crease lines reflects the energy required to rotate the facets relative to each other. This is determined by

the change in the dihedral angles between the adjacent facets, computed from the nodal positions and the geometric constraints of the pattern.

- Π_{work}: The work done by the applied pressure accounts for the external force applied to the structure. The work is calculated as the product of the applied pressure and the change in volume as the cylinder deforms.

The calculation of the total potential energy also incorporates the principle of virtual work during deformation, which accounts for changes in the relative positions of the nodes and the dihedral angles between facet pairs.

Given the assumption of uniform deformation across the structure, the analysis of potential energy focuses on determining the nodal displacements within a single layer. The geometric constraints inherent in the Yoshimura pattern were used to calculate the position and deformation of each node by solving a system of multi-variable equations.

To clarify further, the first segment of the kinematic derivation focuses on the axial deformation of the Yoshimura cylinder, which is a purely linear deformation, as depicted in Figure 3.3-(A). The second segment of the derivation focuses on bending deformation, which can occur simultaneously with axial deformation in the Yoshimura structure. To ensure a comprehensive analysis, these two types of motion are treated in separate analytical processes, allowing a detailed examination of each mode of deformation.

3.3.3 Kinematics Modeling of the Yoshimura Cylinder

Axial Kinematics The axial deformation kinematics of a single layer in the Yoshimura cylinder are determined by correlating the axial force (F_z) with the input pressure (P) and the strain (ϵ), as represented in the following function.

$$F_z = \mathbb{F}_A(\epsilon, P). \tag{3.4}$$

Here, $\mathbb{F}_A$ is a function derived from the three potential energy components outlined in Equation 3.3. The potential energy stored within the bar elements,

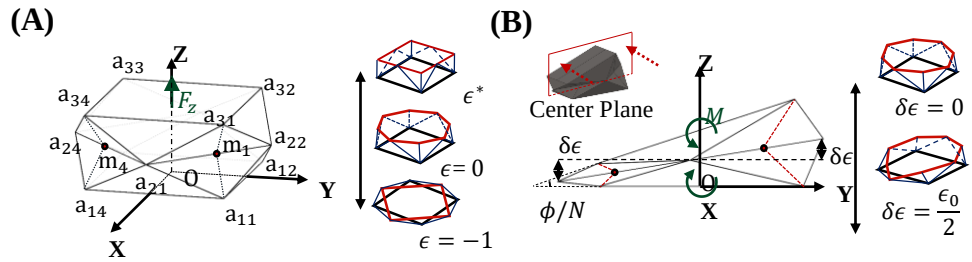


Figure 3.4: (A) Nodes indication of the single layer and linear deformation when axial force (F_z) applied (left) and the brief deformation configuration according to the strain ϵ (right). By the definition, ϵ can vary from -1 to the bounded maximum value ϵ^* that is the geometric limit. (B) Bending configuration of the single layer when moment M is applied and with bending angle ϕ/N. The bending angle can be expressed with the parameter $\delta\epsilon$ and has a range from 0 to $\epsilon_0/2$.

Π^A_{bar}, is expressed as

$$\Pi^A_{bar}(\epsilon) = N \sum_{i=1}^{n_b} \frac{1}{2} k_b \left(L_i(\epsilon) - L_i(\epsilon_0)\right)^2, \tag{3.5}$$

where ϵ_0 represents the initial strain when the cylinder is unloaded and at a neutral state. This type of strain energy based analysis is a well-defined method as shown in prior works on the Yoshimura origami structure [83]. Due to the geometric constraints of the Yoshimura cylinder, the parameter ϵ is theoretically bounded in the range $\epsilon \in [-1, \epsilon^*]$, where ϵ^* is a value related to the height of the constituent isosceles triangle, $L \tan\theta_d$ (see Figure 3.3-(A)). As depicted in Figure 3.4-(A), there are a total of n_b bar members shown in a three-dimensional coordinate system. The length of the i^{th} member is a function of ϵ and is expressed as $L_i(\epsilon)$. Assuming a uniform load distribution across each i^{th} member, the potential energy is calculated with a constant stiffness per unit length k_b. The length of each bar element at a given strain (ϵ) can be calculated from the distance between each node, based on the geometric constraints of the Yoshimura cylinder.

The subsequent term corresponds to the potential energy of the rotational hinges, which arises from the variation in angle between the facets. This energy can be expressed as

$$\Pi^A_{hinge}(\epsilon) = N \sum_{i=1}^{n_h} \frac{1}{2} k_h L_i(\epsilon) |\theta_i(\epsilon) - \theta_i(\epsilon_0)|^2. \tag{3.6}$$

A dihedral angle, θ_{dih}, is defined as the angle between two adjacent facets in a single layer, as shown in Figure 3.3-(B). Within a single layer, there are a total of n_h dihedral angles that can be defined by the combinations of two neighboring facets connected at each hinge. Similarly to Equation 3.5, the strain energy resulting from this change in angle was considered for the folding lines. The coefficient representing the rotational stiffness per unit length is denoted by k_h and is multiplied by the length of each hinge member $L_i(\epsilon)$ when calculating the total potential energy.

The final component considers the change in volume due to the applied pressure (P) inside the actuator. The volume of a single layer under a given strain ϵ is denoted as $V_A(\epsilon)$ and can be calculated from the nodal positions. Hence, the virtual work resulting from the pressure P can be explicitly derived as

$$\Pi^A_{work}(\epsilon, P) = P\Delta V_A = P(V_A(\epsilon) - V_A(\epsilon_0)), \tag{3.7}$$

which is a term linearly proportional to the pressure P.

Ultimately, the three terms that comprise Equation 3.3 can be expressed using the design parameters (N, θ_d, L, and l_0) and two fitting parameters (k_b and k_h). Incorporating Equations 3.5, 3.6, and 3.7 into Equation 3.3 allows the total potential energy to be articulated as a function of strain (ϵ) and pressure (P) as follows.

$$\Pi^A(\epsilon, P) = \Pi^A_{bar}(\epsilon) + \Pi^A_{hinge}(\epsilon) - \Pi^A_{work}(\epsilon, P). \tag{3.8}$$

The axial force is equal to the derivative of the axial potential energy with respect to the variable ϵ, under the assumption of energy conservation. There-

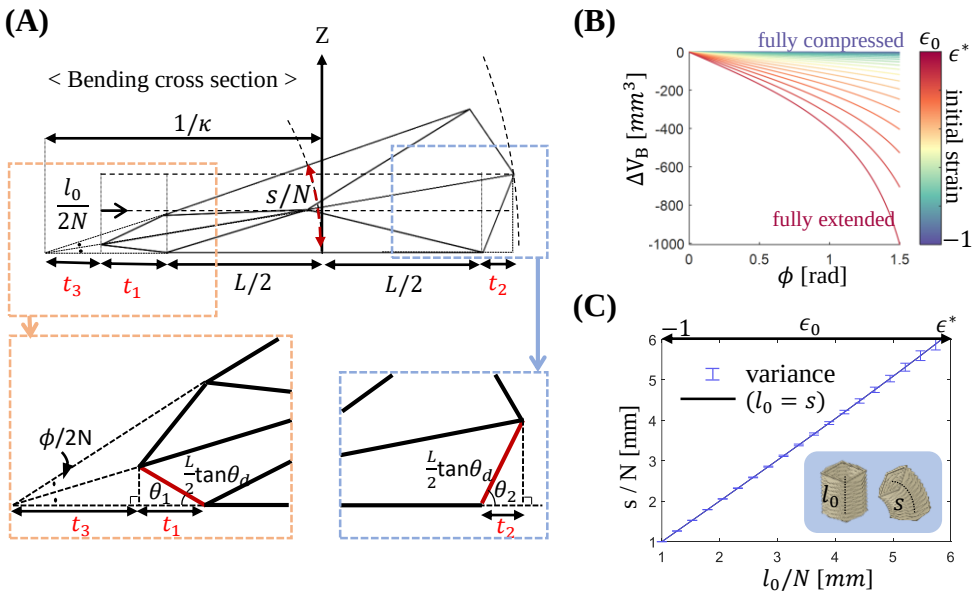


Figure 3.5: (A) Cross section of a single layer under pure bending. Additional variables (t_1, t_2, and t_3) are defined and depicted in the cross section of the single layer. These variables are used to derive the relation between the configuration parameters (ϕ, κ, and s) with the length l. (B) Numerical volume variation by the bending angle (ϕ/N) calculated for each initial strain (ϵ_0) from -1 to ϵ^*, as shown with the color code. The larger initial strain gives the larger variation. (C) Backbone length (s) of the single layer maintained the same as the initial value l_0, during bending. The solid line shows the numerical calculation of s compared with the initial length l_0/N of the single layer. The maximum deviance was 0.183 mm at l_0/N = 6 mm, where the strain was the maximum ϵ^*.

fore, the kinematics relating the net axial force (F_z) to the strain (ϵ) can be expressed as

$$F_z(\epsilon, P) = \frac{\partial \Pi^A(\epsilon, P)}{\partial \epsilon} := \mathbb{F}_A(\epsilon, P), \tag{3.9}$$

which maintains the form presented in Equation 3.4.

Bending Kinematics When a moment is applied to the Yoshimura cylinder, it undergoes bending deformation. The bending of a single layer is characterized by two configuration variables: its neutral axis length (s/N) and its

bending angle (ϕ/N). We define a function $\mathbb{F}_B$ that maps these configuration variables to the moment in a single layer:

$$M = \mathbb{F}_B(\phi/N, s/N). \tag{3.10}$$

Assuming identical deformation across all N layers, a moment M applied to the entire cylinder is distributed uniformly, such that each layer experiences the same moment and a bending angle of ϕ/N. Due to the cylinder's structural symmetry, each layer exhibits pure bending. Figure 3.4-(B) illustrates a single layer bending in the y-z plane, where the nodal displacements determine the bending angle ϕ/N. For example, nodes a_{22} and a_{24}, initially located at the same z-coordinate ($z = l_0/2N$, determined by the initial axial strain), move symmetrically during bending. Defining this z-coordinate deviation magnitude as δl, their new coordinates become $(l_0/2N+\delta l)$ and $(l_0/2N-\delta l)$, respectively. The variable δl, which itself depends on the bending angle ϕ/N and neutral axis length s/N, is used to concisely express all nodal positions. As depicted in Figure 3.5-(A), the geometric relationship between the configuration (ϕ, s) and δl can be derived by introducing additional angles and edge lengths for calculation. Two key angles in the y-z plane are defined as

$$\begin{aligned} \theta_1(\delta l, l_0) &= \sin^{-1}\left(\frac{l_0/2N - \delta l}{L\tan\theta_d}\right), \text{ and} \\ \theta_2(\delta l, l_0) &= \sin^{-1}\left(\frac{l_0/2N + \delta l}{L\tan\theta_d}\right). \end{aligned} \tag{3.11}$$

Using these angles, the additional edge lengths $t_1 \sim t_3$ can be expressed as

$$\begin{aligned} t_1(\delta l, l_0) &= L\tan\theta_d\cos\theta_1, \\ t_2(\delta l, l_0) &= L\tan\theta_d\cos\theta_2, \text{ and} \\ t_3(\delta l, l_0) &= \frac{L\tan\theta_d\sin\theta_1}{\tan(\phi/2N)}. \end{aligned} \tag{3.12}$$

These variables, in turn, allow for the derivation of the bending angle and curvature of the N-layered cylinder as

$$\phi(\delta l, l_0) = 2N \tan^{-1}\left(\frac{L \tan\theta_d \sin\theta_1}{t_3}\right), \tag{3.13}$$

$$R(\delta l, l_0) = t_1 + t_3 + L/2, \text{ and } \kappa(\delta l, l_0) = \frac{1}{R}. \tag{3.14}$$

Assuming uniform curvature across the layer, the neutral axis length s (i.e., the backbone length) is the arc length:

$$s(\delta l, l_0) = R\phi. \tag{3.15}$$

Based on Equations 3.13-3.15, all configuration parameters for a single layer can be expressed as functions of δl and l_0. The feasible range of the bending angle is also determined by geometric constraints related to the layer's initial length, l_0. For instance, the z-coordinate of node a_{24} cannot become negative, as this would imply a geometric collision with the adjacent layer. Therefore, the feasible range of δl is $0 < \delta l < l_0/2N$.

A notable result here is that both the initial length l_0 and the neutral axis length s are independent of the bending angle ϕ/N. We verified this characteristic numerically by calculating all nodal positions and confirming they satisfy Equation 3.15. As shown in Figure 3.5-(C), once the initial layer height l_0/N is determined by the axial deformation (later derived in the inverse mapping section), the neutral axis length s remains constant throughout bending, with a maximum numerical error of only 0.183 mm. This result indicates that bending can be decoupled from axial deformation; consequently, pressure and axial force primarily affect the longitudinal length l_0 (and thus s), while the moment primarily affects the bending angle ϕ.

The potential energy stored in the virtual bars and hinges can be calculated from the nodal coordinates, which are expressed in terms of δl, using the same geometric constraints as the axial case. As the nodes are displaced, the facets deform, altering the dihedral angles. The potential energy contributions from

the bars (Π^B_{bar}), hinges (Π^B_{hinge}), and the work done by pressure (Π^B_{work}) are formulated as

$$\Pi^B_{bar}(\delta l, l_0) = N \sum_{i=1}^{n_b} \frac{1}{2} k_b \left(L_i(\delta l + l_0) - L_i(l_0)\right)^2, \tag{3.16}$$

$$\begin{aligned} &\Pi^B_{hinge}(\delta l, l_0) = \\ &N \sum_{i=1}^{n_h} \frac{1}{2} k_h L_i(\delta l) \left|\theta_i(\delta l + l_0) - \theta_i(l_0)\right|^2, \text{ and} \end{aligned} \tag{3.17}$$

$$\Pi^B_{work}(\delta l, l_0) = P \Delta V_B(\delta l, l_0). \tag{3.18}$$

As with the axial case, these formulations sum over all relevant facet pairs and their corresponding dihedral angles. Here, ΔV_B denotes the volume change induced by bending. The layer's volume is calculated numerically via tetrahedral decomposition based on the nodal positions [125]. Figure 3.5-(B) plots the volume change ΔV_B as a function of the bending angle (ϕ/N) for several initial strains (ϵ_0). The feasible bending range (ϕ/N) increases with the initial layer length (l_0/N), allowing for more significant volume changes. Unlike the axial case where ΔV_A was formulated explicitly, the bending volume change ΔV_B is calculated numerically.

Eventually, the total potential energy stored in a single layer due to the bending moment is given by

$$\Pi^B(\delta\epsilon, \epsilon_0) = \Pi^B_{bar} + \Pi^B_{hinge} - \Pi^B_{work}. \tag{3.19}$$

For notational consistency with the axial kinematics, δl and l_0 are converted to strains $\delta\epsilon$ and ϵ_0 using Equation 3.1. The actuator's bending moment is then derived by differentiating Equation 3.19 with respect to the bending angle ϕ from Equation 3.13:

$$\tilde{M}(\delta\epsilon, \epsilon_0) = \frac{\partial \Pi^B(\delta\epsilon, \epsilon_0)}{\partial \phi(\delta\epsilon, \epsilon_0)}. \tag{3.20}$$

In Equation 3.20, derived moment was denoted as $\tilde{M}$. This distinction is

necessary because our target kinematic relationship, Equation 3.10, directly maps the configuration variables (ϕ, s) to the moment. Since the derived parameters are functions of the strains $(\delta\epsilon, \epsilon_0)$, an equivalence that connects the derived moment function can be plugged into the target form:

$$M(\phi, s) = \tilde{M}(\delta\epsilon, \epsilon_0), \text{ and } \mathbb{F}_B(\phi, s) := M(\phi, s). \tag{3.21}$$

This aligns the derived model with the initial formulation in Equation 3.10.

Integrated Kinematics Based on the geometric constraints and structural assumptions established previously, the kinematics for linear and bending deformations were decoupled into two distinct modes of configuration change. The axial height, l_0, or the initial strain, ϵ_0, was found to be determined exclusively by the axially applied force component F_z and the input pressure P. After this axial configuration is set, an applied moment M causes the cylinder to bend, which in turn alters configuration variables like the bending angle ϕ and the backbone length s. When an external force $\vec{f}_{ext} = [F_x, F_y, F_z]^T$ is applied to the tip of the Yoshimura cylinder, the axial force component F_z affects the axial elongation, whereas the lateral components F_x and F_y contribute to the generation of a moment. The resultant moment is considered only in the direction that causes the cylinder to bend, as represented by M in Equation 3.21. This is based on the assumption that twisting moments are negligible and the bending orientation is nonspecific due to the cylinder's symmetrical geometry. Consequently, the external load can be represented as

$$\tau_{ext} = \begin{bmatrix} F_z \\ M \end{bmatrix}. \tag{3.22}$$

Since both components of τ_{ext} have been derived in Equations 3.9 and 3.21, the kinematics can be integrated into a unified expression.

$$\tau_{ext} = \begin{bmatrix} \mathbb{F}_A(\epsilon, P) \\ \mathbb{F}_B(\phi, s) \end{bmatrix} = f_{OCM}(P, C). \tag{3.23}$$

This results in a comprehensive formulation consistent with the target kinematics model in Equation 3.2. The configuration parameter C therefore effectively encapsulates ϵ, ϕ, and s, providing a unified method to describe the kinematics of the Yoshimura cylinder.

3.3.4 Inverse Mapping using Lookup Table Methods

To efficiently compute the forward and inverse kinematics (FK/IK) of the origami manipulator for real-time control tasks, a data-driven approach that utilizes lookup tables (LUTs) was employed. Based on the forward kinematics mapping function in Equation 3.23, an inverse mapping function was derived to estimate the actuator's deformation when the input pressure and external loads are given.

The inverse kinematics of the OCM can be formulated with two inverse functions. The first, $\mathbb{F}_A^{\dagger}$, maps the axial force to the corresponding strain (ϵ), while the second, $\mathbb{F}_B^{\dagger}$, maps the bending moment to the bending configuration. These mappings are not analytically tractable due to the complex interaction between the structural elasticity and the pneumatic actuation. Instead, a dataset was precomputed through high-fidelity simulations, which created a structured LUT to allow for rapid inference during runtime.

For each OCM, a dataset of (F_z, P, ϵ) tuples was generated over fine intervals, enabling efficient interpolation for the inverse axial mapping.

$$\epsilon = \mathbb{F}_{\mathbb{A}}{}^{\dagger}(F_z, P). \tag{3.24}$$

Similarly, for the bending estimation, precomputed values for the bending

angles ϕ were stored to form the inverse bending map.

$$\phi = \mathbb{F}_{\mathbb{B}}{}^{\dagger}(M, s). \tag{3.25}$$

Using these inverse mapping functions, for a known external force and pressure, the corresponding strain and bending parameters can be directly retrieved or interpolated from the table. Furthermore, these functions made it possible to simplify the kinematic modeling of the full manipulator and to conduct model-based control in real time.

3.3.5 Model of the Origami Manipulator

The kinematic model for the Yoshimura cylinder, which was derived in the preceding sections, provides a foundational connection between the individual OCMs and the soft continuum manipulator constructed from them. While established forward kinematics (FK) and inverse kinematics (IK) for continuum robots are well developed [90, 126], their use is predicated on having prior knowledge of the mapping function that defines the relationship between an actuator's length and its corresponding input signals. Tendon-driven actuators, for example, often estimate their lengths from multiple encoder measurements, whereas pneumatic actuators lack the ability to directly measure this length information without the use of additional modeling or sensors. Therefore, the derived kinematic model (Equation 3.23) becomes an essential component for estimating the state of the continuum manipulator.

This continuum robot is designed with multiple segments, or trunks, where each is powered by three independently operated actuators, as shown in Figure 3.6-(A). The FK model for such a manipulator involves mapping the length of each actuator ($l_1 \sim l_3$) to the overall configuration of the manipulator [90,126]. As depicted in Figure 3.6-(B), a minimal set of three parameters is required to determine the configuration of a single trunk. These are the length of the backbone (S_c), the bending angle of the trunk (θ_c), and the direction

angle of the bending axis (ϕ_c), which are calculated as follows.

$$
\begin{aligned}
S_c &= \frac{1}{3}\left(l_1 + l_2 + l_3\right), \\
\theta_c &= \frac{2\sqrt{l_1^2 + l_2^2 + l_3^2 - l_1 l_2 - l_2 l_3 - l_1 l_3}}{3d}, \text{ and} \\
\phi_c &= \tan^{-1}\left(\frac{l_3 + l_2 - 2l_1}{\sqrt{3}(l_2 - l_3)}\right).
\end{aligned}
\tag{3.26}
$$

By assuming the PCC condition for each segment, which implies that no buckling or twisting occurs and that each trunk possesses a unique curvature, it becomes possible to uniquely determine the position and orientation of the tip at the moving platform [80]. This allows the special Euclidean group $SE(3)$ transformation from the base frame $\{B\}$ to the moving frame $\{S\}$ to be calculated using the configuration parameters detailed in Equation 3.26 (see Figure 3.6-(C)).

Spatial Configuration Estimation of the Origami Manipulator The origami manipulator is composed of multiple extended OCMs, as shown in Figure 3.1, with each undergoing complex deformation when subjected to applied forces and moments. In contrast to conventional rigid-link manipulators, the actuator lengths in pneumatic continuum manipulators are not directly controlled. This necessitates an inverse kinematics (IK) formulation to determine the configuration from the input pressures (P) when an external load (τ_{ext}) is applied.

Each extended OCM is constructed from three serially stacked OCMs arranged in a column, with three such columns positioned in parallel at 120° intervals. A consequence of the PCC assumption is that all OCMs within the same column share an identical internal pressure (P), an identical strain response, and an identical bending angle (ϕ_C), which ensures uniform deformation across the manipulator.

Following the inverse mapping formulation from Equations 3.24 and 3.25, the configuration of the origami manipulator is derived. This is accomplished

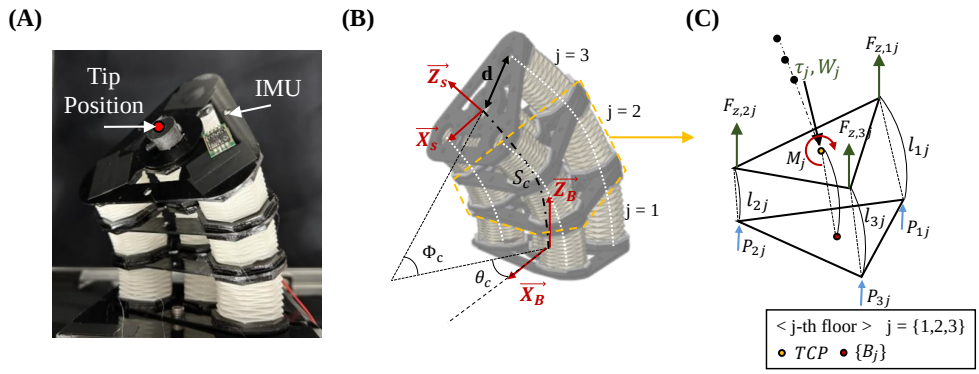


Figure 3.6: (A) Continuum origami manipulator with OCMs. (B) Configuration information of the origami manipulator, showing two different frames: the fixed base coordinate B_j and the moving frame S_j for each j^{th} parallel manipulator segment. The introduced manipulator in this paper has three segments, indicated by dashed lines. (C) Simplified drawing the j^{th} segment, with three pressure inputs (P_1 to P_3) independently controlling the three actuators. The mass load from neighboring segments (W_j) and the external loads (τ_j) are accounted for in the kinematics. The actuator lengths (l_{ij}) can be estimated using the derived inverse mapping functions.

by extending the kinematic model of a single OCM while also accounting for the propagation of loads throughout the stacked structure.

Force and Moment Equilibrium for the Origami Manipulator Each extended OCM is composed of three actuators positioned at 120° intervals around a central backbone. The deformation of each individual actuator is dependent on the applied external force (τ_{ext}), the internal pressure (P), and the constraints imposed by any neighboring modules.

As illustrated in Figure 3.6-(C), the total force equilibrium at the tool center point (TCP) for the j-th floor of the manipulator is given by the following equation.

$$F_{z1,j} + F_{z2,j} + F_{z3,j} = F_z + W_j + \sum_{k=j+1}^{3} W_k, \tag{3.27}$$

Here, F_z represents the external force applied at the top plate, while W_j

accounts for the gravitational weight contribution from the current and all upper floors. The moment equilibrium equations, which consider the external moments M_x and M_y, are formulated as

$$\begin{aligned} M_x &= d(F_{z2,j}\sin 120^\circ + F_{z3,j}\sin 240^\circ), \text{ and} \\ M_y &= d(F_{z1,j} - F_{z2,j}\cos 120^\circ - F_{z3,j}\cos 240^\circ). \end{aligned} \tag{3.28}$$

The torsional moment M_z is considered to be zero due to the high torsional stiffness of the origami structure [86]. The forces for the individual actuators can then be solved.

$$\begin{aligned} F_{z1,j} &= \frac{F_z + W_j + \sum_{k=j+1}^{3} W_k + \frac{M_y}{d} + \frac{M_x}{\sqrt{3}d}}{3}, \\ F_{z2,j} &= \frac{F_z + W_j + \sum_{k=j+1}^{3} W_k - \frac{2M_x}{\sqrt{3}d}}{3}, \text{ and} \\ F_{z3,j} &= \frac{F_z + W_j + \sum_{k=j+1}^{3} W_k - \frac{M_y}{d} + \frac{M_x}{\sqrt{3}d}}{3}. \end{aligned} \tag{3.29}$$

These equations define how the overall load distribution affects the individual external forces within a single OCM when it is integrated into the complete manipulator.

Bending and Strain Estimation Using LUTs To determine the general configuration of the manipulator, it is necessary to calculate the bending angle (ϕ_c) and the bending direction (θ_c) for each OCM. The total bending moment is first calculated.

$$M = \sqrt{M_x^2 + M_y^2}. \tag{3.30}$$

By applying the precomputed LUT for bending, the deformation parameters can be estimated.

$$\theta_C = {\mathbb{F}_{\mathbb{B}}}^{\dagger}(M, S_C), \text{ and } \phi_C = \tan^{-1}\left(\frac{M_y}{M_x}\right). \tag{3.31}$$

Here, S_C is the backbone length of a single extended OCM. Since all three OCMs in an extended OCM deform identically, the total bending angle of the manipulator is

$$\theta_{\text{total}} = 3\phi_C. \tag{3.32}$$

In a similar manner, the strain in each OCM is obtained from its corresponding inverse mapping function.

$$\epsilon_{ij} = \mathbb{F}_{\mathbb{A}}{}^{\dagger}(F_{z,ij}, P_i). \tag{3.33}$$

Actuator Length Estimation for the Origami Manipulator Once the strain values are retrieved from Equation 3.33, the individual actuator lengths for the j-th floor can be computed.

$$\begin{aligned} l_{1j} =& S_C + \epsilon_{1j} l_0 + d(\cos\theta_C - 1), \\ l_{2j} =& S_C + \epsilon_{2j} l_0 + d(\cos(\theta_C - 120^\circ) - 1), \text{ and} \\ l_{3j} =& S_C + \epsilon_{3j} l_0 + d(\cos(\theta_C - 240^\circ) - 1). \end{aligned} \tag{3.34}$$

These equations incorporate the combined effects of both axial and bending deformations. This ensures that the actuator length is correctly determined and can be applied in Equation 3.26 to estimate the resultant tip position and orientation.

Pressure-Controlled Configuration Estimation By extending the inverse mapping method from a single OCM to the entire manipulator, the combined effects of external forces and moments on the stacked OCMs can be considered. The configuration parameters, which include the actuator lengths (l_i), can be determined using the inverse function.

$$C = f_{OCM}^{\dagger}(P, \tau_{ext}). \tag{3.35}$$

Here, τ_{ext} represents the total external load, which includes gravity and any externally applied forces.

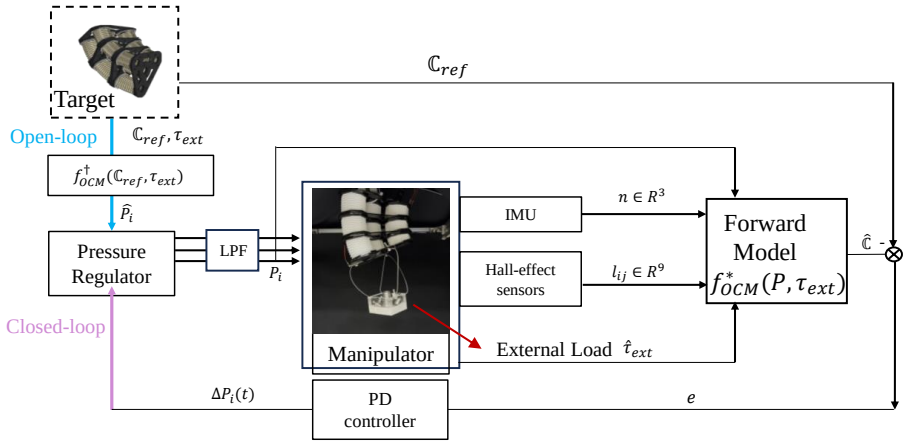


Figure 3.7: Control schematic of the origami manipulator. The closed-loop system controls the manipulator to reach a desired configuration (C) by adjusting the pressure regulators through a PD controller. The blue lines show the flow of the open-loop control, where the inverse kinematics model is used to determine the required pressure inputs without feedback.

For a manipulator that consists of multiple extended OCMs in series, the LUT method is applied iteratively, which accounts for the propagation of the load from the top plate downwards. The pressure required to maintain a specific configuration can then be calculated.

$$\hat{P} = f^{\dagger\dagger}_{OCM}(\tau_{ext}, C). \tag{3.36}$$

In this expression, $f^{\dagger\dagger}_{OCM}$ represents the inverse function that estimates the required pressure based on external forces and a target configuration. This function provides a pathway for implementing real-time pressure control to achieve a desired manipulator configuration. Figure 3.7 shows the proposed model-based control scheme. It uses the derived models as a plant for both open-loop and closed-loop control, which will be explained further in the next section.

3.3.6 Characterization of the Origami Cylinder Module

Experiments were conducted to examine the kinematics of the constructed OCMs and to validate the accuracy of the developed model. Since the derivation had decoupled the linear and bending kinematics, separate experiments were performed to measure the axial deformation and the pure bending of the OCMs.

For the axial deformation analysis, the normal force (F_z) and the actuator length (l) were measured to characterize the system's response under a constant pressure input and an external load. Figure 3.8-(A) illustrates the setup where force and strain data were collected using a tensile tester (34SC-1, Instron), while the pressure input (P) was controlled via a pressure regulator (ITV2030-212cl, SMC). The head of the tensile tester was programmed to move at a rate of 0.01 mm/s, which ensured a quasi-static condition. During these tests, the pressure was varied from 0 kPa to 30 kPa in 2 kPa increments, resulting in 16 distinct isobaric force-strain curve profiles. The initial length (l_0) of an individual actuator was set to 40 mm, with the strain spanning a range from -60% to 60%. This range was selected to ensure operational durability without causing permanent damage to the origami structure.

The setup for the bending experiment is illustrated in Figure 3.8-(B), wherein a tendon cable was affixed to the actuator's tip to apply a bending moment. A bending state was induced in the actuator by pulling the cable a certain distance from the neutral axis. Subsequently, moment-angle curves were obtained while the length of the neutral axis (s) was held constant. In the absence of external loads, the pressure within the origami chamber was adjusted to define the initial length (l_0) before bending commenced. The actuator's length was varied from a fully compressed state of $l = 20$ mm ($\epsilon = -0.6$) to a fully extended state of $l = 80$ mm ($\epsilon = 0.6$) in 3 mm increments, yielding 21 distinct bending profiles. The range of the bending angle was determined experimentally to stay within the feasible ROM of the actuator.

To analyze the motion, the bending sequence was video-recorded to track

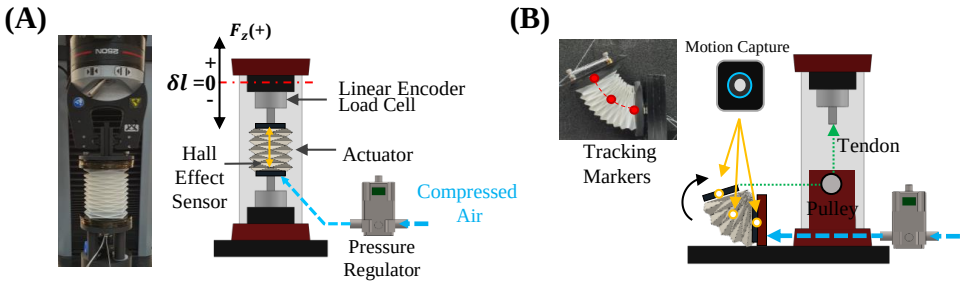


Figure 3.8: (A) Experimental setup for loading-unloading test using a tensile tester. The input pressure was controlled using a pressure regulator. (B) Experimental setup for bending experiment, using a pulley system to exert a moment on the origami cylinder. Three markers attached to the side of the OCM were tracked using a motion capture system.

markers placed on the actuator, while the curvature and bending angle were tracked with a motion capture system (OptiTrack, NaturalPoint). The error from the motion capture system and the accumulated propagation error are summarized in Table 3.2. The moment was calculated indirectly using the moment arm length and the force data acquired from the tensile tester.

The origami cylinder structure showed converging force-displacement curves under repetitive actuation, a behavior primarily attributed to stiffness degradation from wear along the crease lines, as has been reported in a prior study [127]. Accordingly, the origami cylinder specimen underwent 400 loading and unloading cycles to achieve a stable curve behavior. As shown in Figure 3.10-(A) and Figure 3.10-(B), both linear and bending actions produced convergent curves, indicating consistent performance during the different tests.

3.3.7 Control Task Implementation in the Soft Manipulator

Building upon the preceding sections, a series of experiments was performed to assess the performance of the manipulator integrated with OCMs. For manipulator control, three types of sensors were employed, and three origami cylinders were actuated independently by three pressure regulators. The orientation of the top moving plate $\{S\}$ with respect to the fixed base frame $\{B\}$

Metrics	Result
OCM characterization	$\mathbb{F}_A$: [e_F : 7.4% (0.77 N), e_{hyst} : 4.7% (0.48 N)] $\mathbb{F}_B$: [e_M : 18.0% (0.014 Nm), e_{hyst} : 5.9% (0.005 Nm)]
Origami manipulator ROM	ROM ($l_{\max}$ = 157.79, $\theta_{\max}$ = 0.788) RMSE(X, $\hat{X}$): 2.15% (3.45 mm), RMSE(q, $\hat{q}$): 2.67% (0.028 rad)
Trajectory tracking	Triangle: [OL: 15.30 mm, CL: 7.22 mm] Circle: [OL: 7.62 mm, CL: 5.54 mm]
Position control with the external load	Random 10 points: [OL: 14.29 mm, CL: 5.86 mm] Average rise time=3.48 s, settling time=7.32 s
Orientation regulation with dynamic external load	RMSE(q, $\hat{q}$): 3.15% (0.033 rad) Average rise time: 1.21 s, settling time: 2.55 s

* Calibration errors are included in the experiments conducted with the motion capture system (mean error: 0.122 mm 3D).

Table 3.2: Summary of modeling and control results across different experimental scenarios.

was determined using Inertial Measurement Units (IMUs) (ebimu-9dofv5-r3, E2box). Concurrently, Hall-effect sensors located within each chamber provided calibrated estimates for the lengths of the bent actuators. The combination of the IMU and Hall-effect sensors enabled the measurement of the frame $\{S\}$ orientation and the manipulator's backbone length (s_c).

The application of the PCC assumption for the continuum manipulator allowed for the determination of a unique position at the tip of the frame $\{S\}$, a method explored in prior research [126]. Throughout the experiments, both the position ($\vec{X}$) and the orientation (n_X, n_Y, and n_Z) of the manipulator's tip were systematically measured.

The following experiments were designed to evaluate the performance of the origami manipulator and to validate the accuracy of the derived model.

Range of Motion Test The initial experiment was designed to monitor the centroid position of the frame $\{S\}$ by employing various combinations of pressure inputs, in order to explore the accessible workspace of the designed manipulator. Each test iteration followed 16 discrete pressure steps, which mirrored the axial deformation experiments. Given that the manipulator features

three degrees of freedom, the total number of combinations was $16^3 = 4096$ points. The estimated positions derived from the forward kinematics ($\vec{X}$) were compared against the experimentally measured points ($\hat{X}$) using the root-mean-square error (RMSE). The orientation error was calculated in a similar manner by comparing the Euler angles (n_X, n_Y, and n_Z). The actual position and orientation of the tip were measured using the IMUs and Hall-effect sensors.

Trajectory Tracking Tasks The second experiment focused on trajectory tracking. Two time-dependent trajectories, a triangle and a circle, were used to provide references for open-loop control, as shown in Figure 3.9-(A). Once the target trajectories were predefined, the required actuation profiles for each actuator were obtained using the function from Equation 3.26. When the time variance of each required length is known, the input pressure profiles can be derived from the relationship between pressure input and configuration, as given by Equation 3.35. For comparative purposes, proportional-derivative (PD) control was also conducted using feedback from the embedded sensors (see Figure 3.7).

Position Control with External Load The final experiment involved position control using the kinematic model while an external load was applied. As illustrated in Figure 3.9-(B), a 250 g weight, which is nearly equivalent to the total mass of the manipulator ($W_m = 230\ g$), was attached to the manipulator's tip during actuation to evaluate its performance under payload. A set of 10 random coordinates within the manipulator's range of motion was selected and scheduled for continuous tracking. The input pressure profiles for the three actuators were designed to guide the tip position of the manipulator to follow these selected points. To verify the accuracy of the programmed model in reaching the target positions, a settling time of 10 seconds was allowed at each point to ensure the system reached a steady state.

Adaptation to Dynamic Loading To evaluate the force feedback and configuration control capabilities of the origami manipulator, a dynamic loading scenario was tested. The external force was estimated using the model derived in Equation 3.22 for each OCM that composes the manipulator. During the estimation of the external force, a specific configuration, such as the tip position or the orientation of the upper frame {S}, was actively controlled. The primary objective of this experiment was to maintain the predefined configuration while incremental weights were added to the system. As shown in the setup in Figure 3.9-(B), the external load was applied at the center of the frame {S}, and a closed-loop control scheme was implemented to achieve this task.

3.3.8 Various Applications

To highlight the versatility and potential practical applications of the proposed OCMs, this section presents example configurations and their corresponding operations.

Origami Gripper The OCMs can be utilized to construct an adaptive origami-based gripper, as shown in Figure 8-(A). This gripper leverages the flexibility and compliance of the OCMs to conform to various shapes and sizes of target objects. The combination of both axial and bending deformation allows the gripper to handle delicate objects while maintaining sufficient stiffness to apply the necessary grasping forces. Such a system holds potential for applications in areas that require delicate manipulation, for instance, handling fragile items in unstructured environments.

Origami Continuum Robot The OCMs can also be extended to form a two-segment continuum robot, as depicted in Figure 8-(B). By connecting multiple OCMs in series, the manipulator becomes capable of performing complex motions, such as bending and reaching into constrained spaces. The robot can also be augmented with an end-effector, such as a gripper for object manipulation, making it well-suited for applications in inspection, exploration,

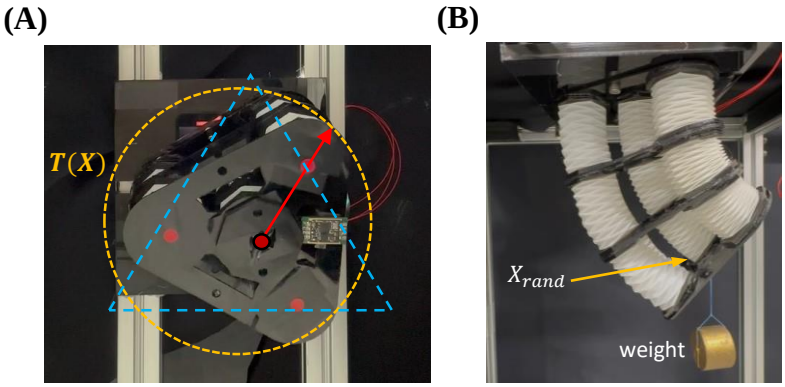


Figure 3.9: (A) Circular and triangular trajectories to be tracked by the manipulator with position control. (B) Position control of random tip position control with a vertical load applied at the tip.

or soft robotic navigation within unstructured environments. The derivation of the kinematics for this two-segment robot is similar to the process described for the single-segment manipulator in Section 3.5, but it additionally accounts for the interaction force and moment from the neighboring segment.

3.4. Results

3.4.1 Validation of the OCM kinematics model

The experiments were designed to validate the models presented in Equations 3.9 and 3.21. Initially, the linear and bending deformations were characterized without any pressure input, with the results shown in Figure 3.10-(A) and (B). These plots show the characterization curves generated from the raw experimental data. In Figure 3.10-(A), the axial force F_z is plotted against the OCM's calculated strain ϵ. The strain range was conservatively limited to $\epsilon \in [-0.6, 0.6]$ to prevent extreme deformation or damage. This experiment revealed a noticeable hysteresis between the loading and unloading curves, which is attributed to the hyperelastic nature of the origami cylinder. We define the hysteresis error as the percentage difference between the loading (F^+)

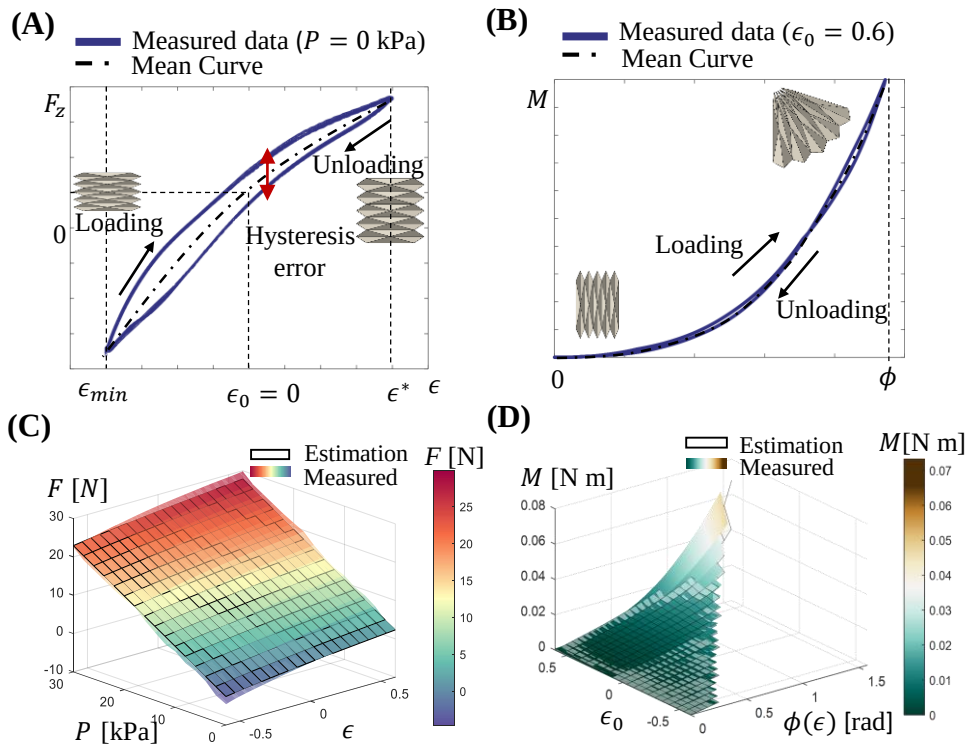


Figure 3.10: (A) Measured data of the cyclic test for linear deformation of the OCMs, showing the force-strain curves when no input pressure was applied. (B) Measured data from the cyclic test for bending deformation, with an initial strain (ϵ_0) of 0.6. (C) Grid plot visualizing the 3D relationship of force, pressure, and strain. Experiments were conducted with 16 different pressure levels with an increment of 2 kPa, within the strain range of -60% to 60%. Solid lines represent the forces estimated by the model, while the filled surfaces show the measured data. (D) Moment and bending angle curves for 16 different initial strains (ϵ_0) of the single layer. The initial strains (ϵ_0) were adjusted through 16 steps of pressure inputs, consistent with the linear deformation experiments. Solid lines indicate the moments estimated by the model, while the filled surfaces show the measured data.

and unloading (F^-) forces relative to the maximum observed force:

$$e_{hyst} = \frac{F_z^+ - F_z^-}{\max(F_z)} \times 100\ (\%). \tag{3.37}$$

To compare with the derived models, which do not account for hysteresis, the mean of the loading and unloading data for both force (F_z) and moment (M) were used:

$$\begin{aligned} \bar{F}_z &= \frac{F_z^+ + F_z^-}{2}, \\ \text{and } \bar{M} &= \frac{M^+ + M^-}{2}, \text{ respectively.} \end{aligned} \tag{3.38}$$

For the bending experiment, the achievable range of the bending angle depended on the cylinder's initial strain, which aligns with the geometric constraints in Equation 3.13. For example, Figure 3.10-(B) shows a bending test where the range was limited to 1.5 radians to prevent damage.

Figure 3.10-(C) presents the extended characterization curves for linear deformation under various pressure inputs. The plot shows all measured data points, each corresponding to a specific axial force (F_z), strain (ϵ), and pressure (P). The values predicted by our model (Equation 3.9) are overlaid on the measured data to show the model's accuracy. While our model can be represented as a continuous surface, the experimental data is shown as a color-filled grid to reflect the discrete pressure levels used during the tests. The estimated force, $\hat{F}_z$, is visualized as this continuous, gradient-colored surface in Figure 3.10-(C). The force estimation error, calculated as

$$e_F = \left| \frac{\hat{F}_z - F_z}{F_z} \right| \times 100 \ (\%), \tag{3.39}$$

was 7.4% on average, corresponding to a mean absolute error of 0.77 N. The error tended to increase at pressures above 20 kPa, reaching a maximum of 13.7%. In addition, the average hysteresis error for the axial force was 4.7%.

For the bending case, Figure 3.10-(D) shows the extended characterization curves measured at various initial strains. A key observation is that the feasible bending range is a function of the initial strain: a larger initial strain permits a wider range of bending angles. This experimental result aligns with our model, which incorporates geometric constraints dependent on the initial strain ϵ_0. Therefore, the pre-bending initial strain can serve as a reliable indicator of the

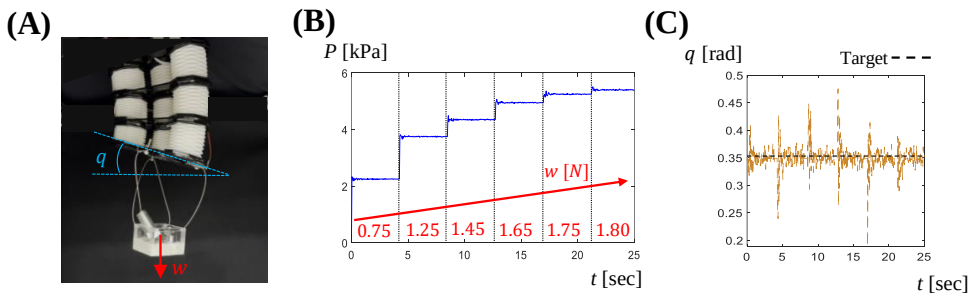


Figure 3.11: (A) Experimental setup for position holding as the load increased over time. The manipulator maintained the target angle q throughout the experiment while the load increased from 0.75 N to 1.80 N over 25 seconds. (B) Pressure input profile as the external load increased. (C) Result of the quaternion angle during the task maintained by the manipulator around the target value.

allowable bending range. The moment-related errors were calculated using the same method as the axial case. The average moment estimation error was 18%, corresponding to a mean absolute error of 0.014 Nm. Furthermore, the average hysteresis error was 5.9%. The estimation error tended to increase with larger initial strains, which corresponded to a wider ROM for the bending angle.

3.4.2 Task Implementation Results

The proposed three-DOF manipulator was actuated using all possible combinations of pressure inputs to evaluate its performance. The experimental results were then compared against the analytical model's predictions from Equations 3.26 and 3.35. First, the manipulator's reachable workspace was evaluated, as shown in Figure 3.12-(A). For each combination of pressure inputs ($P_1 \sim P_3$), the position and orientation errors between the measured data and the model's predictions were recorded. The manipulator operated within its expected range of motion (ROM), exhibiting an average position RMSE between the measured pose $\vec{X}$ and the predicted pose $\hat{X}$ of 8.2 mm. However, several outliers occurred at high-pressure inputs, where the maximum error

reached 24.5 mm.

To better contextualize the position error relative to the overall workspace, a normalized, ROM-based percentage error was calculated. First, the maximum workspace diagonal, l_{max}, was calculated as

$$l_{max} = \sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}, \tag{3.40}$$

where δ_x, δ_y, and δ_z represent the total workspace ranges along each axis. The absolute positional error, $|\vec{X} - \hat{X}|$, was then normalized as follows:

$$e_{\text{ROM,n}} = \frac{|\vec{X} - \hat{X}|}{l_{max}} \times 100\ (\%). \tag{3.41}$$

This normalized metric provides a more insightful measure of the manipulator's accuracy relative to its total workspace. For orientation errors, a quaternion-based metric was used instead of simple Euler angle RMSE to ensure a more robust and accurate measurement. The quaternion error was calculated as

$$e_q = 2 \cdot \arccos(|q_{\text{m}} \cdot q_{\text{e}}|), \tag{3.42}$$

where q_{m} and q_{e} are the quaternions for the measured and estimated orientations, and $\cdot$ denotes the dot product. This method provides a more holistic orientation error metric compared to analyzing individual vector components. Similarly, a normalized quaternion error was derived by dividing the error by the maximum possible bending angle, θ_{max}:

$$e_{q,\text{n}} = \frac{e_q}{\theta_{\text{max}}} \times 100\ (\%). \tag{3.43}$$

We observed that regions of extreme deformation contributed most significantly to the overall average position error. To better assess the model's precision in regions of moderate deformation, the errors within a reduced ROM was analyzed, defined by $x, y \in [-50, 50]$ mm and $z \in [70, 140]$ mm.

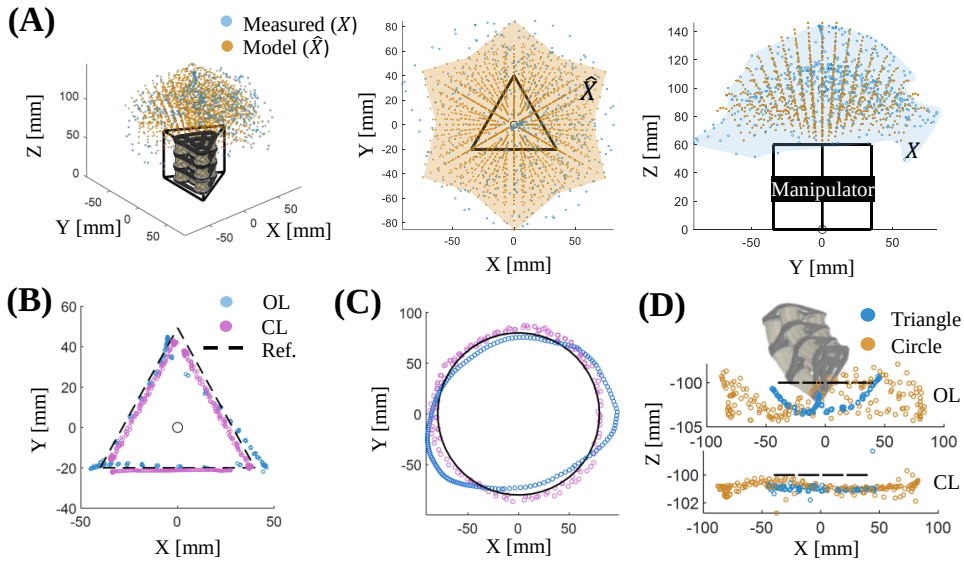


Figure 3.12: (A) Workspace analysis with all combinations of pressure inputs, comparing the model predictions with the actual measurements. (B) Result of triangular trajectory tracking conducted on a $z = -100$ mm plane. (C) Result of circular trajectory tracking control performed on a $z = -100$ mm plane. (D) z-axis positions measured during both tracking tasks, comparing the model-based open-loop (OL) control with the closed-loop (CL) control that uses the embedded IMU and the Hall-effect sensors.

A summary of all experimental errors is provided in Table 3.2.

For the trajectory tracking tasks, two reference trajectories were designed on the x-y plane at a constant z coordinate, both of which were within the manipulator's measured ROM. These reference trajectories, denoted $T(\vec{X})$, were defined as time-varying position functions. For the open-loop case, the required pressure inputs for each OCM were calculated from the reference trajectory using the inverse model (Equation 3.35), assuming zero external load. This model-only strategy constitutes our open-loop control (OL). In addition to the open-loop validation, a closed-loop control strategy that utilized feedback from the integrated IMU and Hall-effect sensors was also tested. For these evaluations, triangular and circular reference trajectories were set. For the triangular trajectory, the average position RMSE was 15.30 mm for open-loop

control and 7.22 mm for closed-loop control. For the circular trajectory, the corresponding errors were 7.62 mm (open-loop) and 5.54 mm (closed-loop). The results for both trajectories are plotted in Figure 3.12-(B), (C), and (D). The open-loop controller completed each cycle in a consistent 10 seconds, as its timing was determined solely by the programmed model. In contrast, the closed-loop controller's completion time varied per trial due to the PD error correction, typically taking approximately 5 seconds for the triangular path and 10 seconds for the circular path.

In the position control task, the manipulator successfully reached 10 randomly selected target points while under an external load. The target points and measured trajectories are shown in Figure 3.13-(A), with the corresponding coordinate errors plotted in Figure 3.13-(B). Allowing 10 seconds to reach each target, the resulting average position errors were 5.6 mm (x), 4.7 mm (y), and 7.3 mm (z), yielding a mean 3D spatial error of 5.86 mm. To further assess performance, the rise and settling times at each target were measured, which averaged 3.4 s and 7.3 s, respectively. For comparison, the open-loop model alone resulted in a significantly higher average distance error of 14.29 mm, similar to the trajectory tracking results.

In the force control task, the manipulator maintained its pose while an external load was gradually increased. We observed an orientation regulation error (q) of 3.15%, which corresponds to 0.033 rad. The system's dynamic response was faster than in the position control task, with average rise and settling times of 1.21 s and 2.55 s, respectively.

The stability and accuracy of the manipulator were evaluated through a position-holding experiment under varying external loads, as depicted in Figure 3.11. In this setup, the manipulator was tasked with maintaining a constant target angle, q, while an external load was gradually increased. As shown in Figure 3.11-(A), the load increased from 0.75 N to 1.80 N over a period of 25 seconds.

To counteract the increasing external force, the system adjusted the pressure input to the manipulator, a process detailed in Figure 3.11-(B). This active

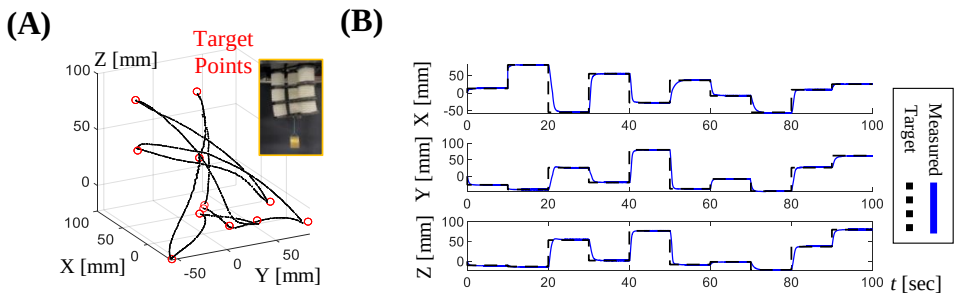


Figure 3.13: (A) 10 target points with random selections in red circles and the actual trajectories of the manipulator's tip. 10 seconds were allowed to reach each goal position, allowing the entire task completion time of 100 seconds. (B) Position control results in three axes, demonstrating the manipulator's performance in reaching the targets.

control mechanism was crucial for maintaining the desired position. The results of the experiment, displayed in Figure 3.11-(C), demonstrate the manipulator's ability to maintain the target quaternion angle accurately throughout the task, despite the significant increase in external load.

3.4.3 Applications

The soft gripper prototype, shown in Figure 3.14-(A), consists of three bending OCMs. Each OCM is fixed on one side, allowing it to bend toward the center of the gripper when pressurized. These OCMs are strategically oriented to converge at the gripper's center, enabling a grasping motion. The full system, as illustrated in Figure 3.14-(B), integrates this soft gripper at the end of the extended manipulator segments, forming a continuum robot. For precise control, an IMU is installed in each trunk segment to measure orientation, while a Hall-effect sensor is embedded within each OCM to track its motion.

The performance of the continuum robot is illustrated in Figure 3.15-(A) and (B). The results provides a visual demonstration of the origami gripper and the continuum robot carrying out a task, which involves picking up an object at a predetermined location. The task necessitated controlling the end-

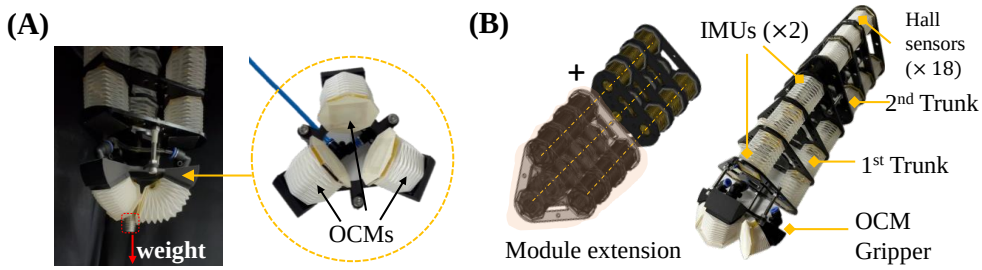


Figure 3.14: (A) Soft gripper prototype composed of three bending OCMs. One side of each OCM is fixed to realize to the module's bending motion when pressurized, with all three OCMs oriented toward the center of the gripper. (B) Extended manipulator segments as a continuum robot with the gripper installed at the end. IMUs are installed in each trunk segment, and a Hall-effect sensor is embedded in each OCM.

effector's orientation and position to prevent collisions with the object and to correctly align the gripper for a secure hold. This reaching procedure was accomplished in five consecutive steps to gradually approach the target object. Upon reaching the object, the gripper was activated to grasp it.

The system utilized six extended OCMs, whose backbone lengths were tracked by Hall-effect sensors. The orientations of each trunk were measured by two IMUs. The pre-set positions and orientations for each step were established before the task and managed by a PD control loop, as detailed in Figure 3.7. The position tracking data for the x, y, and z coordinates are presented in Figure 3.15-(C), while the measured orientations during the task are shown in Figure 3.15-(D). The continuum robot successfully reached and gripped the object in under 25 seconds.

3.5. Discussion

The experimental results verify the feasibility and the effectiveness of the proposed soft continuum manipulator, which leverages OCMs with proprioceptive functionality. Experiments on the linear and bending deformation modes of the OCM confirmed the accuracy of the model. Moreover, integrat-

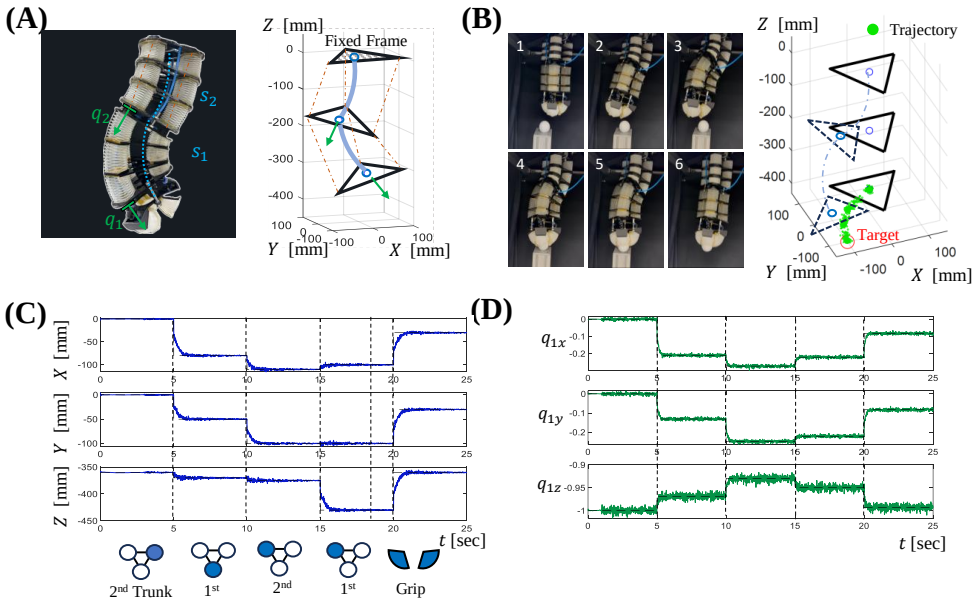


Figure 3.15: (A) Configuration estimation of two trunks using embedded proprioceptive sensors. (B) Object picking task using the continuum robot using the open-loop control. A total of five different steps were planned to reach the predefined object location and pick up and place the object in the end. (C) Position control result of the task implemented by the continuum robot (top) and five input commands during the task (bottom). (D) Orientation changes following the five input steps, measured by the IMU sensor on the first trunk.

ing proprioceptive OCMs into the soft continuum manipulator demonstrated the potential for practical applications.

The validation of the OCM model showed low errors in estimating both force and moment, indicating the possibility of practical implementation of the Yoshimura pattern-based origami actuator in real-world settings. The errors, however, rapidly increased under extreme conditions, such as at high pressures or strains, and hysteresis was also observed in both force and moment measurements. This may have been caused by several reasons, such as variations in the material properties and manufacturing tolerances from manual fabrication. While the model simplifies the complexity in the analysis that typically uses

FEA, it may not capture various nonlinear behaviors. Nevertheless, within a defined ROM for both OCMs and the manipulator, the model proved to be useful.

When assembled into a larger system, the OCMs demonstrated the capability of reaching diverse positions and making different orientations with precision. Although the average errors within the ROM tasks were modest, the outliers at high pressure inputs highlighted the kinematic nonlinearities, particularly over-actuation at extreme pressure levels, which affected the accuracy of the prediction model and sometimes violated the PCC assumptions. The trajectory tracking task showed the efficacy of the model in open-loop control, and the accuracy was even increased with closed-loop control, suggesting its potential to improve the performance in practical implementations. Notably, in tasks involving payloads, the manipulator was able to maintain the high accuracy, with decreased errors and relatively short settling times.

Table 3.3 compares our proposed origami-based soft pneumatic manipulator system with state-of-the-art systems (Cited Papers: [85, 90, 110, 112, 128–137]). Our approach introduces several distinct advantages derived from our novel modeling and design strategies. Unlike existing systems that utilize origami cylinders primarily as structural shells with separate actuators, such as pneumatic muscles or SMA wires, our Origami Cylinder Modules (OCMs) serve simultaneously as the structural framework and the pneumatic actuator chambers. This integrated design significantly reduces complexity and weight, enhancing the manipulator's overall agility and responsiveness.

Our simplified kinematic model explicitly incorporates internal pressure inputs, volumetric deformation, and virtual work, extending conventional strain-energy-based geometric approaches. By establishing explicit mapping functions based on extensive experimental data, our model enables efficient real-time forward and inverse kinematics computations. This modeling strategy provides scalability, seamlessly transitioning from single OCMs to modular manipulators and multi-segment continuum robots without requiring additional assumptions or rigid-link approximations.

Additionally, the integration of proprioceptive sensing mechanisms, specifically Hall-effect sensors for length measurement and IMUs for orientation, significantly enhances real-time closed-loop control capabilities. This embedded sensing strategy allows accurate manipulation and robust performance in complex operational tasks, such as precise trajectory tracking and dynamic payload handling, without sacrificing system simplicity.

Overall, this combination of an integrated pneumatic-actuated origami design, a scalable and simplified kinematic modeling approach, and embedded proprioceptive feedback positions our system uniquely among soft pneumatic manipulators, offering substantial improvements in control efficiency, payload capacity relative to self-weight, and adaptability to complex manipulation tasks.

Future work could include the exploration of the dynamic characteristics of the system, leveraging the numerically derived models and equations for differentiation. Following the approaches in previous studies [101, 138], incorporating derivatives of pressure-to-configuration relationships could yield a dynamic model that includes the velocity and acceleration terms, offering a comprehensive analysis of the Yoshimura cylinder used as a pneumatic actuator.

Another area of future work will be the incorporation of tactile or force sensors into the actuation modules for detecting contacts from the surroundings. Different sensing mechanisms, such as microfluidic [139, 140], capacitive [141, 142], or fiber-optic sensors [143], can be employed, combined with machine learning algorithms. In this way, the robotic arm will be more interactive with the environment, including humans [8] or robust in rejecting disturbances [144].

3.6. Conclusion and summary

The analytical modeling approach detailed in this chapter successfully enabled real-time control for a specific, geometrically complex, and compliant system. While this is a significant accomplishment, there are considerable lim-

Author (Year)	Control	Model	Volume [mm]	ROM [mm, deg]	Payload [N]	Pressure [kPa]	Weight [g]	Error [%]
Greer (2017)	CL	Sensory	38×38×420	400×400×300	2	10.38	–	5
Hao (2018)	CL	FK	L: 625, R: 75	–	5	250	–	4.09
Zhang (2019)	OL	FK	L: 300, R: 39	90°	24.56	100	70.2	1.21*
Huang (2021)	OL	IK	40×40×200	300×300×400	0.6	200	–	2.89
Toshimitsu (2021)	OL	PCC	–	200×200×100	1.17	–	–	4.23*
Robertson (2021)	OL	Geometric	L: 45	45, 42.5°	2	60	56	–
Shen (2021)	CL	Sensory	L: 114, R: 54	78°	2.3	100	–	3.37*
Zou (2022)	CL	Sensory	L: 800, R: 15	800, 120°	2	103	–	–
Liu (2022)	OL	PCC	L: 2300, R: 100	–	–	100	5000	–
Zaghloul (2023)	CL	Sensory	L: 101.6, R:50.8	76.2	440	185	72	–
Fan (2023)	CL	Sensory	L: 60	40×40×40	2000	80	–	5
Mak (2024)	OL	-	L: 97	94.7°	25.4	140	43	–
Ku (2024)	CL	FK, IK	L: 234.5, R: 80	102, 70.4°	20	80	1100	1.07*
Zhang (2024)	OL	FK, IK	L: 430.7	–	0.302	125	355.2	7.8
This Work (OCM)	CL	FK, IK	L: 155, R: 52.5	160×160×80, 60.4°	10	40	238	2.15*

Table 3.3: Comparative Analysis of Soft Robotic Manipulators. The position errors marked with * are the errors calculated using the methods we defined in Equation 3.41.

itations that prevent this method from being broadened to solve general tasks for deformable robots. The primary limitations of the analytical approach are threefold. First, it lacks generality, as the model is derived specifically for the Yoshimura origami pattern, making it difficult to apply to systems with more complex or irregular geometries. Second, it relies on simplifying assumptions, such as decoupled kinematics, which may not hold for more complex, hyperelastic systems. Finally, the process of deriving these models from first principles is labor-intensive and requires deep domain expertise for each new robot.

These limitations lead to the central motivating question for the next phase of this dissertation: How can we create fast and faithful simulation models for more general and complex compliant systems where a simple analytical model is not feasible? To answer this question, the following chapter introduces a fundamentally different and more generalizable methodology. It presents a data-driven surrogate modeling framework designed to learn the system dynamics directly from high-fidelity simulation data, thereby providing a scalable pathway to creating the simulation environments required for modern robot learning.

Chapter 4.

Data-driven Simulation for Soft Robot Dynamics

4.1. Motivation for a Data-driven Approach

As detailed in the previous chapter, analytical modeling can successfully control a soft, deformable robot for specific, well-defined tasks. However, this method has significant limitations that prevent it from being a general solution for soft robots. The core problems lie in the inherent constraints, simplifying assumptions, and highly specific nature of these models. Even a small change to the robot's hardware or task requires a sensitive and labor-intensive recalibration of its parameters. This lack of generality makes analytical models unsuitable for frameworks like RL, which aim to perform a wide variety of tasks. Since RL is a key method for enabling robots to handle diverse and complex tasks, a new approach is necessary. As discussed in Chapter 2, related work, using an end-to-end RL approach directly on a physical soft robot is not feasible. The physical fragility of soft robots makes them prone to damage during extensive exploration, and the process of gathering sufficient data is too time-consuming. Therefore, the research is naturally driven toward creating a robust simulation environment for deformable bodies. This simulation will provide a safe and efficient platform for developing RL-based control policies, overcoming the limitations of both physical experimentation and traditional analytical models. This work is based on the previous publication, *"Bridging High-Fidelity Simulations and Physics-Based Learning Using A Surrogate Model for Soft Robot Control"* published in the Advanced Intelligent System in 2025.

4.2. Introduction to High-fidelity Soft Robot Simulation

4.2.1 Soft Robotics and Modeling Challenges

The rise of soft robotics has enabled the creation of highly adaptable and safe systems capable of interacting with complex, unstructured environments [76, 145, 146]. These robots, made from flexible, deformable materials, are particularly effective for tasks requiring delicate manipulation and safe human-robot interaction. While the use of external sensors and feedback control has become standard to enhance their performance and safety [16, 17, 147], relying on sensors alone is often not enough. This necessitates the use of robust modeling techniques to predict and control the complex nonlinear behaviors of soft robots [148–150].

Traditional modeling approaches, such as lumped parameter models [13], Cosserat rod theories [15], and voxelization [14], struggle to accurately capture the hyperelastic properties of soft materials. This limitation highlights a critical need for accessible and precise modeling frameworks to fully realize the potential of soft robots in fields like biomedical devices [151], human-robot interaction [152], and precision manipulation [153].

4.2.2 The Role of Simulation and Learning

The FEM models are widely used for analyzing soft materials due to their ability to provide high-fidelity, multiphysics simulations [154, 155]. However, these models are computationally demanding and therefore not practical for real-time or dynamic applications [19, 156]. Their high resource requirements also limit their use for soft robot control [20, 157].

Recently, data-driven methods like deep learning [77, 158, 159] and reinforcement learning (RL) [160–163] have emerged as promising alternatives for learning the dynamics of soft robots. RL, in particular, allows for the learning of optimal control policies through interaction [22, 23] and has been applied to dynamic control tasks in soft robotics [82, 164, 165]. Nonetheless, these

end-to-end approaches face challenges such as damage to the robot during exploration, susceptibility to noise, and difficulty in collecting sufficient data due to the mechanical fragility of soft robots [99, 151]. These issues underscore the necessity for safe, risk-free methods for acquiring data to support RL-based control.

4.2.3 Towards a Unified Framework

In response to these challenges, platforms such as the Simulation Open Framework Architecture (SOFA) have been developed to enable high-fidelity modeling and real-time interaction for soft robots [7, 166, 167]. While FEM-based simulations in SOFA facilitate physical interactions [168, 169], they still face scalability and efficiency issues. Model Order Reduction (MOR) techniques can improve runtime but are often task-specific and computationally expensive to implement [41, 44].

Concurrently, new RL environments for soft robotics, including SOFA-Gym and others [45, 170, 171], have been introduced to allow for the direct learning of control policies. However, achieving high-precision control in both position and force remains challenging due to the complexity of soft body dynamics and limitations in simulation speed and accuracy. These collective efforts highlight a need for a solution that combines efficient simulation fidelity with learning-based control.

This work introduces a new framework that integrates soft robotic models into a physics-based simulation environment compatible with RL. It bridges the gap between high-fidelity FEM models and fast, physics-based simulations by using a computationally efficient alternative model. The framework is designed to achieve three main goals: (i) transferring dynamics from high-fidelity FEM data to a fast physics-based simulation while preserving essential behaviors, (ii) enabling accelerated simulations and data acquisition for RL, and (iii) validating the framework's accuracy across different simulation domains and in sim-to-real transfer tasks. By improving control and simulation efficiency, this framework provides a solid foundation for data-driven learn-

ing in soft robotics, allowing for rapid iteration and safe policy exploration. It paves the way for using high-resolution, physics-informed simulations to speed up reinforcement learning for tasks like manipulation, interaction, and compliant control in real-world settings.

4.3. Framework Overview

A three-domain framework was developed to create a functional bridge between the physical hardware, a high-fidelity FEM simulation, and an accelerated physics-based reinforcement learning environment. The complete simulation-to-learning pipeline is presented in Figure 4.1. A pneumatic soft manipulator was selected as the representative system for this study due to its high degrees of freedom and significant nonlinear dynamic characteristics, which provide a structured benchmark for evaluating control-oriented simulation strategies [172–174].

The first domain is the high-fidelity simulation, which was constructed in the SOFA framework to serve as a physically grounded digital twin of the hardware. This model incorporated the robot's nonlinear material behavior and pressure-driven actuation, enabling an accurate prediction of the manipulator's response under varying pressure inputs while avoiding the risk of hardware degradation. To manage the high computational cost of this model, MOR techniques were applied. This step enabled the efficient generation of time-series data for subsequent learning of the system's forward and inverse kinematics [41, 44, 175].

Despite its precision, the SOFA-based simulation remains computationally intensive and is therefore unsuitable for reinforcement learning, which requires a massive volume of agent-environment interactions to train a policy [45, 170]. To address this limitation, the second domain of the framework was introduced, which is a computationally efficient surrogate model implemented in physics-based simulation such as PyBullet [30]. This simulator was selected for its real-time performance, robust collision handling, and seamless integration with established machine learning libraries. The surrogate model

itself comprises a combination of revolute and prismatic joints arranged to approximate the primary deformation patterns observed in the FEM simulation, providing an environment fast enough for efficient policy training and data generation.

The primary contribution of this work lies in the integration of these three distinct domains, the real hardware, the FEM-based simulation, and the physics-driven surrogate environment, through a set of data-driven mapping functions. These mappings are the critical connections that ensure dynamic consistency across the pipeline. A calibration process aligns the real robot with the FEM model, a learned dynamics and actuation mapping translates the complex physics from the FEM model to the fast surrogate, and a final sim-to-real protocol enables the transfer of learned policies back to the physical hardware. These connections create a bidirectional flow of information that is essential for the validation of the framework's success.

4.4. Model Calibration

4.4.1 FEM Model Construction

The calibration process is initiated by replicating the configuration of the real robot within the SOFA FEM environment. As depicted in Figure 4.2-(A), the physical system consists of a bellow-type parallel manipulator that is actuated by three independent pressure regulators and mounted on a fixed frame with a movable top plate. This physical setup was reconstructed in SOFA with a volumetric tetrahedron mesh that includes three internal cavity meshes for simulating the actuation, as shown in Figure 4.2-(B).

In order to align the simulation with the physical hardware, the state variables for the model were defined as illustrated in Figure 4.3. These state variables are composed of the position of the tool center point (TCP) ($p \in \mathbb{R}^3$), the length of the manipulator's backbone ($l \in \mathbb{R}^1$), its curvature ($\kappa \in \mathbb{R}^1$), and the relative orientation of the top plate ($q \in \mathbb{R}^3$). Collectively, these variables form a configuration state vector ($\mathbf{X} \in \mathbb{R}^{N_s}$) and its corresponding velocity

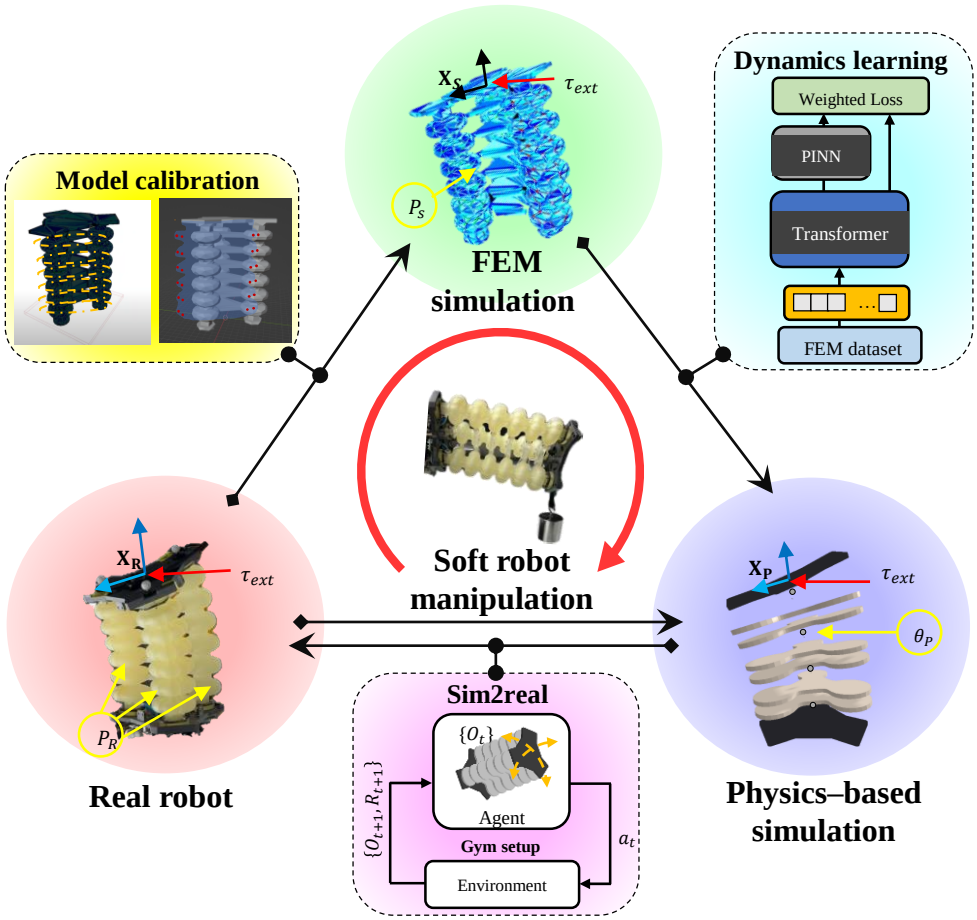


Figure 4.1: Overview of the proposed framework connecting three different domains ("Real robot", "FEM simulation", and "Physics-based simulation") representing a common soft manipulator. The soft manipulator is actuated by the pressure input (P) and the output includes the state positions ($\mathbf{X}$) 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.

vector ($\dot{\mathbf{X}} \in \mathbb{R}^{N_s}$), where N_s denotes the total number of dimensions for the state variables.

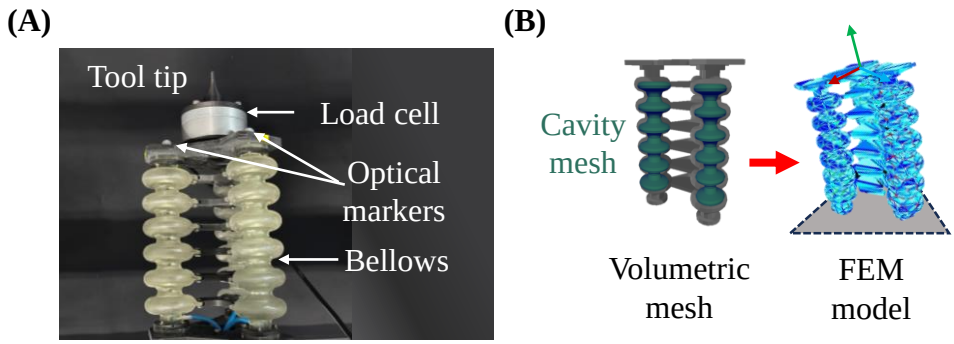


Figure 4.2: (A) Components of the soft manipulator in the real-world setup, including the bellows-integrated manipulator. (B) FEM model of the soft manipulator in the SOFA simulation scene, utilizing a mesh model that replicates the physical structure.

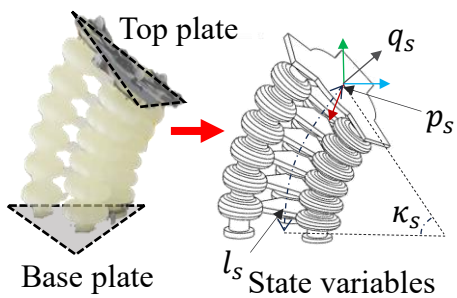


Figure 4.3: Configuration state information (**X**) of the soft manipulator, which are necessary to define the behavior of the manipulator.

4.4.2 SOFA Scene Setup

The interpretation of the elastic body in the SOFA libraries was used with appropriate constraints to define the rigid parts of the robot, such as the bellows connecting parts and the upper frames [44]. The setups are implemented in Python scripts using the Python3 libraries. These code files define the simulation GUI, referred to as the “SOFA scene", which includes the boundary conditions, the definitions of pneumatic actuation, the solver configuration and the properties of the material. As shown in Figure 4.4, the material properties were initially assumed and later refined to achieve a configuration similar to

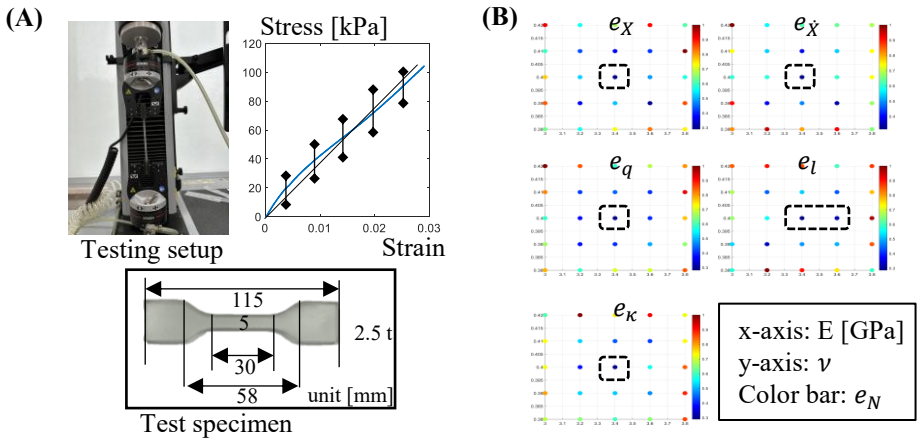


Figure 4.4: (A) Tensile test setup for the elastic resin specimen used in the soft manipulator. (B) Optimization of Young's modulus (E) and Poisson's ratio (ν) to minimize the configuration error in the SOFA simulation. The color bar represents the error range, with the dashed box highlighting the property combination that results in the lowest error.

that of the real robot. For the solvers, the types, thresholds, and maximum iteration numbers were empirically determined, balancing computation time and speed based on our computational setup.

The mesh files required for the SOFA simulation setup were initially exported using the exact design file used to fabricate the soft manipulator [172]. Therefore, the dimensions and specifications of the model in the SOFA simulation were identical to those of the real robot. Initially, Standard Tessellation Language (STL) files were exported and converted to volumetric meshes for the elastic body. The 3D Visualization Toolkit (VTK) file format was used to reconstruct the bellow model. As shown in Figure 4.2-(B), the cavity mesh (green mesh) only required surface information since it was used to define pneumatic actuation. Therefore, the STL mesh files were directly utilized for this purpose.

4.4.3 Material Characterization and Property Estimation

As shown in Figure 4.4-(A), dogbone specimens were printed using Elastic 50A Resin V2 (Formlabs) following ASTM D638 Type IV standards. Tensile tests were conducted at a crosshead speed of 0.01 mm/s under room temperature conditions ($20°C$) using a tensile tester model, with 20 repetitions per sample group. Young's modulus (E) was extracted from the initial linear region of the stress-strain curve (up to 3% strain), while Poisson's ratio (ν) was initially set based on the material properties.

To calibrate the FEM model, both the material parameters and the solver configurations were refined to match the physical behavior of the soft manipulator. The SOFA simulation scene incorporated appropriate boundary conditions and custom plugins, with Young's modulus (E) initially derived from tensile tests and Poisson's ratio (ν) from the manufacturer's specifications. However, discrepancies between the simulated and experimental behaviors necessitated further tuning. A grid search around the nominal values was performed to minimize configuration error with respect to the real robot.

Although initial estimates of E and ν were obtained from material testing and datasheets, sim2real mismatches motivated a refined search. A two-dimensional sweep was performed around the nominal values. For each pair (E, ν), SOFA simulations were executed, and the configuration error was calculated as the RMSE between the simulated and experimental TCP positions and the orientations of the end effectors. The optimal parameter pair was selected to minimize this error, as visualized in Figure 4.4-(B).

4.4.4 Domain Mapping Functions

In the SOFA environment, the pressurized cavities were simulated using defined boundary conditions and were solved with quadratic program (QP) solvers [169]. A multilayer perceptron (MLP) was employed to correct for discrepancies between the real-world pressures (P_R) and the simulated pressures

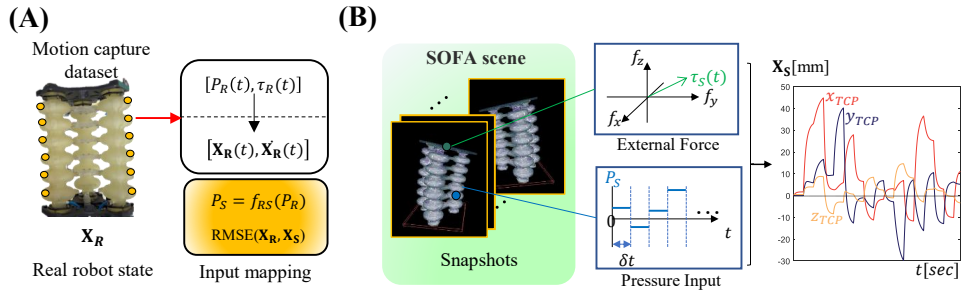


Figure 4.5: (A) Input pressure mapping and calibration process between the real robot and the FEM model, achieved using motion capture data ($\mathbf{X}_R$) to align the dynamics between the two domains. (B) Snapshot collection from the SOFA simulation, where the three bellows were actuated with varied pressures (P_S) and external forces (τ_S). The right plot presents the position data ($\mathbf{X}_S$) collected over the simulations.

(P_S).

$$P_S = f_{RS}(P_R), \tag{4.1}$$

This function, f_{RS}, calibrates the input pressures that are required to achieve consistent TCP positioning between the two domains. As illustrated in Figure 4.5-(A), the network maps the measured inputs, which include pressures ($P_S \in \mathbb{R}^3$) and external forces at the TCP ($\tau \in \mathbb{R}^3$), to the estimated dynamic states ($\mathbf{X}, \dot{\mathbf{X}}$). The detailed information regarding this mapping process, loss function and the hyperparameters used for this MLP are provided in Appendix B.

4.4.5 Model Order Reduction by Proper Orthogonal Decomposition

To enable an efficient simulation while preserving the essential deformation characteristics of the soft manipulator, Model Order Reduction (MOR) was applied using the Proper Orthogonal Decomposition (POD) method [41, 44]. The full-order dynamics of the deformable manipulator are governed by the following equation.

$$\mathbb{M}(\mathbf{p})\,\ddot{\mathbf{p}} = \mathbb{P}(t) - \mathbb{F}(\mathbf{p}, \dot{\mathbf{p}}) - \mathbb{H}^T \boldsymbol{\lambda}(t). \tag{4.2}$$

Here, the high-dimensional nodal state of the FEM model at each time step is denoted as $\mathbf{p}(t) \in \mathbb{R}^{3N}$, where N is the number of mesh nodes, which was initially 4327. The terms $\mathbb{M}$, $\mathbb{P}$, $\mathbb{F}$, $\mathbb{H}$, and $\boldsymbol{\lambda}$ represent the mass matrix, external pressure-induced forces, internal elastic and damping forces, constraint Jacobian, and Lagrange multipliers, respectively.

The MOR process began by constructing a snapshot matrix $Sn \in \mathbb{R}^{3N \times M}$ through the collection of M time-series nodal states from the SOFA simulation.

$$Sn = [\mathbf{p}(t_1),\ \mathbf{p}(t_2),\ \ldots,\ \mathbf{p}(t_M)]. \tag{4.3}$$

These snapshots were obtained by applying randomized pressure inputs and external forces to the FEM model to sufficiently explore the configuration space. A total of $M = 20,000$ samples were collected. Singular Value Decomposition (SVD) was then applied to the snapshot matrix Sn to extract the dominant spatial deformation modes.

$$Sn = U\Sigma V^T. \tag{4.4}$$

In this decomposition, the matrix U contains the orthonormal spatial POD modes, Σ holds the singular values, and V^T represents the temporal coefficients. On a workstation equipped with an AMD Ryzen 9 processor and 128 GB of RAM, the full SVD of the matrix $Sn \in \mathbb{R}^{12981 \times 20000}$ was completed in approximately 20 minutes using multi-threaded CPU execution. From this, the first $r = 1223$ dominant modes were retained to construct the reduced basis.

$$\Phi = U_r \in \mathbb{R}^{3N \times 1223}. \tag{4.5}$$

The high-dimensional nodal state could then be approximated by projecting it onto this lower-dimensional subspace.

$$\mathbf{p}(t) \approx \Phi\, \mathbf{a}(t). \tag{4.6}$$

Here, $\mathbf{a}(t) \in \mathbb{R}^{1223}$ denotes the generalized coordinates of the system within the

reduced space. By substituting this approximation into the full-order dynamics and applying a Galerkin projection, the reduced-order system is obtained.

$$\mathbb{M}_r \ddot{\mathbf{a}} = \mathbb{P}_r(t) - \mathbb{F}_r(\mathbf{a}, \dot{\mathbf{a}}) - \mathbb{H}_r^T \boldsymbol{\lambda}_r(t). \tag{4.7}$$

The reduced matrices expressed with lower “r" are defined as follows.

$$\begin{aligned} \mathbb{M}_r &= \Phi^T \mathbb{M}(\mathbf{p}) \Phi, \\ \mathbb{P}_r(t) &= \Phi^T \mathbb{P}(t), \\ \mathbb{F}_r(\mathbf{a}, \dot{\mathbf{a}}) &= \Phi^T \mathbb{F}(\Phi \mathbf{a}, \Phi \dot{\mathbf{a}}), \\ \mathbb{H}_r &= \Phi^T \mathbb{H}, \quad \boldsymbol{\lambda}_r = \boldsymbol{\lambda}. \end{aligned} \tag{4.8}$$

The mass matrix $\mathbb{M}(\mathbf{p})$ was precomputed using the SOFA FEM engine with a constant material density of 1040 $\mathrm{kg/m^3}$. The external force vector $\mathbb{P}(t)$ was generated using pressure constraint conditions in SOFA library, while the internal force term $\mathbb{F}(\mathbf{p}, \dot{\mathbf{p}})$ included a corotational FEM model with Rayleigh damping. The constraint Jacobian $\mathbb{H}$ captured the influence of fixed supports and contact constraints.

This reduced-order formulation significantly accelerated the simulation while maintaining sufficient accuracy for control and learning. The reduced coordinates $\mathbf{a}(t)$ were subsequently used to construct a compact state vector $\mathbf{X}_S(t) \in \mathbb{R}^{N_s}$, which contains physically meaningful quantities required for learning, such as the TCP position, curvature, arc length, and orientation. These reduced-order representations served as the input features for the dynamics models and learning policies developed in this work.

4.5. Dynamics modeling using surrogate model

4.5.1 Transformer-based Physics-Informed Dynamics Modeling

The dynamics model was designed to predict the Cartesian velocity of the manipulator based on a time-series of its configurations and the applied forces based on the calibrated FEM model. As shown in Figure 4.6, the entire structure

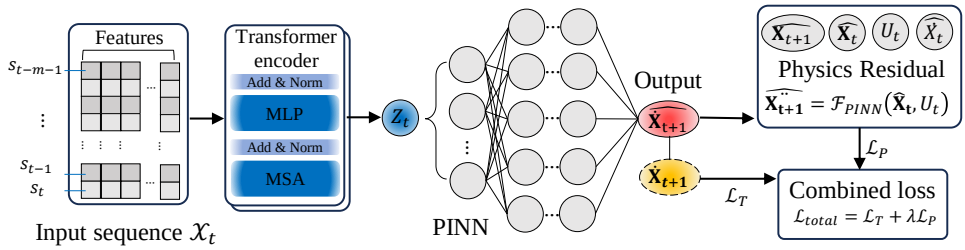


Figure 4.6: Architecture of the data-driven dynamics model combining a Transformer encoder with physics-informed residuals. The training objective integrates data loss and physical constraints into a unified loss function.

of this dynamics model was trained using the dataset collected from the SOFA simulation. The input ($\mathbf{s}_t$) at each time step was a vector defined as

$$\mathbf{s}_t = [\mathbf{X_S}(t), P_S(t), \tau_P(t)] \in \mathbb{R}^{N_s+6}. \tag{4.9}$$

A sequence of $m = 10$ past steps was collected to form the input sequence for the model.

$$\mathcal{X}_t = [\mathbf{s}_{t-m-1}, \ \ldots, \ \mathbf{s}_t] \in \mathbb{R}^{10\times(N_s+6)}. \tag{4.10}$$

This input tensor was first projected into an embedding space of dimension 128 via a linear layer, followed by the addition of positional encoding.

The sequence was then processed by a Transformer encoder, $\mathcal{T}_{enc}$ [176]. This encoder consisted of two attention blocks, each containing a multi-head self-attention mechanism with four heads and a feedforward layer with a hidden size of 256. Layer normalization and residual connections were applied using a pre-norm structure, and no dropout was used during training. The output of the encoder corresponding to the final time step was extracted as a latent vector $\mathbf{Z}_t \in \mathbb{R}^{128}$, which serves as a context-aware summary of the system's history for downstream dynamics modeling.

$$\mathbf{Z}_t = \mathcal{T}_{enc}(\mathcal{X}_t). \tag{4.11}$$

This latent vector was subsequently passed to a Physics-Informed Neural Network (PINN) [177]. The PINN was implemented as a multilayer perceptron with three hidden layers of size 128, each followed by a ReLU activation function. The output of the PINN was the predicted Cartesian velocity of the manipulator.

$$\hat{\dot{\mathbf{X}}}_S(t+1) = f_{\text{PINN}}(\mathbf{Z}_t). \tag{4.12}$$

The training of this hybrid architecture was guided by a combined loss function that integrated both a data-driven term and a physics residual term.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_T + \lambda \mathcal{L}_P. \tag{4.13}$$

The data-driven loss, $\mathcal{L}_T$, measures the direct error between the predicted and ground-truth velocities. The physics residual, $\mathcal{L}_P$, measures how well the network's prediction conforms to a simplified multibody dynamics model [41, 44].

$$\mathcal{R}_t = \mathbb{M}(\hat{\mathbf{X}}_S(t))\hat{\ddot{\mathbf{X}}}_S(t) - \mathbb{P}_S(t) + \mathbb{F}(\hat{\mathbf{X}}_S(t), \hat{\dot{\mathbf{X}}}_S(t)) + \mathbb{H}^\top \lambda(t). \tag{4.14}$$

Here, $\mathbb{M}$ denotes the mass matrix, which was precomputed from the POD-reduced FEM model assuming a material density of 1040 kg/m^3. The term $\mathbb{P}_S(t)$ is the pressure-induced force vector, $\mathbb{F}$ represents the internal elastic and damping forces computed with Rayleigh coefficients $\alpha = 0.05$ and $\beta = 0.005$, and $\mathbb{H}^\top \lambda(t)$ captures the constraint forces. The total loss was thus defined as

$$\mathcal{L}_{total} = \underbrace{\|\hat{\dot{\mathbf{X}}}_S(t+1) - \dot{\mathbf{X}}_S(t+1)\|^2}_{\mathcal{L}_T} + \lambda \cdot \underbrace{\|\mathcal{R}_t\|^2}_{\mathcal{L}_P}, \tag{4.15}$$

where the loss weight λ was manually selected as 0.15 to balance data fidelity with physical consistency [178]. The model was trained for 300 epochs using the Adam optimizer with an initial learning rate of 0.0005 and a batch size of 64 on a dataset of 20,000 samples. All ground truth velocities $\dot{\mathbf{X}}_S(t+1)$ were obtained from the SOFA simulation dataset. This process results in a learned

forward dynamics function of the form

$$\hat{\dot{\mathbf{X}}}_S(t+1) = f_S(\mathcal{X}_t), \tag{4.16}$$

where f_S denotes the combined Transformer and PINN model that maps an input sequence $\mathcal{X}_t$ to the predicted Cartesian velocity while preserving physical consistency.

4.5.2 Inverse Dynamics Modeling

The inverse dynamics model was trained to infer the required pneumatic pressure inputs (P) that produce a desired Cartesian velocity, given the current system state and external contact forces. This process is undetermined and cannot be solved by a conventional equation solver due to the redundancy and system compliance. Therefore, the approach using a supervised neural network with an auxiliary cycle-consistency loss was used to enforce physical plausibility as well as to determine the unique solution of the pressure input when a high-dimensional desired state ($\mathbf{X}(t)$) is given.

The input to the model was a concatenated vector comprising the reduced-order state vector $\mathbf{X}_S(t)$, the external contact wrench $\boldsymbol{\tau}_{\text{ext}}(t)$, and the desired velocity $\dot{\mathbf{X}}_S(t+1)$. This input was fed into a multilayer perceptron, $f_{\text{S}}^{\dagger}$, which was implemented with three hidden layers of size 128 using ReLU activation. The network outputted a 3-dimensional vector $\hat{P}_S(t) \in \mathbb{R}^3$, representing the estimated pressure inputs to the three cavities.

The primary training loss was the supervised L2 loss between the predicted and true pressure values.

$$\mathcal{L}_{\text{S}}^{\dagger} = \|\hat{P}_S(t) - P_S(t)\|^2. \tag{4.17}$$

To improve consistency with the physical dynamics, a secondary cycle-consistency loss was added. This was implemented by passing the predicted pressure through the pretrained forward model f_S to obtain a synthetic prediction of velocity. The new input sequence $\mathcal{X}'_t$ was constructed by replacing the true

pressure in the original sequence with the predicted one. The resulting velocity was compared against the ground truth.

$$\mathcal{L}_{\text{cyc}} = \| f_S(\mathcal{X}'_t) - \dot{\mathbf{X}}_S(t+1) \|^2. \tag{4.18}$$

The total training loss for the inverse model was defined as

$$\mathcal{L}^{\dagger}_{\text{total}} = \mathcal{L}^{\dagger}_{\text{S}} + \beta \cdot \mathcal{L}_{\text{cyc}}, \tag{4.19}$$

where $\beta = 0.1$ was empirically selected to balance direct regression and cycle consistency. Training was performed for 200 epochs using the Adam optimizer with a learning rate of 0.001 and a batch size of 64 on the same dataset used for forward modeling.

4.6. 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 $\mathbf{X}_{\text{S}}$, including the motion of the TCP, and was calibrated to match the ROM observed in the high-fidelity SOFA simulation.

This approach preserves essential deformation and contact characteristics while offering a reduced computational cost, making it well-suited for RL and online control tasks. As shown in Figure 4.7-(A). The model was ultimately implemented in a URDF (Unified Robot Description Format) file with explicit definitions of mass distribution, joint limits, and link dimensions, demonstrating dynamic consistency with the reference configuration space of the real robot.

4.6.1 Geometric Design: Based on FEM Node Analysis

The geometric structure of the surrogate model was determined through a node-level analysis of the high-fidelity FEM model.

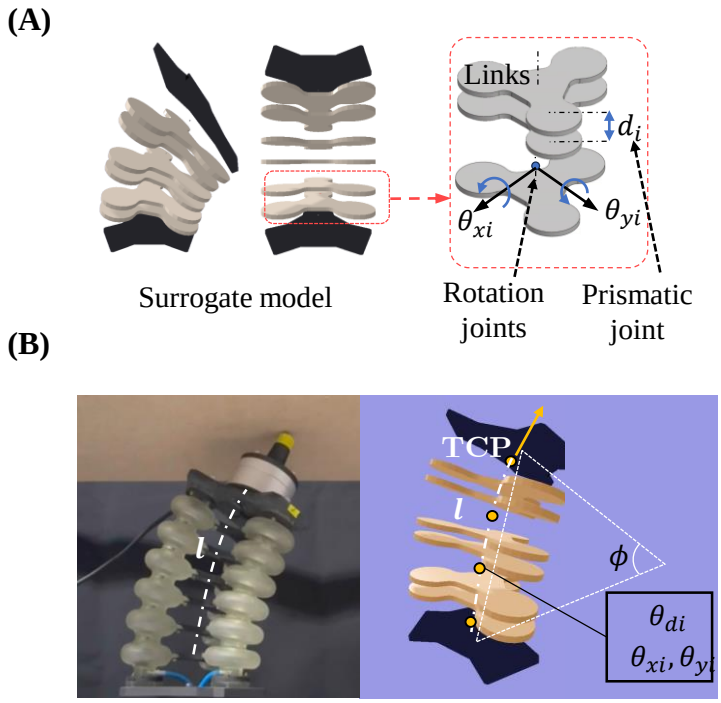


Figure 4.7: (A) Structure of the surrogate model composed of planar links connected by rotational joints ($\theta_{x_i}, \theta_{y_i}$) and prismatic joints ($\theta_{d_i}$). (B) Configuration matching between the surrogate model and the real robot, illustrating consistent deformation behavior and parameter correspondence between the two domains.

Contact Boundary Node Analysis One of the crucial pieces of information about the deformable manipulator was the contact boundary, and this was estimated from the FEM nodal data. First, six boundary nodes ($\mathbf{X}_\mathrm{b}$) were extracted from each of the six bellows segments to define the collision boundary (B_c) of the deformable body. To analyze the hexagon formed by these nodes across a large number of random actuation samples, a regression plane was fitted for each layer to provide a consistent planar representation of the nodal distribution.

The estimation of a regression plane from a set of 3D input points $v \in \mathbb{R}^{n\times 3}$ is based on Singular Value Decomposition (SVD).

- First, the average center of the input points v is calculated as:

$$\overline{v} = \frac{1}{n} \sum_{i=1}^{n} v_i. \tag{4.20}$$

- The points are then centered by subtracting the mean, $v_{cp} = v - \overline{v}$, and SVD is performed:

$$[U, S, V] = \text{svd}(v_{cp}). \tag{4.21}$$

- The normal vector to the regression plane, $\vec{n}$, is obtained as the last column of V. The plane equation is defined as $n_x x + n_y y + n_z z + d = 0$, where $d = -\vec{n} \cdot \overline{v}$.

- Each input point v_i is orthogonally projected onto the estimated plane using:

$$\delta_p = \vec{n} \cdot v_i + d, \quad v_{pi} = v_i - \delta_p \cdot \vec{n}. \tag{4.22}$$

Through this process, the average contact radii by layer (R_i) were measured, and the smallest of these values, R_{min}, was used as a conservative constraint in the surrogate geometry, as illustrated by the red dashed circles in Figure 4.8-(B).

Structure Design Based on this geometric characterization, the surrogate model comprises six identical planar links stacked vertically. The overall height, diameter, and thickness of the links were derived from the length of the arc (l) and the contact boundaries as shown in Figure 4.9.

4.6.2 URDF Implementation and Design Specifications

The surrogate model was designed in an URDF file using XML format.

- **Link Composition**: Two types of surface links were used: the links that compose the center body and the plane bodies placed at the top and bottom of the model.

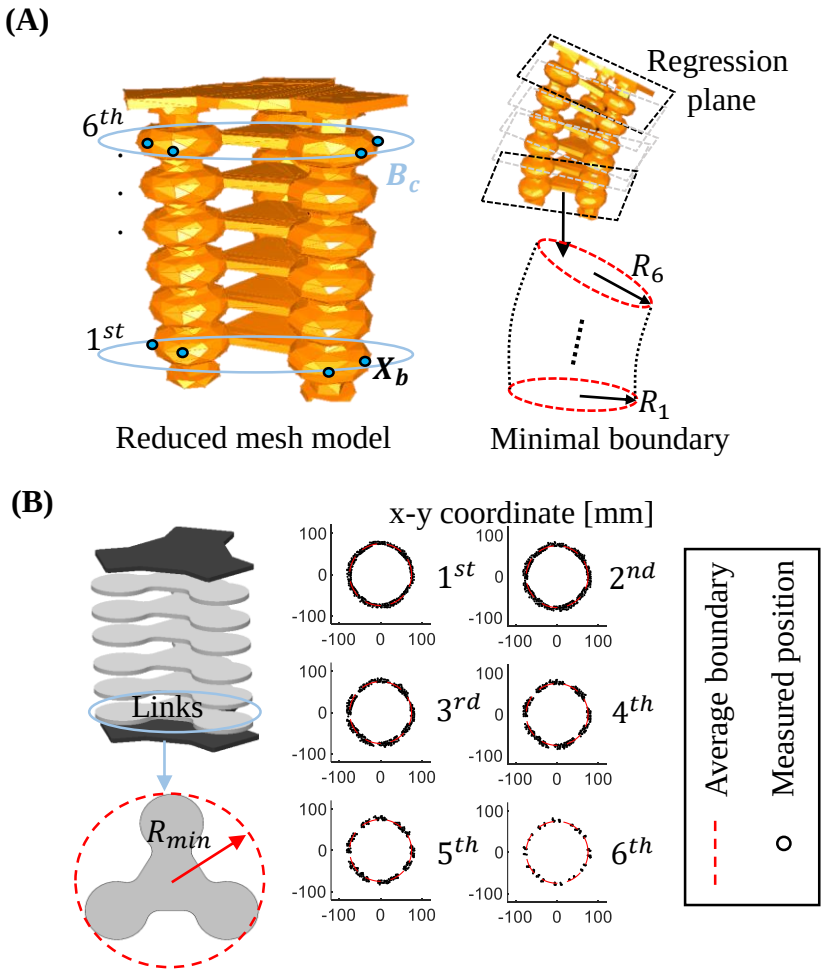


Figure 4.8: (A) Reduced FEM mesh with layer-wise boundary nodes ($\mathbf{X}_\mathrm{b}$) marked in blue. These nodes define the potential collision boundary (B_c) of each segment and are preserved during model order reduction to ensure contact fidelity. (B) Contact boundaries computed by projecting $\mathbf{X}_\mathrm{b}$ 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.

- **Mass**: The actual soft manipulator's mass of approximately 820 g was equally distributed across all layers to set the virtual weight.
- **Joint Configuration**: Each pair of adjacent links is connected via one

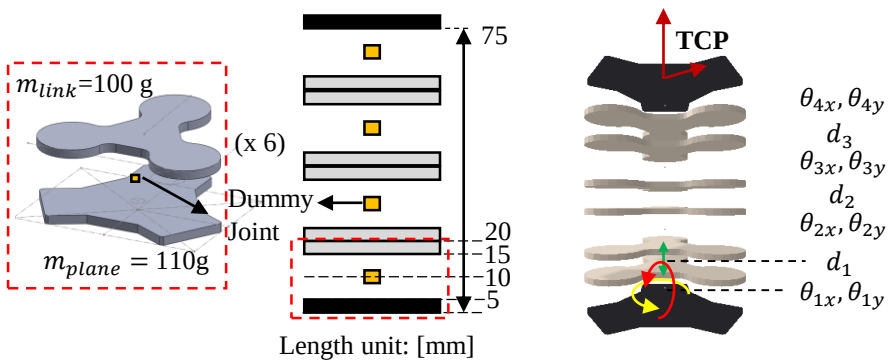


Figure 4.9: Design parameters of the surrogate model

prismatic joint (θ_{d_i}) to mimic longitudinal extension and two rotational joints ($\theta_{x_i}, \theta_{y_i}$) to enable omnidirectional bending. The rotational joints were defined as dummy joints with no mass components. Therefore, a total of eight rotation joints and three prismatic joints were defined in the surrogate model.

Ensuring Physical Realism via Joint Constraints To ensure that the surrogate model exhibits physically realistic deformation behavior, joint constraints were applied to adjacent joint groups. Without such constraints, the model can exhibit unrealistic behavior, such as large curvature changes between links, as shown in Figure 4.10-(A), which are not observed in the real system. To prevent abrupt bending or unrealistic shape transitions, the following inequality constraints were enforced:

$$\begin{aligned} |\theta_{x_i} - \theta_{x_{i+1}}| &< \delta_x, \\ |\theta_{y_i} - \theta_{y_{i+1}}| &< \delta_y, \text{ and} \\ |\theta_{d_i} - \theta_{d_{i+1}}| &< \delta_d, \end{aligned} \tag{4.23}$$

where $\delta_x = 0.01$, $\delta_y = 0.01$, and $\delta_d = 0.005$ are constants that limit inter-joint variations. These bounds were chosen to reflect the smooth, continuous curvature observed in the real robot. The absolute joint limits were also constrained

to reflect the physical range of motion:

$$\theta_x, \theta_y \in [-0.2, 0.2] \quad \text{and} \quad \theta_d \in [0, 0.02]. \tag{4.24}$$

These constraints were applied within the control environment during optimization and policy rollout to restrict the exploration space to feasible regions. They enforce spatial smoothness across joint segments, consistent with the continuous curvature observed in physical continuum robots. As a result, applying the grouped constraints results in smooth and physically plausible postures, as shown in Figure 4.10-(B).

4.7. Aligning Between Two Actuation Spaces

To bridge the high-fidelity FEM simulation and the computationally efficient surrogate model, a data-driven mapping to align their actuation spaces was established. The FEM model is actuated by pressures ($P_S \in \mathbb{R}^3$), whereas the surrogate model is controlled by a vector of joint angles ($\boldsymbol{\theta}_P \in \mathbb{R}^{N_a}$), where N_a is the DOF of the surrogate model. The forward mapping $f : P_S \rightarrow \boldsymbol{\theta}_P$ was trained to ensure that a given actuation command produces consistent motions across both simulations ($\mathbf{X}_\mathrm{S} \approx \mathbf{X}_\mathrm{P}$), as illustrated in Figure 4.11.

This mapping was constructed by first generating a dataset of corresponding actuation pairs. A constrained optimization problem was solved that iteratively adjusted the joint angles $\boldsymbol{\theta}_P$ to minimize the Euclidean distance between the surrogate model's TCP position, $\mathbf{X}_\mathrm{P}(\boldsymbol{\theta}_P)$, and a target TCP position, $\mathbf{X}_\mathrm{S}(P_S)$, obtained from the FEM simulation. The optimization problem is formulated as:

$$\boldsymbol{\theta}_P^* = \arg\min_{\boldsymbol{\theta}_P} \|\mathbf{X}_\mathrm{P}(\boldsymbol{\theta}_P) - \mathbf{X}_\mathrm{S}(P_S)\|_2, \tag{4.25}$$

subject to the joint limits:

$$L_\theta \leq \boldsymbol{\theta}_P \leq U_\theta. \tag{4.26}$$

The optimization problem was solved using the limited-memory Broy-

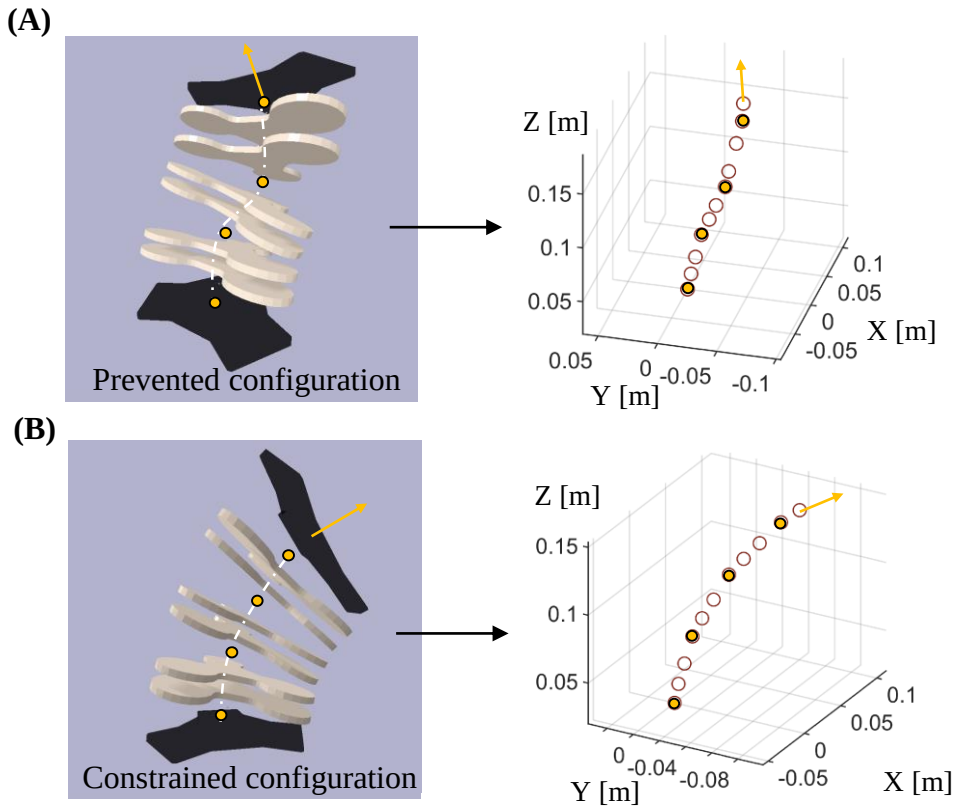


Figure 4.10: (A) Example of an unrealistic configuration in the surrogate model, where multiple curvatures appear across joint segments, deviating from the physical behavior of the real robot. (B) Example of a feasible configuration generated by applying joint constraints, ensuring smooth deformation.

den–Fletcher–Goldfarb–Shanno algorithm with box constraints (L-BFGS-B) [179]. This method is well-suited for handling the redundant DOFs of the surrogate model while ensuring the resulting configurations remain physically feasible. The procedure is detailed in Algorithm 1.

After collecting a dataset of optimized joint configurations $\boldsymbol{\theta}_P^*$ across a wide range of pressure inputs P_S, a fully connected MLP was trained to approximate this mapping directly:

$$\boldsymbol{\theta}_P(t) = f_{\mathrm{SP}}(P_S(t)), \tag{4.27}$$

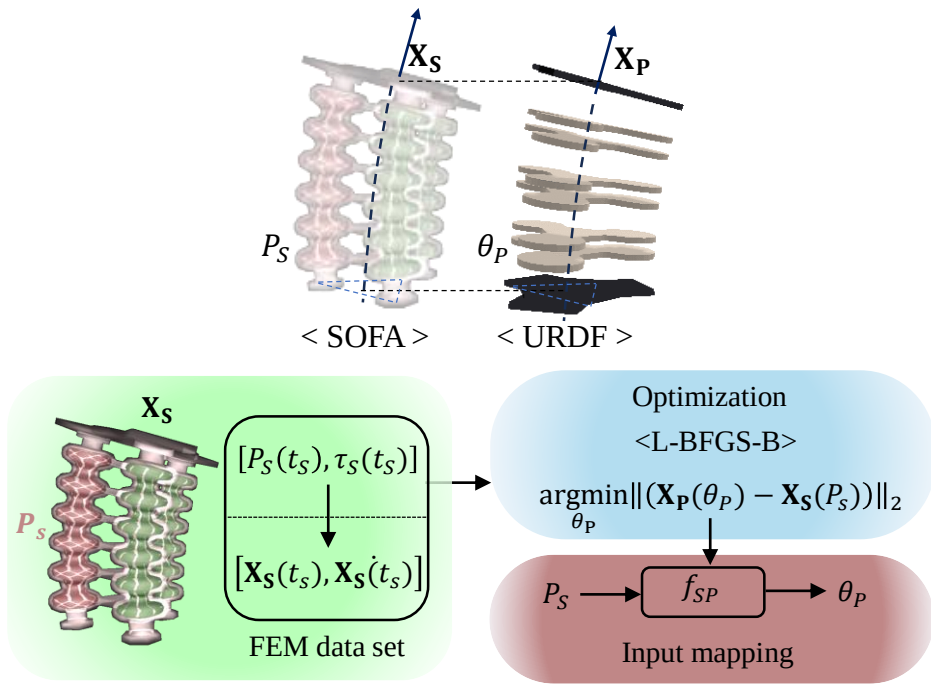


Figure 4.11: Mapping between pressure inputs (P_S) and joint angles (θ_P) across two simulation environments. The dataset, derived from SOFA simulation snapshots, was used to optimize state alignment between the SOFA and URDF models. The trained forward model predicts joint angles (θ_P) from pressure 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.

where f_{map} denotes the learned function approximator. This network provides a continuous and computationally efficient translation from pressure commands to joint space signals, allowing the surrogate model to replicate the behavior specified by the FEM simulation's actuation space.

The complete mapping pipeline, from optimization-based correspondence to the learned regression function, is depicted in Figure 4.11. This cross-domain alignment serves as a critical bridge for sim-to-sim transfer, enabling efficient reinforcement learning in the surrogate domain while pre-

Algorithm 1 L-BFGS-B Optimization for Actuation Space Mapping

Input:

- Target TCP position $\mathbf{X}_{\text{target}} \in \mathbb{R}^3$ (from FEM)
- Initial guess for joint angles $\boldsymbol{\theta}_0 \in \mathbb{R}^{N_a}$
- Joint limits $L_\theta, U_\theta \in \mathbb{R}^{N_a}$
- Maximum iterations $N_{\max}$
- Convergence tolerance $\epsilon > 0$

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

Define objective function:

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

Solve optimization problem:

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

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:

$$\mathbf{X}_{\text{actual}} = \mathbf{X}_{\text{P}}(\boldsymbol{\theta}_P^k)$$

    Compute gradient of the objective function: $\nabla f(\boldsymbol{\theta}_P^k)$

    Update joint angles using an L-BFGS-B step:

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

    (where α^k is step size and $\mathbf{H}_k$ is the inverse Hessian approximation)

    Enforce joint limits (projection):

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

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

        break

    end if

    $k \leftarrow k + 1$

end while

return $\boldsymbol{\theta}_P^*$

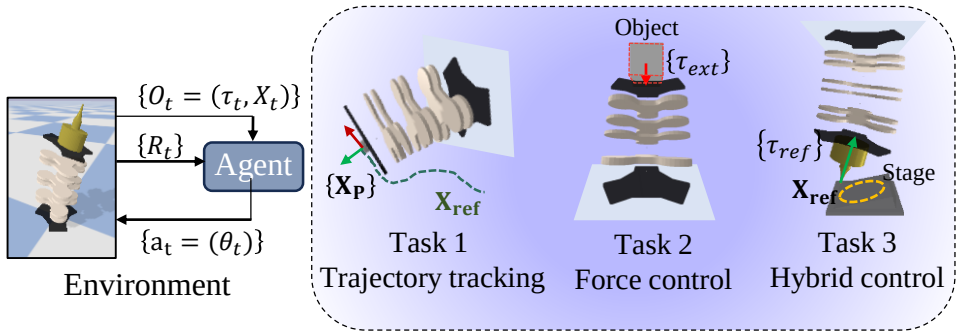


Figure 4.12: Customized RL environment setup illustrating the interaction between the agent and the environment across three tasks involving position and force controls.

serving the physical fidelity of the FEM reference.

4.8. Customized Gymnasium Environment

4.8.1 Environment Setup

A customized RL environment was developed for evaluating the effectiveness of the surrogate model and the trained dynamics. The environment was built for full compatibility with Gymnasium interfaces [180], exposing a conventional agent structure that includes an observation (O_t), action (a_t), and reward (R_t). A modular class structure was implemented to manage the core interaction logic, including actuation, contact detection, and state updates. The overall structure of the environment and the agent's interaction across the defined tasks are illustrated in Figure 4.12.

4.8.2 State and Action Spaces

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

$$O_t = \left[\boldsymbol{\theta}_P,\ \dot{\boldsymbol{\theta}}_P,\ \mathbf{X}_P,\ \dot{\mathbf{X}}_P,\ \boldsymbol{\tau}_{\text{ext}}\right],$$

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

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

$$a_t = \boldsymbol{\theta}_P^{\text{cmd}} \in \mathbb{R}^{N_a},$$

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

4.8.3 Force Calibration of the Surrogate Model

Force calibration was performed to match the realistic values measured using the PyBullet simulation environment. Torque and force were measurable for all joints using the provided functions, and a TCP node was defined to measure contact forces that were included in the state information. Calibration between the real measured force (F_R) and the simulated force (F_P) was conducted. As shown in Figure 4.13-(A), the six axis load cell was attached to the end-effector of the robot and then fixed to the jig. The movement of the robot was constrained and the load cell measured the resultant force generated by the three pressure levels applied. Similar setup was realized in the Pybullet environment, in this case the functions were simply used to measure the force measured at the TCP node. Various combinations of actuation inputs were tested in both domains. Figure 4.13-(B) illustrates the normalized force vector (n_F) in the x-y-z coordinates.

Since the mapping between the two actuation spaces was derived as f_{SP}, the magnitude of the resulting force at specific configurations could be compared. The magnitude of the measured force at the TCP in the simulation is shown in Figure 4.13-(C). Each vector component matched the real data from the load cell, as shown in Figure 4.13-(D). A linear fit of the force components (F_{fit}) was obtained component-wise (F_x, F_y, and F_z). Linear fitting parameters were calculated based on the experimental data and remained unchanged across all task implementations.

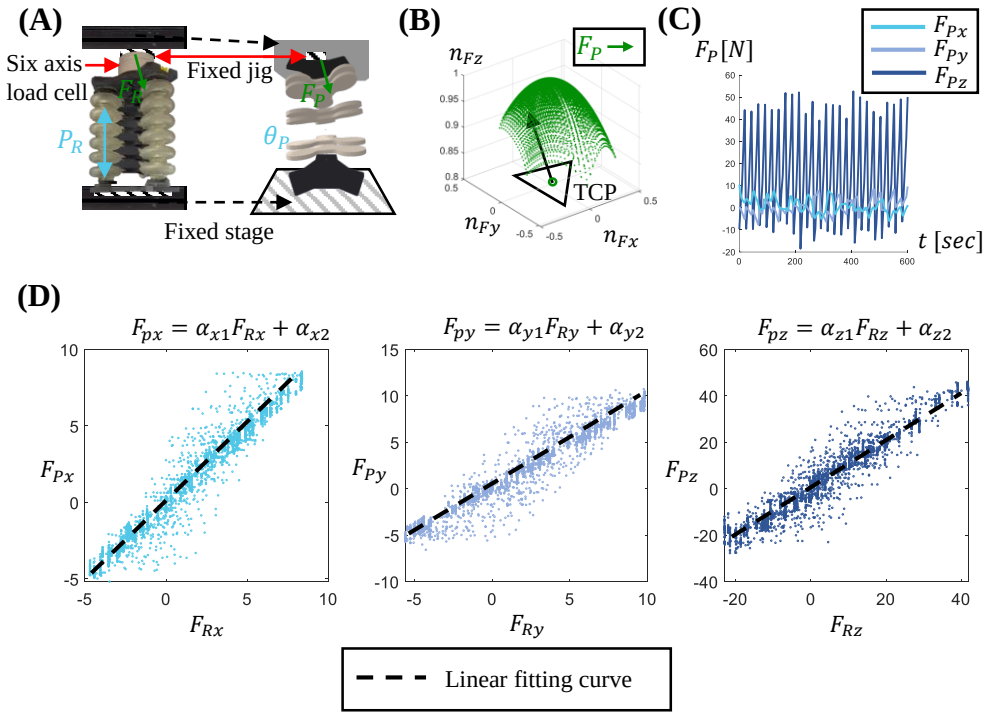


Figure 4.13: (A) Experimental setup for force calibration with the real robot, showing the load cell fixed to the frame and the jig. The robot posture was fixed and the resultant force at the TCP (F_R) was measured as a function of the pressure input (P_R). In the PyBullet simulation, a similar fixed stage and jig were built, where the resulting force (F_P) was measured as the joint angle (θ_P) varied. (B) Normalized force vector (n_F) in x-y-z coordinates for various actuation in both simulations. (C) Measured force magnitude at the TCP in the PyBullet simulation across different configurations. (D) Comparison of force vector components (F_x, F_y, and F_z) between the real robot and the simulation, with linear fit curves used for calibration.

4.8.4 Task Design and Implementation

For evaluating the agent's capabilities under varying physical objectives, from geometric accuracy to compliant contact behavior, three distinct control tasks were defined.

- **Task 1 (Trajectory Tracking)** targets pure position control in Cartesian

space. The agent must follow a predefined reference trajectory $\mathbf{X}_{\text{ref}}$ composed of 100 3-D coordinates. Two shapes were used for testing different dynamics: a circular path (radius of 25 mm on a plane parallel to the $x - y$ plane) and a star-shaped path (inscribed in a 25 mm radius circle, designed for testing rapid direction changes).

- **Task 2 (Force Control)** focuses solely on force regulation. The agent must apply internal joint commands to generate a desired contact force at the TCP, aligned with a target external force vector $\boldsymbol{\tau}^*_{\text{ext}}$.

- **Task 3 (Hybrid Control)** combines position and force objectives. The agent must follow a tilted elliptical trajectory on a sloped contact surface $\mathbf{S}_{\text{ref}}$ while maintaining a constant normal force of 1 N along the z-axis. The trajectory is defined by the parametric equations:

$$\begin{aligned} x &= 25\cos(\theta_{te}) \\ y &= 25\sin(\theta_{te}) \\ z &= 10\cos(\theta_{te} - \pi) + 10 \quad [\text{unit} : \text{mm}], \end{aligned} \tag{4.28}$$

  where θ_{te} ranges from 0 to 2π. An example of this hybrid control task is visualized in Figure 4.14-(C).

4.8.5 Reward Function Design

A composite reward function was designed for simultaneously guiding the agent toward position tracking and force regulation objectives:

$$R = \alpha R_q + (1 - \alpha) R_F, \tag{4.29}$$

where R_q and R_F are the normalized rewards for position and force, respectively, and $\alpha \in [0, 1]$ is a scalar weight to balance their contributions.

The position reward, R_q, encourages sequential trajectory execution. It is a function of the error $\delta_q = \|\mathbf{X}_P - \mathbf{X}_{\text{ref}}\|_2$, but rewards are conditioned

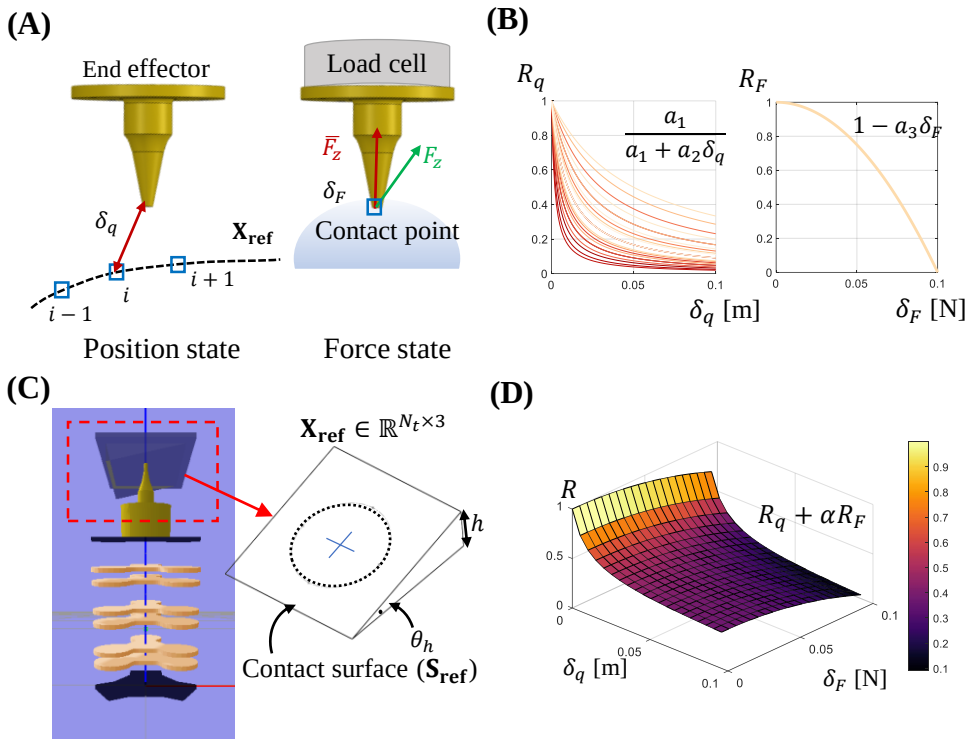


Figure 4.14: (A) (left) Sequential position rewards conditioned on the order of trajectory points, ensuring the agent follows the path in the correct sequence. (right) The force error (δ_F) computed with the target force vector. (B) Reward function shapes for position (R_q) and force (R_F), showing their respective contributions. (C) 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. (D) Manifold of the combined reward function (R), where the weighting factor (α) determines the balance between the position and force rewards.

on the current step index, so only proximity to the *next* target point in the sequence contributes to the return, as visualized in Figure 4.14-(A). The force reward, R_F, is determined by the RMSE δ_F between the measured and

reference contact forces. The specific formulations are:

$$\begin{aligned} R_q &= \frac{1}{1 + 100\delta_q}, \\ R_F &= 1 - 0.5\delta_F. \end{aligned} \tag{4.30}$$

These reward shapes, illustrated in Figure 4.14-(B), were chosen empirically to ensure smooth reward gradients and stable training. No formal optimization was used for determining the reward structure; the described formulations were selected based on a trade-off between positional accuracy and force stability observed during testing. The final profile of the combined reward function is depicted in Figure 4.14-(D).

4.8.6 Policy Learning and Evaluation

For training agents in hybrid control, expert demonstrations and strict precision constraints were incorporated. The agent was required to follow trajectories with an inter-point resolution below 1 mm and achieve a target accuracy of 0.1 mm.

Expert data for behavioral cloning (BC) and offline pretraining was generated using an approximate inverse dynamics model from the FEM simulation. The dataset consists of state-actuation pairs:

$$\text{Expert dataset} = \left\{ \left((\theta_X, \theta_Y, d),\ (x, y, z) \right) \right\}_{i=1}^{N}, \tag{4.31}$$

where (θ_X, θ_Y, d) are the joint inputs and (x, y, z) is the resulting TCP position.

Three learning algorithms were evaluated on the most challenging hybrid task (Task 3): Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and BC. All agents successfully learned stable control policies. As shown in Figure 4.15, the BC-enhanced agent achieved the highest final reward after 100K episodes, confirming the benefit of using expert priors from FEM-driven demonstrations.

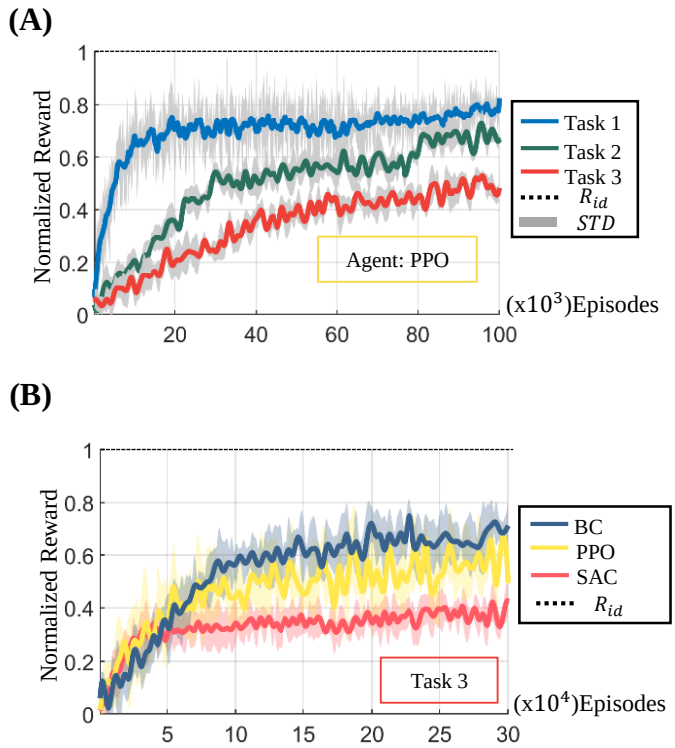


Figure 4.15: (A) Normalized reward progression by episode for Task 3 using the PPO agent. (B) Performance comparison of hybrid control (Task 3) across three agents, showing normalized reward improvements over training episodes.

4.8.7 Sim-to-Real Policy Transfer

The application of a trained policy π^* to the physical robot was accomplished through a **sim-to-real transfer** procedure. The policy, trained in the surrogate model environment, outputs actions as joint angles, $\hat{\theta}_P(t)$. However, the real robot is actuated by pressure inputs. A mapping function f_{SP} was used to bridge this domain gap by converting the policy's joint angle outputs into corresponding pressure profiles:

$$P_R(t) = f_{\text{SP}}(\hat{\theta}_P(t)), \tag{4.32}$$

where $\hat{\theta}_P(t) = \pi^*(\theta|O_t)$ and $P_R(t)$ is the target pressure vector for the real robot.

The calculated pressure profiles were transmitted to the robot's bellows via pressure regulators. A real-time control loop synchronized these commands with the hardware execution. The resulting TCP position and contact forces were measured using a motion capture system and load cells to validate the performance of the transferred policy.

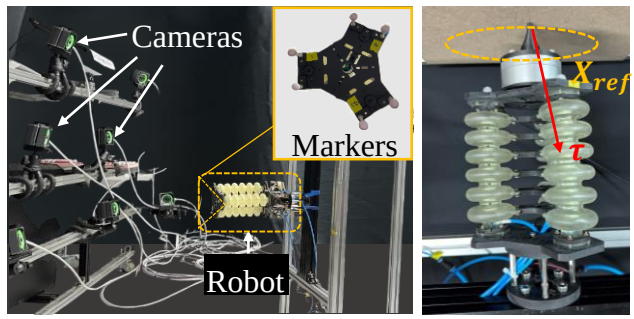
4.9. Experiment

4.9.1 Sim2Real Learning Framework

The proposed learning pipeline was evaluated across three distinct domains: a real soft manipulator, an FEM simulation in SOFA, and a physics-based surrogate model in physics-based simulation. A pneumatic soft manipulator, configured as a parallel mechanism with three bellow actuators between two rigid plates, was used consistently across all platforms. The bottom plate was fixed to a base frame, while the top plate moved freely in response to internal pressure changes. For tasks requiring physical interaction, a load cell and custom tool tips were attached to the top plate to permit external force application and measurement, as depicted in Figure 4.16. The complete experimental setup includes the placement of optical markers for the motion capture system for the ground truth data of the soft manipulator.

Actuation for both the real and simulated systems was constrained to a pressure range of -20 kPa to 35 kPa. This range was selected to ensure structural safety and maintain consistency in the ROM. The surrogate model replicates this ROM through joint angle actuation, which is derived from learned mappings.

The sim2real transfer pipeline is depicted in Figure 4.17-(A). A trained policy, $\pi(\theta_P(t) \mid O_t)$, generates joint commands. These commands are first converted into simulated pressures via the mapping function f_{SP} and subsequently into pressures for the real robot using f_{PR}. The final pressure signals



Motion capture Setup Task environment

Figure 4.16: Motion capture setup used for collecting real-world robot data, incorporating the optical markers and the cameras for precise state estimation.

$P_R(t)$ are smoothed by a moving average filter before being sent to the hardware. While the policy inference loop operated at over 300 Hz, hardware constraints limited the real system's actuation frequency to 2 Hz. The quantitative analysis of the hardware system bandwidth was conducted and reported in the Results section and Figure 4.18.

4.9.2 Data Acquisition and Agent Training

Data collection was performed comprehensively in each domain. In the real system, 5000 distinct pressure configurations within the range of -20 kPa to 30 kPa were applied to each actuator. The system was held at each configuration for stabilization, and the resulting motions were recorded by the motion capture system.

In the FEM domain, pressure inputs (P_S) and external contact forces (τ_{est}) were applied to the model using the same pressure range. Contact forces did not exceed 2 N, corresponding to the manipulator's maximum payload. With a simulation time step of 0.01 s, outputs were generated at 5.56 Hz. This process yielded approximately two million samples, which were used for training the dynamics model in Equation 4.16.

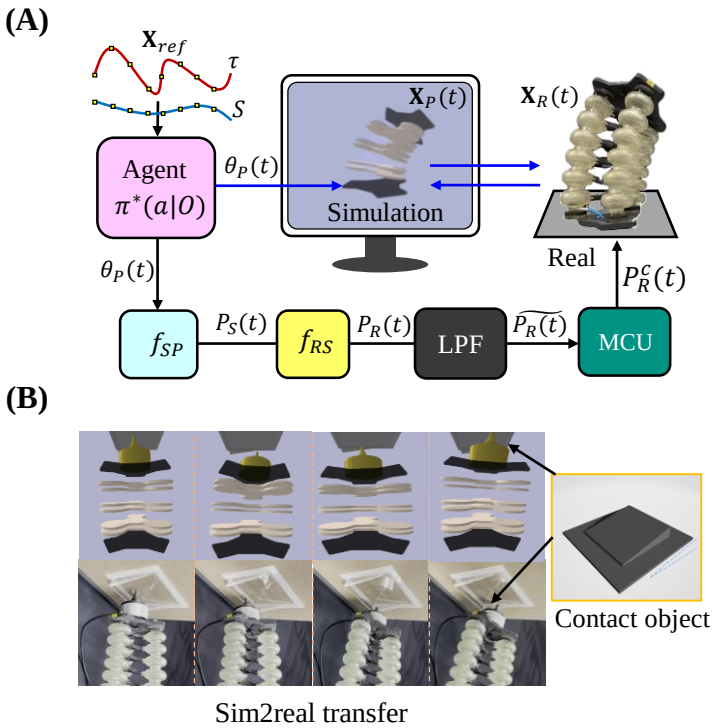


Figure 4.17: (A) Sim2real transfer pipeline illustrating a real-time mapping between the PyBullet simulation and the real robot. (B) Mirrored real-world task implementation setup in the PyBullet environment, including the force measurement units and a contact object.

For the physics-based surrogate model, joint angle limits were derived from the FEM-ROM through the mapping function f_{SP}. The surrogate model simulation was sampled at over 1000 Hz, and each configuration was executed for 20 control steps, enabling the efficient generation of a training dataset for reinforcement learning.

The surrogate model served as the basis for a custom Gym-compatible RL environment. Its observation space was defined to include joint states $(\boldsymbol{\theta}_P, \dot{\boldsymbol{\theta}}_P)$, TCP position and velocity $(\mathbf{X}_P, \dot{\mathbf{X}}_P)$, and the external contact force $(\boldsymbol{\tau}_{\text{ext}})$. The action space was composed of continuous joint commands. Three agent architectures were tested as mentioned: SAC [181],PPO [182], and BC [183]. These agents, representing off-policy, on-policy, and supervised

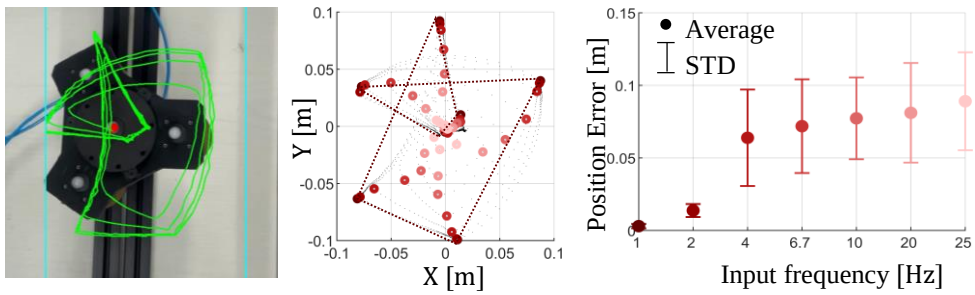


Figure 4.18: Quantitative characterization of the hardware-imposed control bandwidth is presented by plotting the mean position error versus input pressure command frequency.

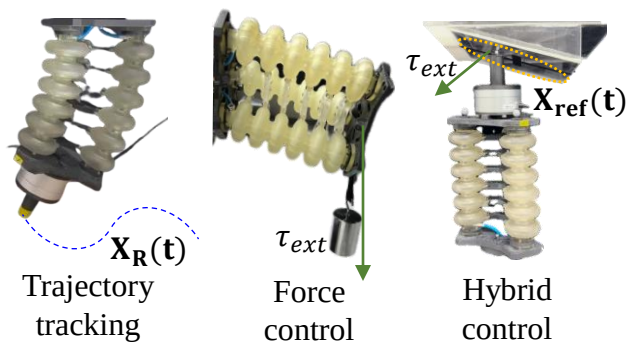


Figure 4.19: Three types of Sim2real tasks: i) trajectory tracking, ii) force control, and iii) hybrid control.

imitation learning methods respectively, were implemented using the stable-baselines3 library [184].

4.9.3 Task Setup and Evaluation

Validation of the framework was performed using three distinct control tasks of increasing complexity, as shown in Figure 4.19. Each task was implemented on both the hardware setup and in the simulation.

The **trajectory tracking task** required the agent to follow predefined

Cartesian paths, such as circular and star-shaped trajectories. A circular path, for instance, was defined by $\mathbf{X}_{\text{ref}}(t) = [r\cos t,\ r\sin t,\ z_0]$. The trained policy's ability to reproduce these paths on the real robot with high spatial accuracy was confirmed by comparing the simulated trajectory $\mathbf{X}_P(t)$ with the real measurement $\mathbf{X}_R(t)$.

The **force-induced deformation estimation task** assessed the agent's ability to predict displacement resulting from external loading. A weight was applied to the TCP, causing the configuration to shift from its unloaded state $\mathbf{X}_{\text{w/o}}$ to a deformed one $\mathbf{X}_{\text{w}}$. The agent's objective was to infer the deformed position $\mathbf{X}_{\text{w}}$ based on the applied force (τ_{ext}) and the current state of the manipulator.

The **hybrid control task** merged trajectory tracking with force regulation. Here, the agent was tasked with following a tilted elliptical path while sustaining a constant normal contact force. The agent achieved accurate performance in both objectives, producing contact force vectors that closely matched the target values throughout the maneuver.

For ensuring effective policy transfer, all tasks were conducted in a mirrored physics-based simulation. This simulation featured identical geometry, contact surfaces, and force measurement instrumentation, as shown in Figure 4.17-(B), to maintain consistency between the simulation and real-world deployments.

4.10. Result

4.10.1 Calibration and Validation

A comparison of the reachable configuration spaces was performed for the real robot, the FEM simulation, and the surrogate model under identical actuation constraints. The resulting Cartesian trajectories, shown in Figure 4.20, indicate that the surrogate model successfully covered the entire reachable space observed in both the FEM simulation and the real robot. This included all positions recorded during physical experiments. A close agreement in the

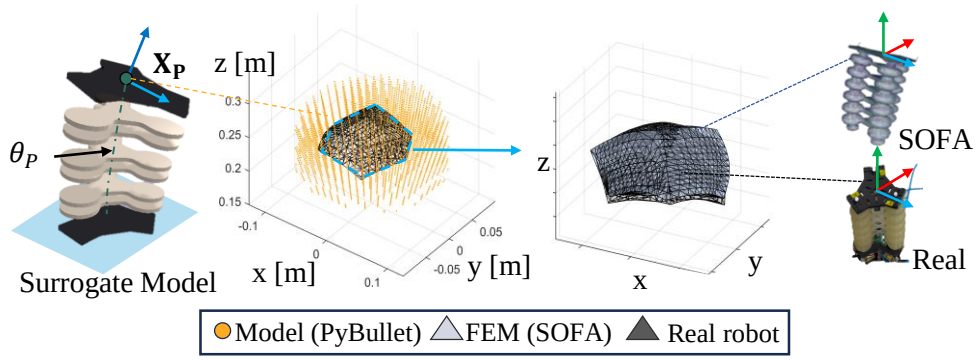


Figure 4.20: Comparison of the reachable configuration space across the three domains: real robot, FEM simulation, and surrogate model. Each boundary represents the range of motion (ROM) achieved under actuation constraints.

range of motion was observed between the FEM model and the real robot.

Differences between the FEM model and the real robot were quantified using five configuration variables: tip position (p_S), tip velocity ($\dot{p}_S$), arc length (l_S), orientation (q_S), and curvature (κ_S). The errors for each variable were normalized using min-max scaling within their domain-specific ranges. Figure 4.21-(A) visualizes the normalized mean errors and standard deviations, with a detailed summary provided in Table 4.1.

Translations between domains were handled by two neural network mappings: f_{RS} (real to simulated pressures) and f_{SP} (simulated pressures to surrogate joint angles). The average prediction error for f_{RS} was found to be 0.13 kPa. For the f_{SP} mapping, the average prediction errors were 0.32 deg for rotational joints ($\theta_{x,y}$) and 4.23 mm for prismatic joints (θ_d), as detailed in **Table 4.1**. Notably, these joint-level errors do not propagate directly to the end-effector's Cartesian position. This outcome is a result of the soft manipulator's kinematic redundancy, where the mapping from a three-dimensional pressure input to a higher-dimensional joint space permits multiple joint configurations to achieve a similar end-effector pose.

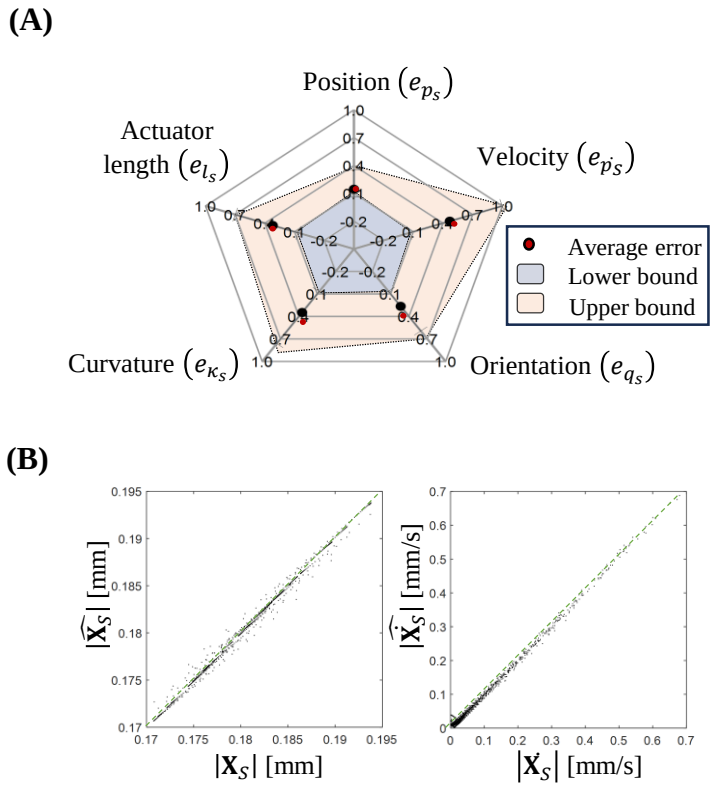


Figure 4.21: (A) Normalized average errors and standard deviations for key configuration variables, comparing FEM simulation and real-world measurements. Metrics include position, velocity, orientation, curvature, and actuator length. (B) Prediction accuracy of the learned forward dynamics model f_S, 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.

4.10.2 Learned Dynamics Model Results

The performance of the learned forward dynamics model (f_S) was assessed using validation sequences from the FEM simulation. The predicted states ($\widehat{\mathbf{X}}_S$) and velocities ($\dot{\widehat{\mathbf{X}}}_S$) over time are presented in Figure 4.21-(B). Quantitative prediction errors, reported in Table 4.1, show mean errors of 3.35 mm for position and 14.56 mm/s for velocity.

The transfer of the learned forward dynamics model f_S to the surrogate's

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
	$\dot{p}_S$	12.69 ± 4.29	mm/s
Dynamics learning (SOFA–PyBullet)	$\hat{\mathbf{X}}_p$	3.35 ± 0.43	mm
	$\widehat{\dot{\mathbf{X}}}_p$	14.56 ± 4.38	mm/s
Inverse dynamics	$\hat{P}_S$	0.38 ± 0.05	kPa
	$\hat{\theta}_p$	0.17 ± 0.04	deg

Table 4.1: Mapping errors between domains

joint space using the mapping f_{SP} is illustrated in Figure 4.22-(A). For a given target state ($\mathbf{X}_S$) and velocity ($\dot{\mathbf{X}}_S$), the corresponding joint configuration ($\widehat{\boldsymbol{\theta}}_P$) was estimated and then applied to the surrogate model.

Actuator pressures for desired motion targets were estimated by the inverse dynamics model. This model predicted the required input pressures ($\widehat{P}_{S,i}$) needed to drive the system to a target state ($\mathbf{X}_S$) and velocity ($\dot{\mathbf{X}}_S$). Figure 4.22-(B) shows the estimated pressures for multiple samples. A mean pressure prediction error of 0.38 kPa and a corresponding joint angle reconstruction error of 0.17 deg are reported in Table 4.1.

4.10.3 Policy Evaluation

An evaluation of the trained reinforcement learning agents was conducted across the three control tasks. Each policy was trained in the customized gym environment with the surrogate model and subsequently transferred to the real robot using the mapping functions f_{SP} and f_{PR} (Figure 4.23). Policies were trained using three different RL algorithms, with reward progressions detailed in the previous RL environment section. For each task, the policy exhibiting

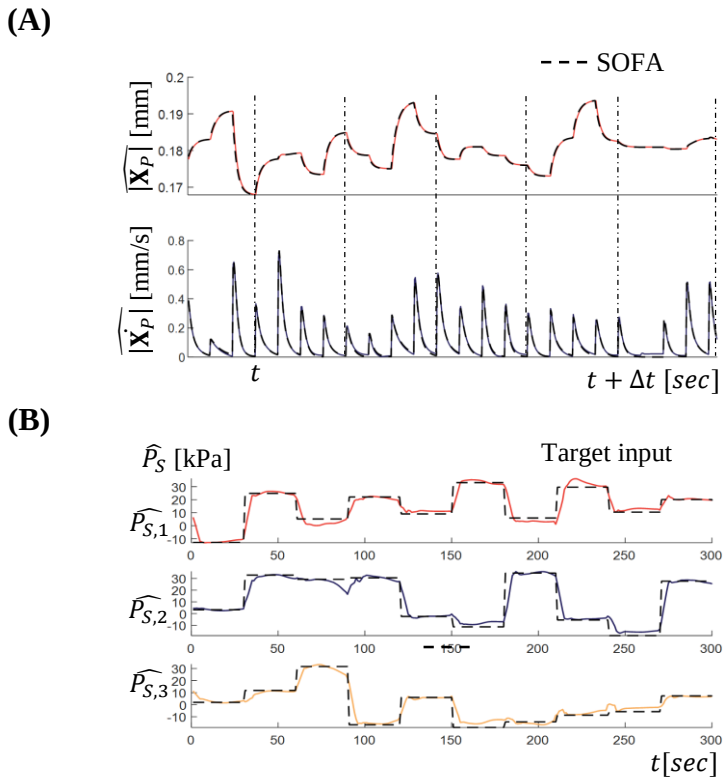


Figure 4.22: (A) Optimized mapping from FEM dynamics to the joint space of the surrogate model, aligning the predicted states $\widehat{\mathbf{X}}_P$ and $\widehat{\dot{\mathbf{X}}}_P$ over time $[t, t + \Delta t]$. (B) Inverse dynamics results predicting actuator pressures $\widehat{P}_S$ from target motion states, demonstrating the capability to reconstruct input pressures from desired trajectories.

the highest final reward was selected for the final evaluation.

Task 1 involved the execution of predefined Cartesian trajectories. The tracking results for both the simulation ($\mathbf{X_P}$) and the real system ($\mathbf{X_R}$) are presented in Figure 4.23. Table 4.2 reports the average tracking errors, with surrogate-level errors of 1.02 mm (circular) and 1.24 mm (star-shaped), and corresponding sim2real errors of 3.38 mm and 6.79 mm.

Task 2 required the agent to estimate the resulting position under external loading (τ_{ext}). Figure 4.24 displays the unloaded state ($\mathbf{X}_{\text{w/o}}$) and the

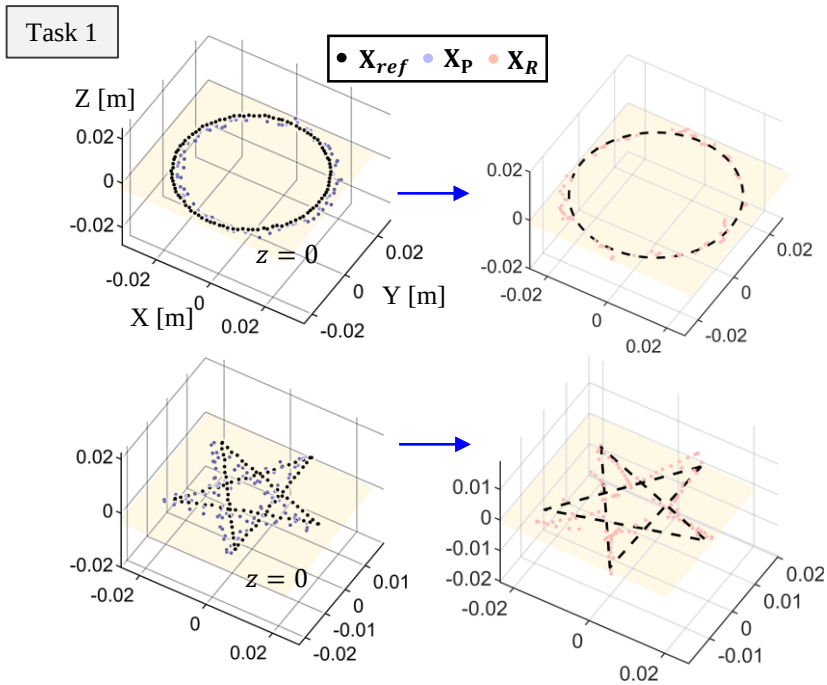


Figure 4.23: Trajectory tracking results for Task 1. The plot on the left shows the prediction of the surrogate model in simulation ($\mathbf{X}_P$, blue), while the plot on the right shows the sim2real results using the real robot ($\mathbf{X}_R$, red). Both results follow the predefined target trajectory $\mathbf{X}_{\text{ref}}$.

target deformed position ($\overline{\mathbf{X}}_{\text{w}}$), with the agent's predictions ($\mathbf{X}_{\text{w}}$) shown from multiple trials. The average position errors relative to the simulation target were 3.20 mm in simulation and 4.68 mm in sim2real, as listed in Table 4.2.

Task 3 combined trajectory following on a tilted ellipse ($\mathbf{X}_{\mathbf{ref}}$) with contact force regulation (F_n). The executed trajectory is shown in Figure 4.25-(A), while the force tracking error and component-wise force profiles are shown in Figures 4.25-(B) and (C). As reported in Table 4.2, simulation errors were 1.35 mm for position and 0.55 N for force, with corresponding sim2real errors of 4.20 mm and 0.70 N.

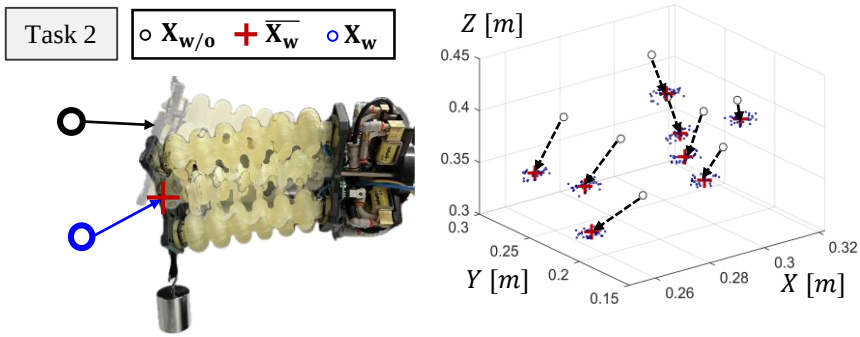


Figure 4.24: Task 2: deformation compensation under external load. The unloaded position $\mathbf{X}_{w/o}$ and the externally deformed position $\overline{\mathbf{X}_w}$ are shown as red crosses, while the corrected position predicted by the agent $\mathbf{X}_w$ is shown as blue circles.

4.10.4 Additional Validation of Performance and Robustness

Three additional experiments were conducted for further validation of the proposed sim2real agent in more challenging scenarios. These experiments, focusing on policy generalization, robustness to disturbances, and system limitations, collectively demonstrate the practical viability of the pipeline.

Generalization to Complex 3D Trajectories The policy's generalization capability was evaluated using a complex, non-planar 3D trajectory not seen during training, which required tracing a figure-eight path on a curved surface. As depicted in Figure 4.26-(A), the real robot successfully tracked this intricate path with precision comparable to Task 1, recording an average tracking error of 5.34 mm. The close alignment among the reference trajectory ($\mathbf{X}_{ref}$), the surrogate model's prediction ($\mathbf{X}_p$), and the real robot's execution ($\mathbf{X}_R$) confirms the framework's ability to generalize effectively.

Robustness to External Disturbances The robustness of the agent as a closed-loop controller was assessed through a physical perturbation experiment. During the circular trajectory task, a disturbance was induced by inten-

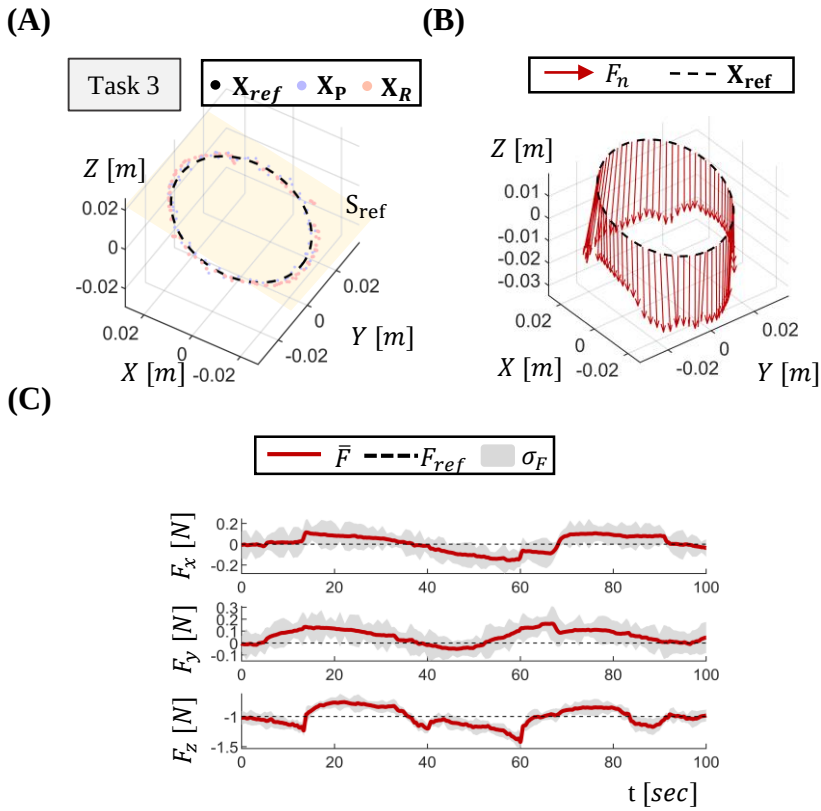


Figure 4.25: (A) Task 3: hybrid control results in which the agent follows a tilted elliptical trajectory $\mathbf{X}_{\text{ref}}$ on the sloped reference surface $\mathbf{S}_{\text{ref}}$, while maintaining contact. (B) Norm-2 force tracking error $\|\mathbf{F} - \mathbf{F}_{\text{ref}}\|$ during Task 3, showing temporal accuracy of the contact force. (C) Component-wise tracking of the target contact force (F_x, F_y, F_z) over time, confirming stable force regulation.

tionally disconnecting the air supply to one pneumatic chamber. The supply was reconnected five seconds later to observe the agent's recovery. As shown in Figure 4.26-(B), this action caused a sharp deviation from the trajectory, followed by a rapid and stable recovery, with the policy converging back to the reference path within five seconds.

Bandwidth Limitation in Policy Rollout The system's operational speed is a critical factor for hardware validation. A quantitative analysis of the

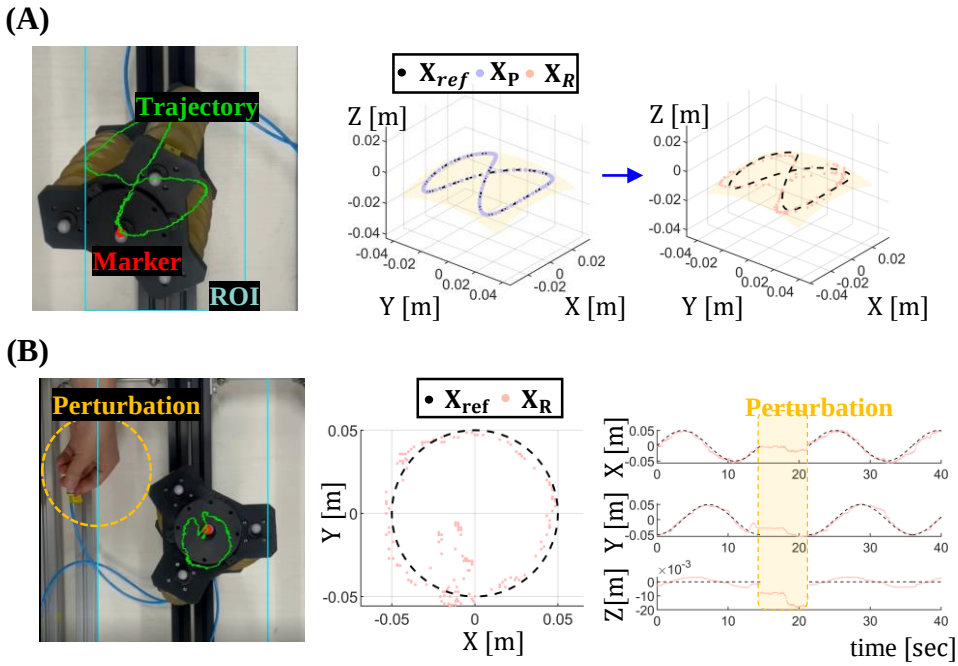


Figure 4.26: (A) Generalizability of sim2real transfer is demonstrated by tracking a complex 3D non-planar figure-eight trajectory not seen during training. (B) Robustness of the RL policy is evaluated by showing successful recovery from an unexpected physical disturbance (temporary pneumatic line disconnection) during a circular trajectory task.

hardware's bandwidth was performed by measuring the mean position error for a point-to-point task while systematically increasing the input command frequency. The results in Figure 4.18 reveal a hardware bottleneck, with a sharp increase in tracking error as the frequency exceeded approximately 2 Hz. This analysis shows that the control speed is constrained by the physical response of the pneumatic hardware, not the policy's inference rate.

System	Task	Target	Value (Mean ± Std)	Unit
RL agent ($\hat{\mathbf{X}}_P$)	Task 1	$\mathbf{X}_{\text{ref}}$: Convex	1.02 ± 0.61	mm
		$\mathbf{X}_{\text{ref}}$: Non Convex	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 ($\hat{\mathbf{X}}_R$)	Task 1	$\mathbf{X}_{\text{ref}}$: Convex	3.38 ± 1.49	mm
		$\mathbf{X}_{\text{ref}}$: Non Convex	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

Table 4.2: Task implementation errors

4.11. Discussion

This study presented an integrated sim2real framework for soft robot control, which connects high-fidelity FEM simulation, dynamics modeling, surrogate abstraction, and reinforcement learning. The design of the framework addresses key challenges in soft robot learning—including nonlinear deformation, high compliance, and actuation redundancy—while ensuring compatibility with standard reinforcement learning toolkits.

The FEM-based model, constructed in SOFA, underwent calibration against real robot measurements obtained from dense pressure sweeps and motion capture data. An MOR formulation using POD enabled efficient simulation while preserving essential deformation modes. The dataset generated from this process supported the training of both forward and inverse dynamics models with a Transformer-PINN architecture.

Real-time simulation and policy training were facilitated by the development of a data-driven surrogate model. This model, implemented in PyBullet with a jointed link structure, successfully replicated the ROM and actuation responses of the FEM simulation. Learned mappings between FEM pressure inputs and surrogate joint angles ensured physical consistency, allowing the

surrogate to be used interchangeably during training while remaining aligned with FEM-based predictions.

Accurate motion prediction and actuation inference within the FEM and surrogate domains were made possible by the forward and inverse dynamics models. The learned dynamics generalized across unseen trajectories and supported the generation of stable behaviors when transferred to the surrogate model. The utilization of actuation-space mappings ensured that predictions from the forward model could be faithfully reproduced by the joint-level abstraction, thereby preserving temporal structure and physical consistency.

Across all evaluated tasks, the learned policies demonstrated robust trajectory execution and force regulation in simulation. Upon transfer to the real robot, the policies successfully maintained their task objectives but exhibited increased errors. These discrepancies arose from physical phenomena not captured in the simulation pipeline, such as 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. Further experiments were performed to probe the framework's real-world capabilities. These results confirmed the policy's ability to generalize to complex, non-planar 3D trajectories not encountered during training. Moreover, the controller demonstrated notable robustness by successfully recovering from a large, intentional physical disturbance, which highlights its effective closed-loop nature.

Several limitations of the current framework remain. The surrogate model, while effective for training, does not explicitly model elastic body interactions or distributed compliance. The incorporation of pseudo-continuum elements or compliant joint structures could enhance realism in contact-rich scenarios. Additionally, the inverse model's assumption of a deterministic mapping may limit its robustness under conditions of uncertain loading or sensor noise.

Future work will involve extending the framework to more contact-rich tasks, such as manipulation, locomotion, or interaction with deformable substrates. Improvements in transfer accuracy may be achieved by enhancing the

fidelity of contact simulation and tactile feedback within the surrogate domain.

Although the focus of this study was a pneumatic manipulator, the architecture is modular and actuator-agnostic, allowing for its application to other systems like fiber-reinforced actuators or tendon-driven arms. In contrast to task-specific pipelines [82, 164, 170, 185] or finely-tuned classical controllers, this approach places a strong emphasis on generalization and adaptability. While a classical controller might achieve superior performance on a single, known task, the strength of the presented RL-based method lies in its generalization capabilities, as shown by its success on unseen tasks. Furthermore, unlike other RL frameworks that often report only qualitative task success, this pipeline provides a pathway for developing controllers for high-precision tasks demanding 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.

4.12. Summary of Part A

The overarching challenge addressed in this dissertation is the simulation gap in deformable robotics, where a fundamental trade-off exists between the physical fidelity required to represent continuum mechanics and the computational speed necessary for data-driven learning. While this challenge pervades the entire robotic system, the functional requirements for simulating the robot's internal structure differ from those of its contact interface. Consequently, Part A of this research has focused specifically on the domain of **action**, investigating methodologies to model the macroscopic dynamics of the deformable body for control and actuation.

To address this challenge, a two-stage methodological progression was pursued. The investigation began in Chapter 3 with a classical approach, deriving a first-principles analytical model for a pneumatically actuated origami manipulator. By leveraging the geometric regularity of the Yoshimura pattern, a closed-form kinematic model was established. While this method successfully

enabled real-time nonlinear control, it revealed critical limitations regarding generality and the reliance on simplifying assumptions, thereby motivating the need for a more scalable, data-driven approach.

Chapter 4 responds to these limitations by introducing a generalizable surrogate modeling framework. This method utilizes high-fidelity FEM simulations as ground truth to learn the complex, non-linear dynamics of the soft body. By abstracting the continuum physics into a virtual kinematic chain, the framework bridges the gap between accurate offline physics and fast online inference, enabling the integration of soft robots into standard rigid-body RL environments.

The key deliverables of Part A include a validated nonlinear controller for origami robots and a high-fidelity surrogate simulation environment capable of training RL policies. These contributions demonstrate that by abstracting continuum mechanics into computationally tractable forms, it is possible to realize fast and physically faithful simulations that enable the learning of robust control policies for deformable manipulators. With the problem of how the robot acts and moves now addressed, the dissertation proceeds to Part B, which tackles the complementary challenge of how the robot perceives the world through the mechanics of the contact interface.

: Modeling Soft Contact Mechanics

Chapter 5.

Modeling the Deformable Contact Interface

5.1. Motivation

The preceding part of this dissertation established methodologies for creating fast and faithful simulation models of a compliant robot's actuator-structure dynamics. While this is a critical component for simulating how a robot acts, a complete model of physical interaction must also account for the boundary where the robot meets the world. The contact interface is as important as the deformation of the body itself. As established in the introduction, there is an increasing demand to synchronize rich force and contact data with the vast kinematic and visual datasets used to train large-scale robot learning models. For a robot to move beyond simple position-based tasks and master complex, contact-rich manipulation, it must be able to reason about the relationship between force and deformation.

Once again, the generation of this essential data requires simulation, and the same trade-off between computational speed and physical fidelity becomes the central challenge. The toolsets developed in Part A, which explored both analytical and data-driven methods for modeling deformable systems, can be adapted and applied to this new problem of analyzing the contact interface.

As reviewed in Chapter 2, a promising hardware-level solution is the advent of vision-based tactile (ViTac) sensors. The increasing use of these sensors in the robotics community, along with the development of tools for analyzing and extracting features from their high-dimensional RGB output, is a significant trend. It is not a coincidence that the tip of a ViTac sensor is designed with a deformable gel. This compliant interface is capable of capturing a wealth of information about the contact event, such as distributed pressure and

fine geometric details, that conventional rigid sensors and coarse simulation models cannot provide. Therefore, the ViTac sensor serves as an excellent physical testbed for developing and validating simulation methodologies for the soft contact interface, which can in turn have a major impact on the future of robot manipulation research.

The limitations of conventional force/torque sensors, which provide only coarse, low-dimensional information about physical interaction, have driven the development of advanced sensor technologies capable of capturing the rich, distributed nature of contact. The most prominent among these are vision-based tactile (ViTac) sensors, such as GelSight and the DIGIT sensor. These sensors provide a powerful hardware solution to the data gap in contact mechanics. By embedding an internal camera to observe the deformation of a soft, elastic surface, they output high-resolution, image-like data that visually encodes a wealth of information about the contact event, including the precise contact geometry, surface texture, and the distribution of forces across the contact patch. The richness of this data has led to their increasing adoption in the robot learning community, enabling the mastery of fine-grained manipulation skills in advanced, data-driven architectures.

The success of these sensors has, in turn, created a strong and urgent demand for simulation environments that can accurately reproduce their high-fidelity outputs for scalable data augmentation and policy training. A variety of rendering simulation approaches have been developed to meet this demand, including methods like TACTO [61] and Taxim [64] . However, a critical limitation persists in this domain. As noted in recent reviews, many of these simulators prioritize visual realism over physical accuracy and fall short of producing physically grounded data that jointly and accurately captures force distribution, contact shape, and surface deformation from first principles.

To address this physical grounding deficit, the work detailed in Chapter 5 of this dissertation introduces a novel bidirectional data pipeline. This framework uses a meticulously calibrated FEM model of the tactile sensor to create a bridge between the physical and digital domains, providing a methodology

to not only generate physically consistent synthetic data from simulation, but also to add dense physical annotations to real-world images. Finally, Chapter 6 takes this a step further. It demonstrates how this high-fidelity data pipeline can be used to create a true data-driven contact model: a Neural Physics Engine (NPE). This learned model directly captures the sensor's complex, soft-body contact dynamics, providing a fast, modular, and robust framework for generating the high-quality, physically grounded contact datasets required to teach the next generation of robot learning models how to master physical interaction.

5.2. Introduction

Robotic policy learning and the execution of complex tasks have been significantly advanced by the availability of large-scale datasets [186–188]. Beyond diverse state observations, rich force feedback from the end effector is recognized as a crucial component for achieving precision and robustness in contact-rich manipulation. This is especially true for platforms like humanoids, which require physical interactions that are both safe and adaptive [9, 189].

In-hand manipulation tasks, which involve multiple simultaneous contacts by a gripper, necessitate high-resolution and spatially dense tactile sensing for the accurate capture of both contact forces and geometry [60, 68, 190]. The information provided by conventional proprioceptive sensors, such as joint torque encoders, is often too coarse, failing to capture localized details such as contact points, shapes, or forces [191, 192].

The limitations inherent in proprioceptive sensing have spurred recent progress in the development of ViTac sensors, which employ embedded cameras to capture the deformations of an elastic surface [12, 193, 194]. Such sensors exhibit superior capabilities for extracting physically meaningful contact information—including contact location, surface geometry, and force distribution—by interpreting visual signals [195–197]. This capability enables fine-grained control in manipulation tasks, such as object reorientation and slip detection [68, 198].

Performance in a variety of downstream tasks has been improved through the use of ViTac sensors, including grasp outcome prediction, material recognition, force estimation, and contact-rich manipulation via end-to-end reinforcement learning [199–203]. Such advances depend on access to large volumes of high-quality tactile data, a requirement that motivates the development of simulation environments capable of reproducing high-fidelity ViTac outputs for scalable data augmentation and policy training [61, 197, 200, 204].

This demand has been addressed by the development of various simulation-based approaches, including methods for realistic tactile image generation, physics-informed perception, and rendering within physics-based simulators [61, 64, 66, 68, 197, 205, 206]. Although these methods have led to advancements in tactile rendering and representation learning, the majority are unable to produce physically grounded tactile data that jointly captures force distribution, contact shape, and surface deformation. Finite Element Methods (FEM) can model these interactions with high fidelity, yet their computational expense often renders them impractical for real-time control or accelerated reinforcement learning [207, 208]. Furthermore, the focus of existing work is typically on perception rather than generation, with only a few methods rendering realistic RGB outputs that closely match actual sensor responses [61, 64, 209, 210].

The present study introduces a simulation rendering framework, based on an optimized FEM implementation, for the physically accurate and high-resolution modeling of ViTac sensors. The simulator is designed to compute contact-induced surface deformation and force distribution, thereby producing large-scale datasets with physically grounded annotations. Simulated responses are aligned with real sensor measurements through a calibration procedure that covers various contact shapes and loading conditions. This calibrated model is then used to construct paired datasets by matching simulated nodal outputs with real RGB images from mirrored indentation experiments. The resulting datasets facilitate the training of two bidirectional networks. The first is a perception model for estimating deformation and force from an RGB input, while the second is a rendering model for synthesizing realistic tactile

images from physical-state representations. Collectively, these networks allow for the interpretable annotation of real sensor data and the physics-consistent generation of RGB images from simulation, supporting both scalable data augmentation and simulation-based learning.

5.3. Data Generation Pipeline

The proposed simulation and learning framework, illustrated in Figure 5.1, establishes an integrated ecosystem that connects high-fidelity, FEM-based simulation with a bidirectional mapping between real and simulated tactile signals. At the core of this framework is a detailed physical model of the DIGIT sensor [60], which was developed and calibrated within the SOFA simulation environment [40]. This calibration process utilized a minimal set of real-world indentation experiments to accurately characterize and replicate the complex hyperelastic behavior of the sensor's silicone gel tip. The resulting model employs a fine-resolution tetrahedral mesh, specifically designed to capture nodal displacements and contact forces with a spatial fidelity that matches the physical device. This high-resolution representation is critical for resolving the subtle, high-frequency surface deformations that constitute rich tactile information.

A primary challenge with high-fidelity FEM is its computational expense, which renders it impractical for large-scale data generation. As detailed in Chapter 4, this challenge was addressed using similar MOR [41, 44] methods to reduce the mesh size of the gel tip of the ViTac sensor. The application of MOR involves projecting the system's dynamics onto a lower-dimensional subspace that captures the dominant modes of deformation. This approach achieves a significant acceleration in simulation speed without compromising the underlying physical accuracy, making it feasible to generate extensive datasets in a fraction of the time.

A meticulously controlled procedure was used to construct a paired dataset by reproducing identical contact conditions in both the physical setup and the accelerated simulation. This process yielded a rich collection of cor-

responding data tuples, where each real-world RGB image (I_R) is directly matched with its simulated physical annotations, including the 3D surface deformation field (U) and the corresponding force distribution (F). This paired dataset serves as the foundation for training two complementary deep neural networks:

1. A **perception model**, which performs an analysis task by inferring the physically grounded states (U and F) from a single real RGB tactile image (I_R).

2. A **rendering model**, which performs a synthesis task by generating a realistic, high-fidelity RGB tactile image from a given set of simulated physical contact states.

Together, these components create a powerful, real-time, bidirectional translation engine between the physical (simulation) and visual (real-world) tactile domains. This synergy enables two critical capabilities: the physics-informed annotation of real sensor data for improved interpretability, and the high-fidelity synthesis of realistic tactile images within accelerated simulation environments for scalable, simulation-based policy learning.

5.4. High-Fidelity Tactile Simulation

5.4.1 Material Calibration and Mesh Preparation

The development of an accurate FEM simulation for the Vision-based Tactile (ViTac) sensor necessitated the precise modeling of two key aspects: (i) the intrinsic material properties of the deformable gel tip and (ii) its complex deformation behavior under contact. The foundation of this process was a rigorous material calibration procedure. Force-displacement measurements were collected under quasi-static loading conditions using a universal testing machine. This quasi-static approach was crucial for isolating the hyperelastic properties of the silicone gel from time-dependent viscoelastic effects. The resulting empirical data were then used to fit the parameters of a suitable

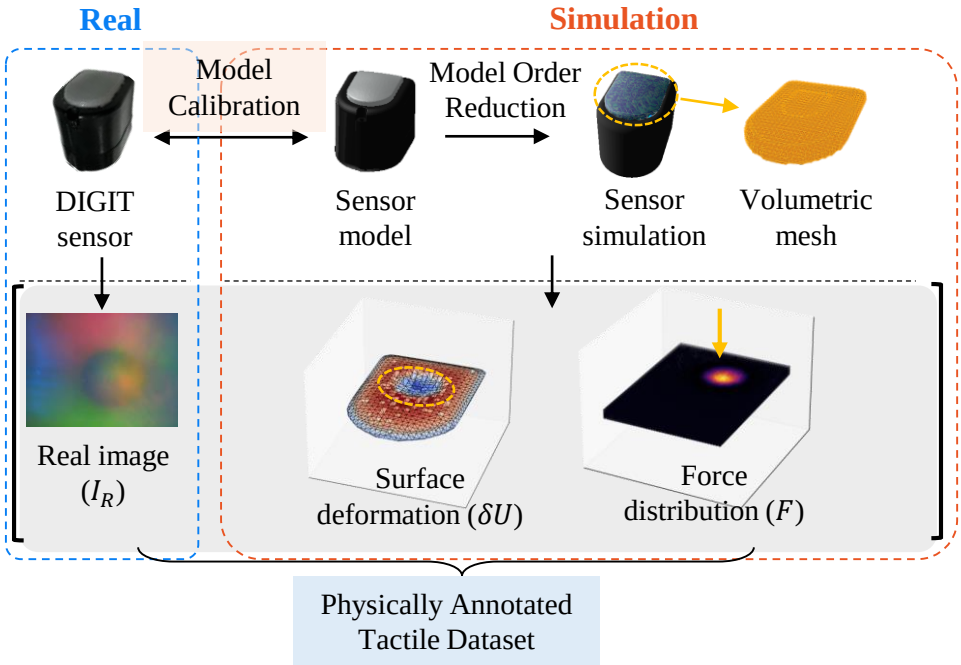


Figure 5.1: Structure of the bidirectional framework linking real and simulated outputs from vision-based tactile sensors. A mirrored setup was used to collect paired data between physical indentation and simulation (gray region).

hyperelastic material model such as Mooney-Rivlin or Neo-Hookean, which govern the relationship between stress and strain within the FEM simulation.

With the material properties defined, the simulation geometry was prepared. The 3D CAD models of the sensor body and the indenter were discretized into high-resolution tetrahedral meshes suitable for FEM analysis. Special attention was given to the mesh quality at the contact interface, as this region experiences high stress and strain gradients. The initial contact surface region of the sensor mesh was locally refined to a higher density, a step that improves the numerical stability of the contact solver and reduces surface noise in the resulting deformation field. Subsequently, iterative Laplacian smoothing [211] was applied to the mesh vertices. This preprocessing step improves the quality and aspect ratio of the tetrahedral elements, enhancing the overall robustness of the simulation by preventing issues like element

inversion during large deformations. The final, preprocessed mesh was then used as the geometric basis for the FEM simulation.

5.4.2 Real-Time Simulation via Model Order Reduction

While high-fidelity FEM provides physical accuracy, its computational cost, stemming from a system with tens of thousands of DOFs, is prohibitive for real-time applications or large-scale data generation. To address this bottleneck, MOR was applied using the Proper Orthogonal Decomposition method [44]. Let $\vec{u}_z(t) \in \mathbb{R}^{N\times 1}$ denote the full-order normal displacement vector at time t, where N is the total number of FEM nodes. The static equilibrium is governed by:

$$\vec{K}(\vec{u_z}) = \vec{f}, \tag{5.1}$$

where $\vec{K}$ is the nonlinear stiffness operator and $\vec{f}$ is the external force vector. The MOR process begins by collecting a sequence of representative deformation "snapshots" from a full-order simulation into a snapshot matrix:

$$\vec{U} = [\vec{u}_{z,0}, \ldots, \vec{u}_{z,T-1}] \in \mathbb{R}^{N\times T}. \tag{5.2}$$

Using singular value decomposition (SVD), this matrix is factorized to extract a set of orthogonal basis vectors that represent the dominant modes of deformation:

$$\vec{U} = \mathbf{\Phi}\mathbf{\Sigma}\vec{V}^{\top}. \tag{5.3}$$

By retaining only the top-r most significant basis vectors (where $r \ll N$), a reduced subspace $\mathbf{\Phi} \in \mathbb{R}^{N\times r}$ is formed. The full displacement field can then be approximated as a linear combination of these basis vectors:

$$\vec{u_z} \approx \mathbf{\Phi}\vec{q}, \tag{5.4}$$

where $\vec{q} \in \mathbb{R}^r$ is the reduced coordinate vector. Substituting this approximation into the original system and projecting onto the reduced basis yields the

reduced-order system:

$$\mathbf{\Phi}^\top \vec{K} \mathbf{\Phi} \vec{q} = \mathbf{\Phi}^\top \vec{f}. \tag{5.5}$$

This reduced system, which solves for the small vector $\vec{q}$ instead of the large vector $\vec{u_z}$, preserves the dominant physical behavior while reducing the computational cost by several orders of magnitude, thus enabling real-time execution.

5.4.3 Integrated Simulation Scene in SOFA

As shown in Figure 5.2, these components were integrated into a custom simulation scene within the SOFA framework. The scene was constructed to meticulously reproduce the contact-induced deformation of the ViTac sensor using the reduced FEM model ($M^{(k)}$). The sensor and indenter were represented as separate tetrahedral meshes, with the sensor's contact region featuring the refined mesh for enhanced resolution. During each simulation step, the indenter follows a predefined perpendicular trajectory toward the sensor. Collision detection algorithms identify intersections between the two bodies, and a contact solver computes the resulting interaction forces, accounting for frictional effects. These forces then drive the deformation of the reduced-order model. The primary output is the nodal displacement field (U), a rich, high-dimensional representation of the sensor's physical state. This field was subsequently processed to extract key physical annotations—such as contact location, contact patch area (A), and force distribution ($F(U)$)—which serve as the ground truth data for calibrating the system and training the perception and rendering networks.

5.5. Calibration Metrics

The calibration process is fundamental to bridging the gap between the simulated model and its real-world counterpart, effectively creating a digital twin of the sensor. This was accomplished by reproducing identical contact conditions—such as indenter shape, depth, and orientation—across both the physical DIGIT sensor and the FEM simulation. This parallel setup enables a direct and

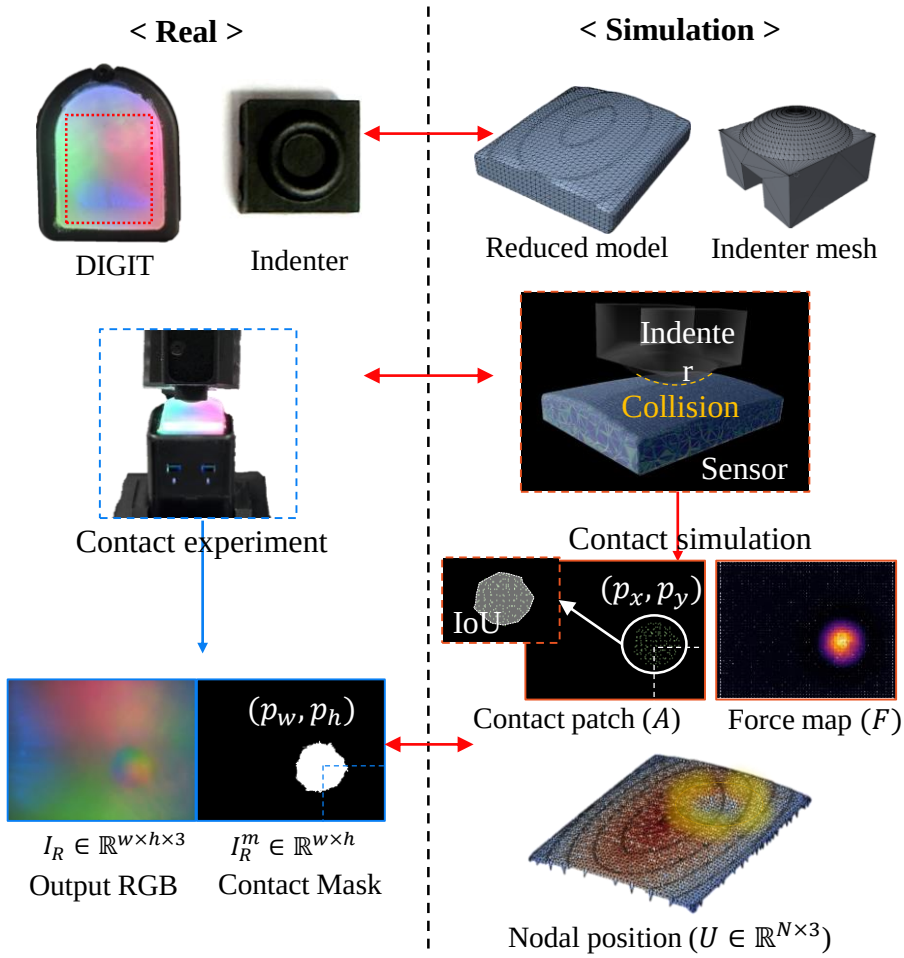


Figure 5.2: Overview of the FEM model calibration process and evaluation metrics. Simulation in SOFA replicates real-world contact to generate surface deformation (U) and corresponding force distribution (F) for model calibration. From the simulated nodal displacement (U), the contact location (p), contact patch (A), and force map (F) are extracted and compared with the real sensor measurements.

quantitative comparison of their geometric and physical responses, ensuring that the simulation is a faithful representation of reality. The following metrics were used to quantify the fidelity of the calibration.

5.5.1 Nodal Position Error

The geometric accuracy of the simulation was first evaluated by quantifying the nodal position error. This metric represents the mean Euclidean distance between the 3D coordinates of surface points measured on the physical sensor (typically via Digital Image Correlation) and the corresponding nodal positions predicted by the FEM simulation under identical indentation conditions. The error is reported as the RMSE, which provides a single aggregate measure of the simulation's ability to replicate the ground-truth surface deformation across the entire contact region.

5.5.2 Contact Patch Geometry

Beyond the 3D surface geometry, the fidelity of the 2D contact footprint was assessed using several metrics. The primary metric, Intersection over Union (IoU), was used to quantify the spatial overlap between the real and simulated contact patches, as depicted in Figure 5.2. The ground-truth contact mask was extracted from the sensor's RGB image (I_R) via background subtraction and image thresholding, while the simulated mask was generated by applying a threshold to the vertical displacement field (δU_z) of the surface nodes. To provide a more granular analysis of any misalignment, the contact centroid error (Euclidean distance between mask centers) and the rotation error (angular difference between their principal axes) were also computed.

5.5.3 Force-Depth Correspondence

The physical realism of the simulation was validated by evaluating the force-depth correspondence. This metric quantifies the agreement between the force-displacement curves obtained from physical indentation trials and those generated by the simulation, ensuring the model's mechanical response matches reality. The total simulated contact force in the vertical (z) direction was computed by integrating the stress over the contact area, derived from the calibrated Neo-Hookean material model. The strain energy density function, W, defines

the potential energy stored in the material as a function of its deformation:

$$W = \frac{\mu}{2}(\bar{I}_1 - 3) + \frac{\kappa}{2}(J-1)^2, \tag{5.6}$$

where μ and κ are the calibrated shear and bulk moduli derived from Young's modulus E and Poisson's ratio ν, and $\bar{I}_1$ and J are strain invariants. From the strain energy, the Cauchy stress tensor $\sigma(U)$ is computed:

$$\sigma(U) = \frac{\mu}{J}(\vec{B} - \vec{I}) + \kappa(J-1)\vec{I}, \tag{5.7}$$

where $\vec{B}$ is the left Cauchy-Green deformation tensor. Finally, the discrete force on each node i, f_i^z, is calculated by projecting the stress onto the nodal surface area:

$$f_i^z(U) = \left[\sigma(U) \cdot \vec{n}_i\right]_z A_i, \tag{5.8}$$

where A_i is the lumped surface area of node i and $\vec{n}_i$ is its outward surface normal. The total simulated force, obtained by summing f_i^z over all nodes within the contact patch, was compared against the force measured by the universal testing machine. The final reported metric is the RMSE between these two force profiles across the full range of indentation depths.

5.6. Bidirectional Tactile Perception and Rendering Networks

This work introduces two complementary networks that form a bidirectional bridge between visual tactile imagery and their underlying physical states. The first, a perception network, infers physical deformation from an RGB image. The second, a rendering network, synthesizes a realistic RGB image from a physical deformation field.

5.6.1 Perception Network: From Visual to Physical State

The perception network, shown in Figure. 5.3, is designed to solve the inverse problem: estimating a dense physical displacement field from a single RGB

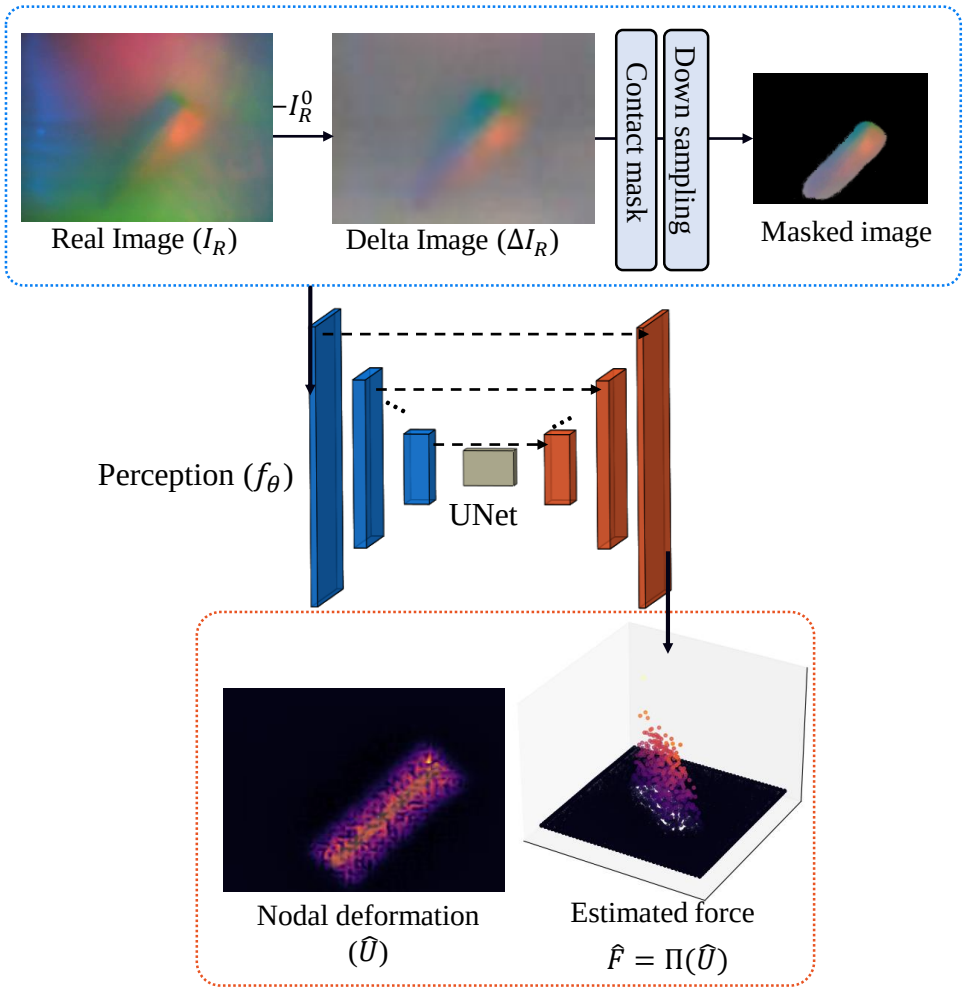


Figure 5.3: Perception model (f_θ): given an input RGB image (I_R), the network predicts a nodal deformation field ($\hat{U}$), consisting of 3D displacement at each node. The output nodal deformation can be projected to form a spatial force map $\hat{F}$ using the sensor model.

tactile observation. This capability is essential for robotic applications that require a spatial understanding of contact surfaces from visual data.

Preprocessing and Data Handling As the training dataset was collected from multiple DIGIT sensors, a preprocessing step is applied to standardize the

input. A sensor-specific background image (I_R^0) is subtracted from each tactile frame (I_R) to produce a delta image, $\Delta I \in \mathbb{R}^{3\times240\times320}$. This process removes static appearance variations and isolates the changes caused by contact. The target for this network, the ground-truth displacement field U_z, is obtained by rasterizing the nodal outputs from the FEM simulation onto the image grid using calibrated sensor bounds.

Network Architecture and Training The network employs a compact U-Net encoder-decoder architecture. The design features a symmetric structure with three encoding levels that process an increasing number of channels $(32, 64, 128)$, each utilizing batch normalization and ReLU activations. Corresponding decoder levels use skip connections to preserve fine-grained spatial details from the encoder. The forward pass can be described as:

$$E_1 = \text{ReLU}(\text{BN}(\text{Conv}_{3\to32}(I))) \tag{5.9}$$

$$E_2 = \text{ReLU}(\text{BN}(\text{Conv}_{32\to64}(\text{Pool}(E_1)))) \tag{5.10}$$

$$E_3 = \text{ReLU}(\text{BN}(\text{Conv}_{64\to128}(\text{Pool}(E_2)))) \tag{5.11}$$

$$D_3 = \text{ReLU}(\text{BN}(\text{Conv}_{128\to64}(\text{Concat}[\text{Up}(E_3), E_2]))) \tag{5.12}$$

$$D_2 = \text{ReLU}(\text{BN}(\text{Conv}_{64\to32}(\text{Concat}[\text{Up}(D_3), E_1]))) \tag{5.13}$$

$$\hat{D} = \text{Conv}_{32\to3}(D_2) \tag{5.14}$$

where Concat$[\cdot]$ denotes channel concatenation and Up$(\cdot)$ represents transposed convolution. The network's output is a three-channel physical field $\hat{F} = [p_x, p_y, \hat{U}_z]$, where p_x and p_y are normalized coordinates providing auxiliary structural context.

The training process focuses specifically on the out-of-plane displacement channel ($\hat{U}_z$) using an MSE loss function:

$$\mathcal{L}_{\text{disp}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{U}_{z,i} - U_{z,i})^2. \tag{5.15}$$

The training pipeline incorporates global normalization of the displacement data based on dataset statistics, with $dz_{min} = -8.0 \times 10^{-4}$ and $dz_{max} = -3.0 \times 10^{-5}$ meters, ensuring consistent training dynamics.

Evaluation Metrics The performance of the perception network is quantified using metrics that assess the geometric accuracy of the predicted contact region: Contact IoU, Contact Centroid Error, and Contact Rotation Error.

5.6.2 Rendering Network: From Physical to Visual State

The rendering network, depicted in Figure 5.4, addresses the forward problem: synthesizing a realistic RGB delta image $\Delta\hat{I}_S$ that visually corresponds to a given physical contact state. This enables the creation of large-scale, physically grounded synthetic datasets for training downstream policies.

Input Preprocessing with Gaussian Splatting The network processes a multi-channel input derived from the sparse nodal data of the FEM simulation. A sophisticated data preprocessing pipeline converts this sparse data into a dense spatial representation. The Gaussian splatting technique [212] is used to transform sparse nodal displacements (x_i, y_i, dz_i) into a dense field using the kernel:

$$K(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma_{px}^2}\right), \tag{5.16}$$

where the Gaussian standard deviation is set to $\sigma_{px} = 1.5$ pixels. The displacement values are normalized to the range $[-1, 1]$:

$$dz_{norm} = \text{clip}\left(\frac{dz}{dz_{scale}}, -1, 1\right), \quad \text{with } dz_{scale} = 1 \text{ mm}. \tag{5.17}$$

The final input tensor combines the dense displacement field, coordinate features (x, y), and Fourier positional encodings with four frequency components

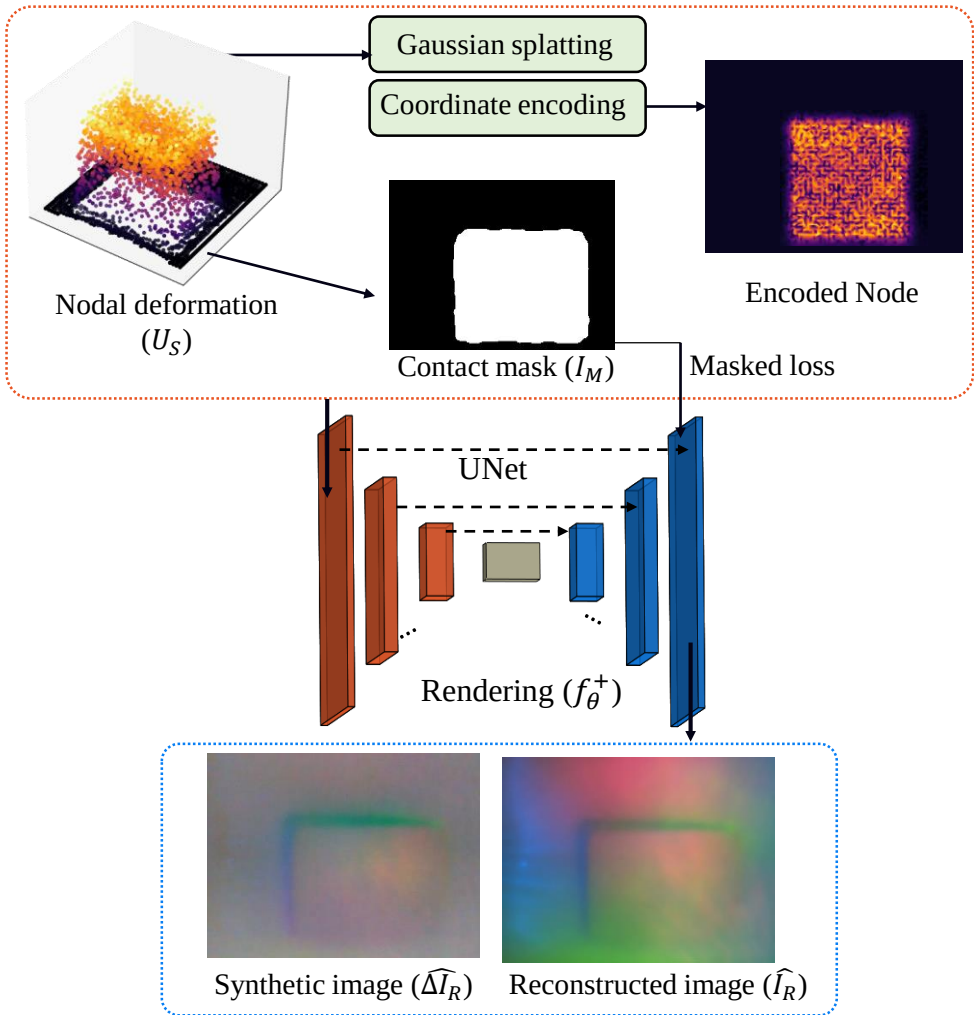


Figure 5.4: Rendering model (f_θ^+): given a physical state input $U_S \in \mathbb{R}^{N\times 3}$, the network reconstructs a synthetic RGB image $\Delta\hat{I}_R$ that mimics the real sensor image.

to provide the network with multi-scale spatial information:

$$\mathrm{PE}(p,k) = \begin{cases} \sin(2\pi k \cdot p_{norm}) & \text{for even indices} \\ \cos(2\pi k \cdot p_{norm}) & \text{for odd indices} \end{cases}, \quad k \in \{1,2,3,4\}. \quad (5.18)$$

Network Architecture and Training The architecture is a lightweight **U-Net** with two encoder stages, a bottleneck, and a symmetric decoder. It utilizes Group Normalization and Gaussian Error Linear Unit (GELU) activations [213] for improved training stability. The final layer employs a tanh activation to produce the RGB delta image in the range $[-1, 1]$.

Training is performed using a sophisticated masked L1 loss function that intelligently weighs different regions of the image. The total loss is a sum of losses for the contact, background, and neutral regions:

$$\mathcal{L} = \mathcal{L}_{\text{contact}} + 0.1 \cdot \mathcal{L}_{\text{bg}} + 0.02 \cdot \mathcal{L}_{\text{neutral}}, \tag{5.19}$$

where each component is a masked L1 loss designed to focus the model on physically meaningful contact regions while appropriately penalizing errors in the background:

$$\mathcal{L}_{\text{contact}} = \frac{\sum_{i,j} M_{i,j} \cdot ||\hat{I}_{i,j} - I_{i,j}||_1}{\sum_{i,j} M_{i,j} + \epsilon}, \tag{5.20}$$

$$\mathcal{L}_{\text{bg}} = \frac{\sum_{i,j} (1 - M_{i,j}) \cdot ||\hat{I}_{i,j} - I_{i,j}||_1}{\sum_{i,j} (1 - M_{i,j}) + \epsilon}, \tag{5.21}$$

$$\mathcal{L}_{\text{neutral}} = \text{mean}(|(1 - M_{i,j}) \cdot \hat{I}_{i,j}|). \tag{5.22}$$

Here, the contact mask M is derived from the simulated vertical displacement δU_z.

Evaluation Metrics The quality of the rendered images is evaluated with three standard metrics:

- L1 Loss: The mean absolute pixel difference between the predicted and ground-truth images.
- Peak Signal-to-Noise Ratio (PSNR): A measure of reconstruction quality based on pixel-wise error, calculated as $\text{PSNR} = 20 \log_{10} \left(\frac{\text{MAX}}{\sqrt{\text{MSE}}} \right)$.
- Structural Similarity Index Measure (SSIM): A perception-based metric

that compares luminance, contrast, and structural similarity.

The predicted delta image $\Delta\hat{I}_R$ can be converted to the final tactile image $\hat{I}_R$ by adding back the sensor-specific background I_R^0.

5.6.3 Hyperparameter Summary

A summary of the key hyperparameters for both networks is provided in Table 5.1.

Component	**Perception Network (f_θ)**	**Rendering Network (f_θ^+)**
Input	RGB Delta Image (ΔI_R)	Physical State ($\hat{U_S}$) with Positional Encodings
Output	Physical State (U_S)	RGB Delta Image (ΔI_S)
Architecture	UNet (3-level)	UNet (2-level)
Activations / Norm	ReLU + BatchNorm	GELU + GroupNorm
Loss Function	MSE on displacement channel (dz)	Masked and Weighted L1 Loss
Optimizer	Adam (1×10^{-5})	Adam (1×10^{-5})

Table 5.1: Hyperparameters of the two bidirectional networks.

5.7. Experimental Methodology

5.7.1 Material Calibration Procedure

The physical basis for the simulation is a hyperelastic material model whose parameters must be precisely identified through empirical characterization. This calibration was performed by collecting high-resolution force-displacement data from the physical DIGIT sensor. As illustrated in Figure 5.5, a planar stage was used to ensure precise alignment between the indenter tip and the specific region of interest (ROI) on the sensor surface. This alignment is critical for eliminating off-axis moments and shear forces, thereby isolating the pure compressive response of the material.

The indentation tests were conducted using a universal testing machine (34SC, Instron), which provides high precision with a force resolution of 10 mN and a displacement accuracy of 20 μm. Measurements were taken at 100 uniformly distributed locations across the sensor's ROI to construct

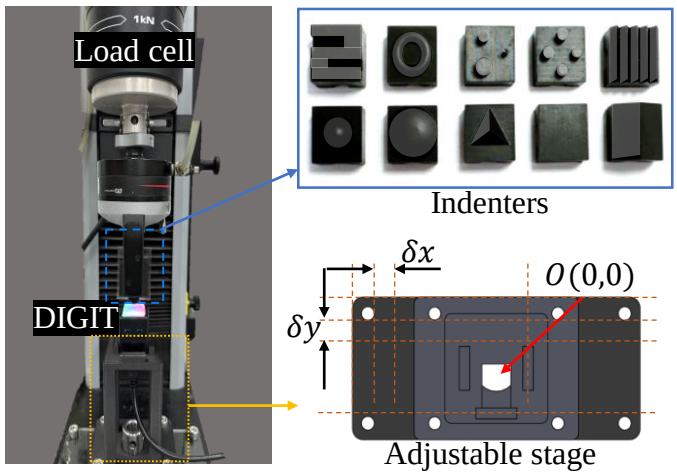


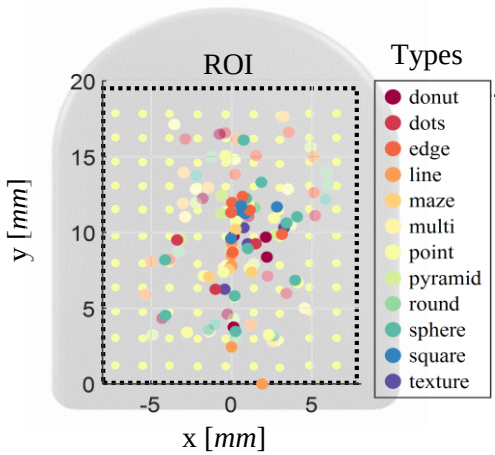
Figure 5.5: Experimental setup using a universal testing machine and ten different indenter tips to press against the vision-based tactile sensor. The tips represent a variety of geometric shapes.

a comprehensive spatial map of the material response and account for any minor non-uniformities in the gel. At each location, a complete indentation-retraction cycle was performed in a quasi-static mode, using a constant, slow indentation speed of 0.01 mm/s up to a maximum depth of 1 mm. Preliminary experiments confirmed that indentation speeds exceeding 0.5 mm/s introduced rate-dependent viscoelastic effects. By operating well below this threshold, the collected data predominantly reflects the hyperelastic properties of the silicone, which is consistent with the constitutive model used in the FEM simulation.

5.7.2 Paired Dataset Acquisition

Following calibration, a large and diverse dataset was acquired for training the bidirectional perception and rendering networks. The objective was to create a comprehensive dataset that spans a wide distribution of contact geometries, pressures, and sensor-specific variations, which is essential for training robust and generalizable models.

The data collection involved real-world indentation tests using 10 different



Rotation	Count
0	10000
$\pi/4$	1250
$\pi/2$	1250
$3\pi/4$	1250
π	1250

Figure 5.6: Contact ROI on the sensor surface, along with the number of rotated indenter configurations used for data collection. For asymmetric indenters, rotations were also applied to collect various contact patch.

indenter geometries, as shown in Figure 5.6. These shapes were selected to represent a variety of geometric primitives encountered in manipulation tasks, ranging from sharp points and edges to smooth curves and complex textures. For each indenter, a step-wise indentation was performed in 100 increments of 10 μm each, up to a maximum depth of 1 mm. The maximum depth was carefully chosen to ensure that the resulting contact force remained below the sensor's safe operational limit of 15 N.

To maximize dataset diversity and prevent the models from overfitting, a systematic variation strategy was employed. Asymmetric indenters (e.g., maze, pyramid) were physically rotated between trials to generate a rich variety of contact patterns and pressure distributions from a single object. For symmetric indenters (e.g., point, sphere), the contact position on the sensor was varied instead, as depicted in Figure 5.6. Furthermore, to ensure the trained models are robust to manufacturing tolerances, the entire set of experiments was repeated using four different physical DIGIT sensors. This practice forces the models to learn the general principles of contact mechanics rather than memorizing the idiosyncratic response of a single sensor.

In total, 15,000 real contact images were acquired. For each real-world

measurement, a corresponding FEM simulation was executed under identical conditions (indenter geometry, position, orientation, and depth). This process yielded a perfectly paired dataset, where each real RGB image is matched with a high-fidelity physical annotation from the simulation, containing detailed nodal deformation and contact pressure information. This one-to-one correspondence is the key enabler for the supervised training of the bidirectional networks. A total of 15% of this dataset was strictly held out as a validation set, used exclusively for evaluating the final inference performance of the trained models on unseen data.

5.8. Result

5.8.1 Calibration Precision

The fidelity of the calibrated Finite Element Method (FEM) simulation was quantitatively validated by reproducing identical indentation sequences in both the physical and simulated setups. This validation ensures that the simulation serves as a high-fidelity digital twin of the physical DIGIT sensor. The key calibration results, summarized in Table 5.2, quantify the correspondence between the two domains in terms of both physical force response and geometric deformation.

The global force error, reported as the Root Mean Square Error (RMSE) between the simulated and measured force-displacement curves, averaged 0.20 N across all indenters, with a maximum deviation not exceeding 0.30 N. This low error, relative to the sensor's operational force range, indicates a strong correspondence in the modeled material behavior. The errors were observed to be smallest in the central region of the sensor and increased slightly toward the edges, a phenomenon attributed to the complex boundary conditions and higher stress concentrations at the perimeter of the convex silicone gel tip.

The geometric accuracy was assessed via the nodal position RMSE, computed as the average Euclidean distance between simulated and measured

nodal coordinates, resulting in a low error of 0.192 mm. This confirms that the simulation accurately captures the three-dimensional shape of the surface deformation. All reported values are averaged over ten distinct indenter geometries, with statistics calculated from a set of 100 uniformly distributed indentation points per indenter. The SOFA-based reduced-order FEM simulation achieves an execution speed of approximately 30 frames per second (FPS) on a 32-core, 64-thread CPU, demonstrating its suitability for generating large datasets efficiently.

Domain	**Metric**	**Value**
FEM Calibration	Force RMSE [N]↓	0.20 ± 0.030
	Nodal position [mm]↓	0.192 ± 0.017
Network Perception	Contact IoU↑	0.85 ± 0.055
	Contact centroid [mm]↓	0.458 ± 0.078
	Contact rotation [rad]↓	0.047 ± 0.014
	Surface force [N]↓	0.11 ± 0.037
Rendering Quality	L1↓	13.59 ± 1.03
	SSIM↑	0.971 ± 0.02
	PSNR↑	39.13 ± 1.81

Table 5.2: Summary of quantitative results for calibration, perception, and rendering tasks. The arrows (↑) indicate that higher values are better, while (↓) indicates that lower values are better.

5.8.2 Perception Network Validation

The perception network, tasked with inferring physical contact states from raw visual data, was evaluated on the held-out validation dataset. The evaluation confirmed its ability to reliably recover both the spatial structure and the physical response of real-world contacts. The similarity of the contact shape, quantified by the Intersection over Union (IoU) between real and predicted contact patches, yielded a high mean value of 0.85 ± 0.05. The positional accuracy of the contact was also high, with a centroid error of only 0.458 ± 0.078 mm and a minimal rotation error of 0.047 ± 0.01 rad.

A key capability of this framework is the estimation of physical forces

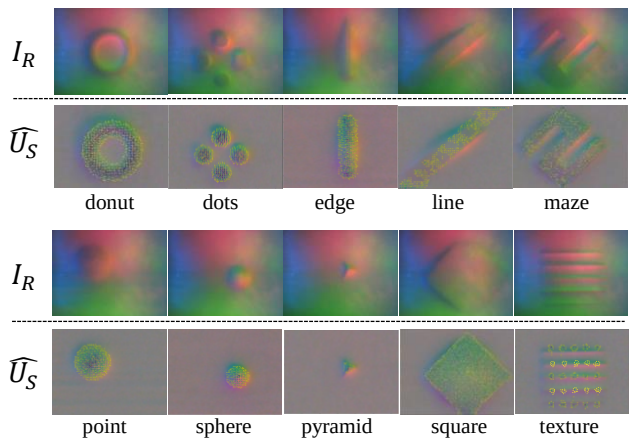


Figure 5.7: Visual examples of perception network. The network predicted deformation field $\hat{U}$ and corresponding force distribution for various indenters. These positions and rotations were not included data in the training.

from visual data. Although the network does not directly predict force, the predicted nodal displacements can be fed into the calibrated physical model (Eq. 5.8) to compute the corresponding contact forces. This hybrid approach, combining deep learning for perception with the underlying physical model, yielded a surface force RMSE of just 0.11 ± 0.03 N.

Qualitative results in Figure 5.7 further illustrate these capabilities, showing that the network accurately reconstructs detailed nodal displacement fields and force distributions for previously unseen contact events. The model not only captures the global contact geometry but also reproduces localized pressure patterns. With an inference speed exceeding 1,000 images per second on an RTX 5090 GPU, the perception model is well-suited for both rapid offline annotation of large datasets and deployment in real-time tactile perception pipelines.

5.8.3 Rendering Network Validation

The rendering network, designed to synthesize photorealistic tactile images from physical simulation data, was evaluated on the paired FEM-RGB validation set. Given multi-channel physical inputs from the FEM simulation (deformation fields and positional encodings), the network generated delta RGB images ($\Delta\hat{I}_R$) representing the contact imprint.

Quantitative evaluation using standard image similarity metrics confirmed the high fidelity of the generated images. The mean absolute error (L1) was 13.59 ± 1.03 on a 0–255 pixel intensity scale, indicating low overall pixel-wise error. The Structural Similarity Index Measure (SSIM) reached 0.971 ± 0.02, a value very close to unity that signifies an extremely high similarity in luminance, contrast, and structural features. Furthermore, the Peak Signal-to-Noise Ratio (PSNR) was 39.13 ± 1.81 dB, reflecting a low reconstruction error and high perceptual quality.

Representative examples in Figure 5.8 show that the rendered images are almost indistinguishable from their ground-truth counterparts, accurately reproducing both global contact shapes and fine, local texture details. These results demonstrate that the rendering network can generate high-fidelity synthetic tactile images from physically grounded FEM outputs, providing a powerful tool for realistic visual data augmentation. The network achieves an inference speed of approximately 220 images per second on the same hardware, supporting the fast, on-the-fly generation of training data.

5.8.4 Generalization to Unseen Data

The practical utility of the bidirectional networks was further assessed in scenarios that tested their generalization capabilities beyond the training distribution, as shown in Figure 5.9.

Perception network on external dataset. The perception model's robustness was tested on samples from the YCB object set [214] provided via the TACTO simulation framework [61]. This represented a significant domain

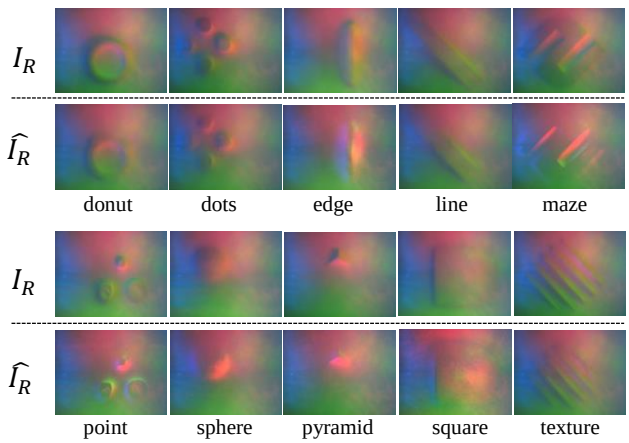


Figure 5.8: Visual examples of rendering results showing synthetic tactile images $\hat{I}_R$ compared with the real image I_R generated from deformation inputs $\hat{U}_S$ for various indenter geometries.

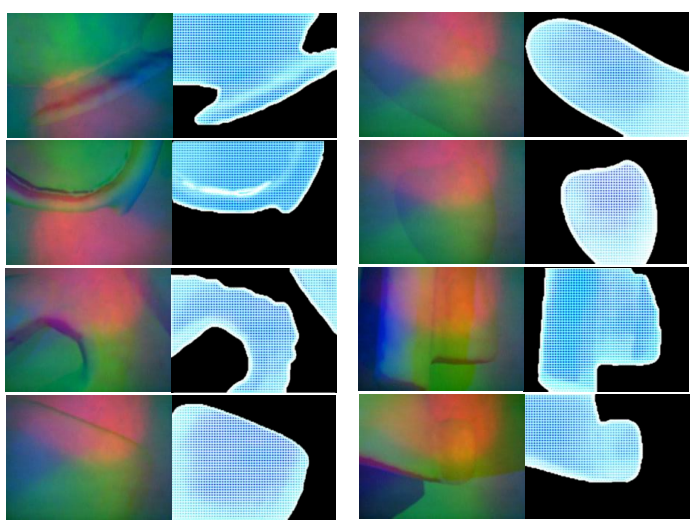


Figure 5.9: Perception network inference of surface deformation using RGB images collected from real-world objects not seen during training.

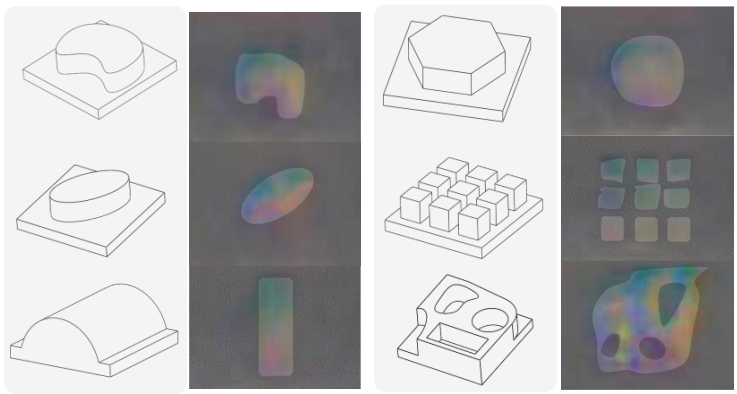


Figure 5.10: Rendering results from text-based indentation experiments, showing high-fidelity RGB outputs for various indenter shapes.

shift, as the lighting, shading, and object textures differed from the training data, and no background image was available for subtraction. As shown in Figure 5.9, despite these variations, the network successfully segmented the contact region and reconstructed a plausible height map of the nodal deformation, demonstrating that it learned the underlying physical relationship between visual features and 3D geometry rather than overfitting to the training conditions.

Rendering network on unseen indenters. The rendering model was tasked with generating images for complex indenter geometries not present in the training set. Nodal deformation fields for these novel shapes were produced purely through FEM simulation, without any corresponding real-world images. As presented in Figure 5.10, the network synthesized realistic delta RGB images for these new contact shapes, accurately capturing their global geometry. While minor artifacts were visible in some fine-scale details, the overall shapes were rendered consistently, confirming the network's potential for generating tactile imagery for entirely virtual objects and scenarios.

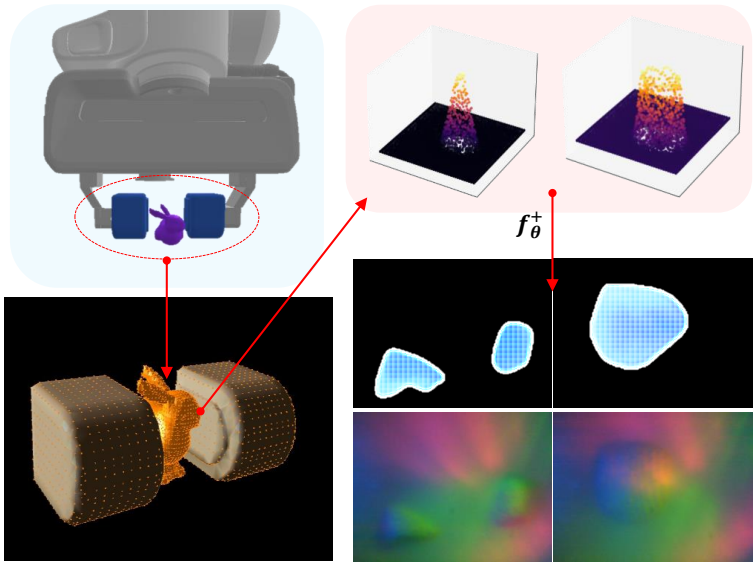


Figure 5.11: Rendering based on high-resolution contact data from a physics-based simulator. Deployment on a robot arm equipped with dual DIGIT sensors in environment. The rendering network synthesizes realistic RGB outputs during dynamic grasping

Rendering sensor in Physics-based simulation. As illustrated in Figure 5.11, the rendering network can be seamlessly integrated into standard physics-based robotic simulation pipelines. In such a setup, contact information from a simulated robot interaction is projected onto the sensor's nodal grid. This physical state is then passed through the rendering network to generate a realistic DIGIT sensor image in real time. This capability enables the scalable augmentation of training data for contact-rich manipulation tasks, such as in-hand control or fine assembly, by replacing simplistic contact signals with high-fidelity visual feedback. By preserving the physical grounding of contact, the generated images can be used to train perception and control policies that are more likely to succeed in the real world, thereby reducing the reliance on labor-intensive physical data collection.

5.9. Discussion

This work presented a bidirectional framework that creates a symbiotic link between high-fidelity finite element simulation and real-world ViTac sensor measurements. The core of the framework consists of two UNet-based networks: a perception model that regresses a dense, three-dimensional deformation field ($\hat{U}$) from a high-resolution RGB image, and a rendering model that synthesizes photorealistic tactile images from simulated physical states. By training these complementary models on a single, carefully paired dataset, a closed simulation-reality loop is established. This enables two powerful capabilities: the large-scale generation of physically grounded synthetic data for training, and the automated physical annotation of unlabeled real-world images, thereby overcoming the data bottleneck in tactile robotics.

Comparison to Prior Work. The contributions of this framework are best understood in the context of existing literature. In the domain of tactile perception, the objective of this work—dense deformation regression—is more fundamental than that of related methods. For instance, Sim2Surf [69] focuses on surface classification, while GenForce [201] targets the estimation of a single, three-axis force vector. The most similar objective is found in SimTacLS [206], which reconstructs skin shape; however, the present framework places a greater emphasis on a rigorously calibrated sim-to-real pipeline to ensure physical accuracy.

On the simulation front, many existing platforms like TACTO [61] and TacSL [65] prioritize scalability by using simplified contact models, such as rigid-body or penalty-based methods. While fast, these approaches do not explicitly resolve the complex, nonlinear deformation of the elastomer. In contrast, the FEM-based approach adopted here, similar in spirit to DiffTactile [66], is designed to capture the calibrated hyperelastic behavior derived directly from real sensor data, prioritizing physical fidelity.

Physically Grounded Learning. A key aspect of this work is the meticulous calibration of the FEM model using force-displacement measurements from a real sensor. This initial step significantly reduces the sim-to-real gap *before* any neural network training begins. By ensuring the paired training states are both visually and physically representative of reality, the networks are trained with physically grounded supervision. This allows the models to learn a more direct and accurate mapping between well-aligned domains, rather than simultaneously learning the mapping and compensating for a large, unknown domain shift.

Quantitative Performance. The perception network's ability to infer physical states from images was validated quantitatively. From the predicted deformation field $\hat{U}_S$, contact geometry and forces were derived and evaluated against the calibrated ground truth. The results demonstrate sub-millimeter precision in localizing contact, with a centroid error of 0.458 mm and a rotation error of 0.047 rad. Furthermore, the framework achieves high-fidelity force estimation, with a force RMSE of only 0.11 N. A comprehensive documentation of the mean errors, confidence intervals, coordinate frames, alignment procedures, and sample counts is provided in the project's Github repository. Checkpoints and evaluation scripts are also supplied to ensure exact reproducibility of these results.

The rendering network, which translates FEM states into RGB images, achieves excellent photorealistic quality, with an SSIM of 0.971 and a PSNR of 39.13 dB on a held-out test set. The network is also highly efficient, with an inference speed of 220 FPS on an RTX 5090 GPU (batch size 8), enabling on-the-fly data generation. Direct performance comparisons with prior work are challenging due to differences in sensors, datasets, and metrics. Therefore, claims of superiority are limited to shared evaluations under identical conditions. For methods like Taxim [64], where standard metrics were unavailable for the present dataset, controlled reimplementations were used for fair comparison.

Reproducibility and Future Directions. To facilitate further research and ensure reproducibility, all project materials are publicly available. This includes data acquisition scripts, FEM setup and calibration utilities, trained network checkpoints, the complete paired dataset, and evaluation code, all accessible at `https://github.com/ndolphin-github/DIGIT_simulation.git`.

The current framework, while robust, has several avenues for future work. The FEM model assumes quasistatic contact and a fixed hyperelastic material law; incorporating rate-dependent effects (viscoelasticity) would allow for the modeling of more dynamic interactions. Other important directions include developing methods for sensor transfer with minimal recalibration to adapt the models to new hardware, and exploring model compression techniques for deployment on resource-constrained edge devices.

Chapter 6.

A Geometry-Aware Neural Physics Engine for Dense Deformation Prediction

6.1. Motivation

The work in the preceding Chapter 5 established a data generation pipeline and the pretrained networks interpret and render. The methodology for generating high-fidelity, physically-grounded datasets that are paired with raw vision tactile RGB images was proposed. This addressed the data scarcity problems and the annotation bottleneck for the tasks where vision-based tactile sensors were utilized [61, 62, 67, 215]. This last research chapter represents a further expansion of that work, showcasing its direct application in creating a functional, data-driven simulation component. The goal is to present a generalizable method for integrating a soft, deformable gel-tip sensor into an accelerated physics engine with the extracted features from the high dimensional vision tactile image data.

This research is driven by the critical need for fast and physically realistic tactile simulation to bridge the sim-to-real gap for contact-rich robot learning. As has been established throughout this dissertation, a persistent trade-off exists between the fidelity of a simulation and its computational speed. This chapter aims to solve that trade-off for the specific, but critical, domain of tactile sensing. As a case study, this work continues to focus on the widely used DIGIT sensor, a representative example of a vision-based tactile sensor with a deformable interface. The ultimate goal of the work presented in this chapter is to answer a key question. How can we construct a simulated vision-based tactile sensor that can be rendered within a standard, fast physics-based simulation environment, while retaining the high physical fidelity of the underlying deformable body dynamics?

6.2. Introduction

The advancement of general-purpose robotics is increasingly tied to the development of large-scale, data-driven policies, often referred to as Robot Foundation Models [186, 216–218]. A primary bottleneck hindering the progress of these models is the lack of large-scale, multi-modal datasets, especially those containing the high-quality contact and force data that are essential for learning robust physical interaction. While the field has made significant strides using kinematic and visual data, a fundamental gap remains in providing learning algorithms with the rich, physically grounded information that governs the nuances of contact mechanics.

The vision-based tactile (ViTac) sensors have emerged as a promising hardware solution for capturing this intricate contact data [12, 60, 196]. By observing the deformation of a soft, compliant surface, these sensors provide high-resolution, image-like data that encodes a wealth of information about the contact event. However, the very success of these sensors highlights a subsequent challenge, which is the need for a scalable method to generate synthetic data for RL and other data-hungry learning paradigms. The direct collection of massive datasets from physical ViTac sensors remains a slow and resource-intensive process.

This necessitates a turn to simulation, which re-introduces the fundamental trade-off between computational speed and physical fidelity. On one hand, slow, high-fidelity FEM simulations can accurately model the deformable physics of the sensor tip but are computationally intractable for large-scale data generation. On the other hand, fast, rigid-body simulators are physically inadequate, as they cannot represent the crucial deformation mechanics that are central to the sensor's operation.

To bridge this critical gap, this chapter introduces the concept of a Neural Physics Engine (NPE) for tactile simulation. An NPE is a data-driven model that directly learns the complex physics of a specific phenomenon from high-fidelity data, serving as a computationally efficient and physically faithful replacement for a traditional physics solver. This research develops a geometry-

aware NPE for a ViTac sensor that is both fast enough for large-scale data generation and accurate enough for meaningful sim-to-real transfer.

The primary contribution of this work is the development and validation of this Neural Physics Engine, which is a novel Graph Neural Network (GNN) [219, 220] based proxy model trained on calibrated FEM data. The resulting NPE accurately predicts full-field deformations under contact, and its rendered output faithfully reproduces real-world sensor data from a dynamic trajectory. Its efficacy is demonstrated by showing its utility in enabling downstream perception tasks. This NPE provides a scalable solution for generating high-fidelity tactile data, thereby advancing the development of contact-aware robotic intelligence. Minor contributions that support this central work include the creation of a general trained FEM contact network that can estimate deformation without requiring a live simulation, a real-to-sim calibration methodology validated through a peg-in-hole task, and the final packaging of the NPE as a modularized simulated sensor that can be readily integrated into standard physics-based simulation environments.

6.3. Related Works

While a broad overview of the field was presented in Chapter 2, this section provides a more detailed and comparative analysis of the specific literature that contextualizes the contributions of this chapter. The following review examines the current state of the art in tactile sensor simulation and learned physics, highlighting the specific limitations of current contact and deformation simulations developed.

A prominent line of research in tactile simulation has focused on fast, rendering-based approaches. Variety of the simulators [61, 66, 221] have made significant strides by providing open-source tools that can generate high-resolution visual outputs for tactile sensors within standard physics simulations. These frameworks prioritize rendering speed and ease of integration, which have been crucial for their adoption in the robot learning community. However, their primary limitation is a lack of deep physical grounding. The

contact mechanics in these simulators are often based on simplified approximations [61,221] or penalty methods inherited from the underlying rigid-body engine [29–31], which do not explicitly resolve the nonlinear elastomer deformation that is central to the operation of the gel-sight-like sensors. This can lead to a physical grounding deficit, where the generated images are visually plausible but may not accurately reflect the true underlying force and deformation fields.

In direct contrast, another research thrust has focused on high-fidelity physics simulators. Some of the differentiable simulators [43,64,66] utilize the Finite Element Method (FEM) to accurately model the hyperelastic behavior of the sensor's soft components. This approach provides unparalleled physical accuracy, capturing the detailed stress-strain relationships that are missed by simpler models. The primary drawback of these methods, however, is their computational cost. The need to solve complex systems of equations at each time step makes them fundamentally unsuitable for the real-time, high-throughput data generation required for most modern reinforcement learning applications. They are faithful, but not fast.

The methodology presented in this chapter, namely learned physics and surrogate modeling, offers a hybrid approach that balances these competing demands. This work is distinct from research that uses geometry-aware GNN [219] to learn general physical laws from scratch. Instead of attempting to learn the physics of an entire scene, this research focuses on creating a highly specialized model for a single, critical subsystem: the soft-body contact of the tactile sensor. By training a geometry-aware GNN on high-fidelity, calibrated FEM nodal data, the resulting NPE serves as a fast and accurate proxy for a slow analytical solver.

Finally, this work could contribute to an alternative to the rigid-body contact solvers. Significant progress has been made in developing fast and stable online solvers for rigid-body contact, including iterative methods like Projected Gauss-Seidel [222], operator-splitting methods like ADMM [223], and Newton-based methods [224]. These solvers are essential for the efficient

simulation of articulated robotic systems. The NPE developed in this chapter does not seek to replace these general-purpose solvers. Instead, it addresses the complementary problem of soft-body deformation at the contact interface, a domain that rigid-body solvers are not designed to handle. The NPE is designed to be a modular component that can be integrated into these larger simulation frameworks to provide high-fidelity contact resolution where it is most needed.

In summary, the existing landscape of tactile simulation presents a clear trade-off between the speed of rendering-based approaches and the fidelity of physics-based ones. This chapter introduces an NPE that is designed to occupy the ideal space in this spectrum. It is physically grounded, having been trained on data from a meticulously calibrated FEM model. It is fast enough for large-scale data generation, running orders of magnitude faster than the high-fidelity simulation it replaces. It utilizes an efficient and generalizable structure, a geometry-aware GNN, to accurately predict the full-field deformation of the sensor. This provides a scalable and robust solution for generating the high-quality data required for advancing contact-aware robotic intelligence.

6.4. Framework Overview

The proposed framework enables the training of contact-rich manipulation policies within a fast simulation environment and facilitates zero-shot transfer to the real world. The core innovation lies in establishing a common physical representation—the full-field nodal deformation $\mathbf{U}$—which serves as a unified interface between the visual domain of the real world and the physics domain of the simulation. As illustrated in Figure 6.1, the overall pipeline consists of three interconnected modules described below.

6.4.1 Offline Perception and Rendering Learning

To ground the simulation in reality, a bi-directional mapping is first learned between raw tactile images and their underlying physical states. The Perception Network (f_θ), from the previous Chapter 5, acts as the sensory interface in the

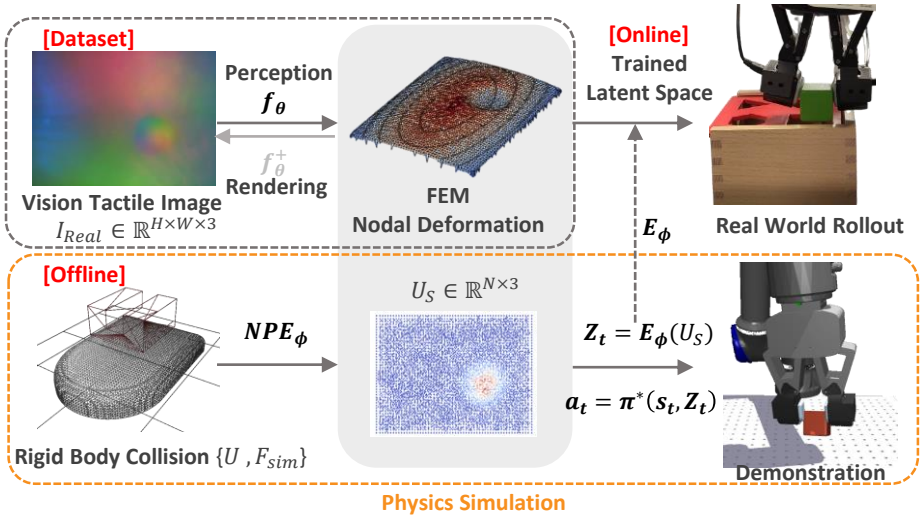


Figure 6.1: The proposed Sim-to-Real framework. (Left) The Perception Network f_θ and Rendering Network f_θ^+ are trained offline using real-world tactile images. (Right) The Neural Physics Engine (NPE_ϕ) predicts high-fidelity nodal deformations U_S from sparse simulation contacts. A shared Tactile Encoder E_ϕ maps these physical states into a latent vector Z_t, enabling the policy π^* to operate seamlessly across both domains.

real world, taking a raw vision-based tactile image $\mathbf{I}_{Real} \in \mathbb{R}^{H\times W\times 3}$ as input to estimate the dense nodal deformation field $\mathbf{U}_{Real} \in \mathbb{R}^{N\times 3}$ of the sensor surface:

$$\mathbf{U}_{Real} = f_\theta(\mathbf{I}_{Real}) \tag{6.1}$$

Complementing this, the Rendering Network (f_θ^+), also from the previous Chapter 5, serves as a validation and data augmentation tool by reconstructing the tactile image from a given physical deformation state $\mathbf{U}$. This reconstruction, denoted as $\hat{\mathbf{I}} = f_\theta^+(\mathbf{U})$, ensures the physical consistency of the representation. These networks are trained using a collected dataset of real-world interactions paired with ground-truth physics labels derived from high-fidelity FEM analysis.

6.4.2 Online Simulation with Neural Physics Engine

Standard rigid-body simulators, such as MuJoCo [29], provide only sparse contact points and forces ($\mathbf{C}, F_{sim}$), which are insufficient for accurately modeling soft-body interactions. To bridge this gap, the NPE is integrated directly into the simulation loop. At each simulation step t, the NPE_ϕ takes the sparse rigid-body collision data and predicts the high-fidelity nodal deformation $\mathbf{U}_S$:

$$\mathbf{U}_S = NPE_\phi(\mathbf{C}_t, F_{sim,t}) \tag{6.2}$$

This integration effectively upgrades the rigid-body simulator to possess soft-body characteristics in real-time (< 10 ms), circumventing the computational cost associated with full FEM solvers.

6.4.3 Unified Policy Learning and Deployment

The Sim-to-Real strategy relies on the shared tactile latent space. Instead of feeding raw images or raw mesh data directly to the policy, a shared Tactile Encoder (E_ϕ) was employed. This encoder maps the high-dimensional deformation field $\mathbf{U}$, whether generated by NPE_ϕ in the simulation or estimated by f_θ in reality, into a compact latent vector $\mathbf{Z}_t$:

$$\mathbf{Z}_t = E_\phi(\mathbf{U}) \tag{6.3}$$

The control policy π^* then observes the robot's proprioceptive state s_t alongside this unified tactile latent $\mathbf{Z}_t$ to output the optimal action $a_t = \pi^*(s_t, \mathbf{Z}_t)$. Since $\mathbf{Z}_t$ represents the same physical phenomenon in both domains, the policy trained in the simulation can be deployed directly to the real robot without fine-tuning.

6.4.4 Architecture of the Geometry-Aware Neural Physics Engine

The raw input from the simulation consists of a set of discrete contact points $C = \{\mathbf{p}_i\}_{i=1}^{N_t}$, where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the spatial coordinates of the i-th contact

point in the sensor's local frame. To process this variable-sized, unstructured input without information loss, a DeepSets architecture Ψ is employed. This ensures that the geometric embedding is invariant to the permutation of input points. The global geometric feature vector $\mathbf{g} \in \mathbb{R}^{d_g}$ is defined as:

$$\mathbf{g} = \rho\left(\sum_{i=1}^{N_t} \phi(\mathbf{p}_i)\right), \tag{6.4}$$

where $\phi : \mathbb{R}^3 \to \mathbb{R}^{d_h}$ is a point-wise Multi-Layer Perceptron (MLP) that projects each coordinate into a high-dimensional feature space, and $\rho : \mathbb{R}^{d_h} \to \mathbb{R}^{d_g}$ is a global MLP that processes the aggregated signal.

To account for the magnitude of interaction, the extracted geometric embedding $\mathbf{g}$ is fused with the scalar resultant force $F_{sim} \in \mathbb{R}$ obtained from the physics engine. The global physical context $\mathbf{h}_{global}$ is computed as:

$$\mathbf{h}_{global} = \text{MLP}_{fusion}([\mathbf{g} \parallel F_{sim}]), \tag{6.5}$$

where $\parallel$ denotes the concatenation operation.

The sensor surface is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to predict the dense nodal displacement field $\mathbf{U} \in \mathbb{R}^{M\times 3}$, where $M = 7509$ is the number of FEM nodes. The hidden state $\mathbf{h}_v^{(0)}$ of every node $v \in \mathcal{V}$ is initialized by concatenating the global physical context with the node's unique, fixed spatial coordinate $\mathbf{pos}_v \in \mathbb{R}^3$:

$$\mathbf{h}_v^{(0)} = [\mathbf{h}_{global} \parallel \mathbf{pos}_v] \tag{6.6}$$

Subsequently, a multi-layer Graph Convolutional Network (GCN) [225] is utilized to simulate the propagation of stress. For each layer $l \in \{0, \ldots, L-1\}$, the node features are updated as follows. First, neighbor features are aggregated using mean pooling:

$$\tilde{\mathbf{h}}_v^{(l)} = \mathbf{W}^{(l)} \cdot \frac{1}{|\mathcal{N}(v)|} \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l)}, \tag{6.7}$$

where $\mathcal{N}(v)$ is the set of neighbors for node v, and $\mathbf{W}^{(l)}$ is a learnable weight

matrix. The final node update applies Layer Normalization (LN), non-linearity, and a residual connection:

$$\mathbf{h}_v^{(l+1)} = \mathbf{h}_v^{(l)} + \text{ReLU}\left(\text{LN}\left(\tilde{\mathbf{h}}_v^{(l)}\right)\right) \tag{6.8}$$

After L layers, the Z-directional displacement d_v for each node is predicted by a shared readout MLP:

$$\hat{d}_v = \text{MLP}_{readout}(\mathbf{h}_v^{(L)}) \tag{6.9}$$

The final output is the dense displacement field $\hat{\mathbf{U}} = \{\hat{d}_v\}_{v=1}^{M}$.

6.5. Neural Physics Engine

The major methodology was designed to create a computationally efficient, yet physically grounded, surrogate model for soft tactile sensing within rigid-body simulators. The process involves three key stages: the generation of a high-fidelity ground truth dataset using a calibrated Finite Element Method simulation; the design of a novel, Geometry-Aware GNN that serves as our (NPE); and a principled training process that pairs fast simulation data with the slow FEM ground truth.

6.5.1 Physics-Grounded Truth Data Generation

The foundation of our NPE is a large-scale, physically-annotated dataset designed to capture a comprehensive range of contact phenomena. As illustrated in Figure 6.2, this ground truth data was generated using a calibrated FEM model of the tactile sensor. To ensure the learned model is robust and generalizable, we created a dataset with extensive geometric and spatial diversity. A set of diverse and complex indenters was used, ranging from simple primitives like spheres and squares to shapes with sharp edges, textures, and non-convex features. These indenters were used to create contacts at 492 unique contact points distributed across the sensor's surface, ensuring dense coverage of the

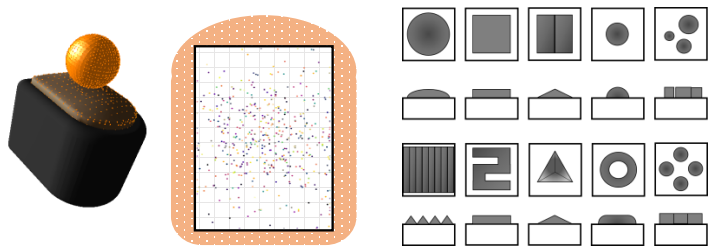


Figure 6.2: Over 50,000-sample dataset used to train the NPE. Data was generated using a calibrated FEM Model (left) by systematically applying a set of diverse and complex Indenters (right) at 492 unique contact points (center) distributed across the sensor's surface, ensuring comprehensive coverage for training a generalizable model.

entire region of interest. This systematic process resulted in a total of 50,000 unique contact samples, providing a rich and varied dataset for training our geometry-aware network. As shown in Figure 6.3, each sample consists of a full-field nodal displacement field for the 7,509 nodes on the sensor's surface mesh. For supervised learning tasks requiring contact regions, ground truth contact masks are derived directly from the FEM nodal data. A contact is defined where the normal displacement in the z-axis exceeds a threshold of 30 micrometers. The resulting binary mask undergoes a morphological processing pipeline, including morphological closing with a 15×15 kernel and contour-based refinement, to produce clean, continuous contact regions.

6.5.2 Architecture of the Geometry-Aware Neural Physics Engine

The raw input from the simulation consists of a set of discrete contact points $C = \{\mathbf{p}_i\}_{i=1}^{N_t}$, where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the spatial coordinates of the i-th contact point in the sensor's local frame. To process this variable-sized, unstructured input without information loss, a DeepSets architecture Ψ is employed [226]. This ensures that the geometric embedding is invariant to the permutation of input points. The global geometric feature vector $\mathbf{g} \in \mathbb{R}^{d_g}$ is defined as:

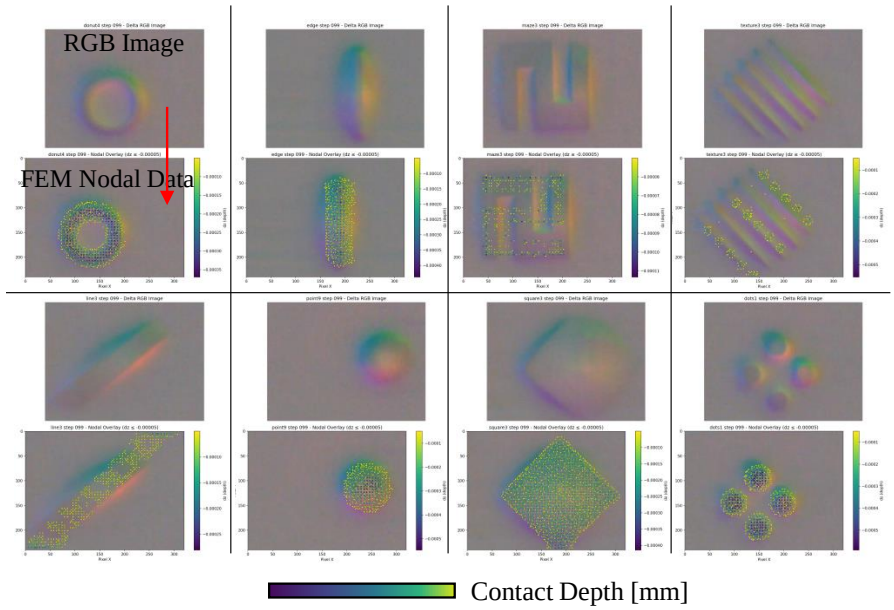


Figure 6.3: Representative samples from the 50,000-sample training dataset, showcasing a variety of paired contact geometries. Each pair consists of the rendered RGB Image (top) and the corresponding ground truth physical state, shown as the FEM Nodal Data (bottom).

$$\mathbf{g} = \rho\left(\sum_{i=1}^{N_t} \phi(\mathbf{p}_i)\right), \tag{6.10}$$

where $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^{d_h}$ is a point-wise Multi-Layer Perceptron (MLP) that projects each coordinate into a high-dimensional feature space, and $\rho : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{d_g}$ is a global MLP that processes the aggregated signal.

To account for the magnitude of interaction, the extracted geometric embedding $\mathbf{g}$ is fused with the scalar resultant force $F_{sim} \in \mathbb{R}$ obtained from the physics engine. The global physical context $\mathbf{h}_{global}$ is computed as:

$$\mathbf{h}_{global} = \text{MLP}_{fusion}([\mathbf{g} \parallel F_{sim}]), \tag{6.11}$$

where $\parallel$ denotes the concatenation operation.

The sensor surface is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to predict the dense nodal displacement field $\mathbf{U} \in \mathbb{R}^{M\times 3}$, where $M = 7509$ is the number of FEM nodes. The hidden state $\mathbf{h}_v^{(0)}$ of every node $v \in \mathcal{V}$ is initialized by concatenating the global physical context with the node's unique, fixed spatial coordinate $\mathbf{pos}_v \in \mathbb{R}^3$:

$$\mathbf{h}_v^{(0)} = [\mathbf{h}_{global} \parallel \mathbf{pos}_v] \tag{6.12}$$

Subsequently, a multi-layer Graph Convolutional Network (GCN) [225] is utilized to simulate the propagation of stress. For each layer $l \in \{0, \ldots, L-1\}$, the node features are updated as follows. First, neighbor features are aggregated using mean pooling:

$$\tilde{\mathbf{h}}_v^{(l)} = \mathbf{W}^{(l)} \cdot \frac{1}{|\mathcal{N}(v)|} \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l)}, \tag{6.13}$$

where $\mathcal{N}(v)$ is the set of neighbors for node v, and $\mathbf{W}^{(l)}$ is a learnable weight matrix. The final node update applies Layer Normalization (LN), non-linearity, and a residual connection:

$$\mathbf{h}_v^{(l+1)} = \mathbf{h}_v^{(l)} + \mathrm{ReLU}\left(\mathrm{LN}\left(\tilde{\mathbf{h}}_v^{(l)}\right)\right) \tag{6.14}$$

After L layers, the Z-directional displacement d_v for each node is predicted by a shared readout MLP:

$$\hat{d}_v = \mathrm{MLP}_{readout}(\mathbf{h}_v^{(L)}) \tag{6.15}$$

The final output is the dense displacement field $\hat{\mathbf{U}} = \{\hat{d}_v\}_{v=1}^{M}$.

6.5.3 Training Methodology

The Geometry-Aware Graph Neural Network (GA-GNN) is trained using a supervised learning framework to approximate high-fidelity FEM simulations. The training pipeline, illustrated in Figure 6.4, utilizes a paired dataset of rigid-body interactions and ground-truth FEM deformations to establish a mapping

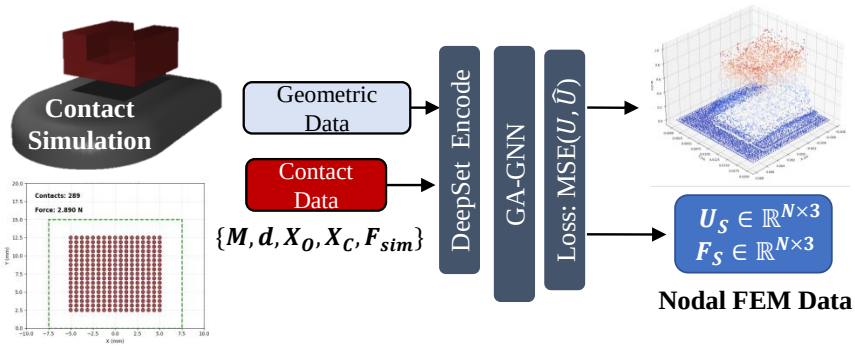


Figure 6.4: Training architecture of the Neural Physics Engine. The input vector $\{M, d, X_O, X_C, F_{sim}\}$ derived from the rigid-body simulation is encoded via DeepSets and processed by the GA-GNN. The network is optimized to minimize the Mean Squared Error (MSE) between the predicted nodal state ($\hat{U}$) and the ground-truth FEM data (U).

from sparse contact primitives to dense continuum mechanics.

Data Generation and Input Representation To ensure robust generalization, a training dataset is generated by simulating a diverse set of contact interactions using primitive indenter geometries (e.g., spheres, cylinders, flat plates) within both the rigid-body simulator (MuJoCo) and the FEM solver. For each interaction frame t, the input vector $\mathcal{I}_t$ is extracted from the rigid-body simulation:

$$\mathcal{I}_t = \{M, d, X_O, X_C, F_{sim}\} \tag{6.16}$$

where:

- M: The topological connectivity (adjacency matrix) of the sensor mesh.
- d: The indentation depth of the contacting object.
- X_O: The local point cloud representing the object's geometry.
- X_C: The sparse set of detected contact points on the sensor surface.

- F_{sim}: The resultant contact force vector computed by the physics engine.

Ground Truth Physics Targets The target output $\mathcal{T}_t$ corresponds to the full-field nodal data obtained from the high-fidelity FEM simulation. This includes the dense displacement field $\mathbf{U} \in \mathbb{R}^{N\times 3}$ and the nodal force distribution $\mathbf{F} \in \mathbb{R}^{N\times 3}$ for all $N = 7509$ nodes on the sensor surface. These values serve as the ground truth for supervision.

Network Architecture and Loss Function The sparse geometric inputs (X_O, X_C) are initially processed by the DeepSets Encoder to extract a permutation-invariant geometric embedding. This embedding is fused with the scalar physics parameters (d, F_{sim}) to form a global context vector, which is then injected into the GA-GNN.

The GA-GNN predicts the estimated nodal state $\hat{\mathbf{S}} = \{\hat{\mathbf{U}}_S, \hat{\mathbf{F}}_S\}$. The network parameters are optimized end-to-end by minimizing the Mean Squared Error (MSE) between the prediction and the FEM ground truth. The total loss function $\mathcal{L}_{total}$ is defined as:

$$\mathcal{L}_{total} = \text{MSE}(\mathbf{U}, \hat{\mathbf{U}}_S) + \lambda \cdot \text{MSE}(\mathbf{F}, \hat{\mathbf{F}}_S), \tag{6.17}$$

where $\mathbf{U}$ represents the ground-truth nodal displacement, $\hat{\mathbf{U}}_S$ is the predicted displacement by the NPE, and λ is a weighting coefficient for the force reconstruction term. This objective forces the network to learn the underlying continuum mechanics governing the sensor's deformation, ensuring that the predicted high-dimensional state maintains physical plausibility.

6.6. Simulation Experiment

6.6.1 Generative Surrogate Model for FEM Data Synthesis

To facilitate the creation of a large and diverse training dataset for the NPE without the prohibitive computational cost of continuous FEM simulation, a generative model for FEM was first developed. This model, termed the "FEM

point network," was trained on the initial 50,000-sample FEM dataset, which is the extended dataset from the one in Chapter 5. It learns to map a target contact state, defined by an indenter's local geometry point cloud and a desired indentation depth, directly to the corresponding high-fidelity physical outputs, such as the full-field nodal displacement, surface contact force, contact patch, and location.

Figure 6.5 presents a qualitative validation of this FEM generation model's performance. The model takes a target contact state (indenter geometry point cloud and depth) as input and outputs the complete nodal displacement field, bypassing the need for a full FEM simulation. The visual results demonstrate the model's ability to accurately reproduce a wide range of complex contact patterns. The figure displays a predicted nodal displacement fields for 16 different indenter geometries at their maximum indentation depth of 1 mm. The color map represents the normal displacement (d_z), clearly showing the unique contact signature of each shape. The high fidelity of these predictions validates the surrogate model's capability to function as a data factory. This allows for the rapid, offline synthesis of vast amounts of physically grounded training data, a crucial step for effectively training the final, real-time NPE.

6.6.2 Evaluation of Shared Latent Space via Behavior Cloning in Peg-in-Hole Task

The efficacy of the proposed shared latent representation is evaluated through a high-precision Peg-in-Hole task simulated within the MuJoCo environment. The experimental setup, illustrated in Fig. 6.6, consists of a UR5e manipulator equipped with a parallel gripper and dual DIGIT tactile sensors. To validate the utility of the NPE in downstream learning tasks, a Behavior Cloning (BC) framework is implemented.

During the expert demonstration phase, the robot is teleoperated to insert a peg into a hexagonal aperture. At each time step, the NPE processes the sparse contact primitives to generate high-fidelity nodal displacement fields for both the left and right sensor pads ($N = 2552$ nodes per sensor). As shown in the

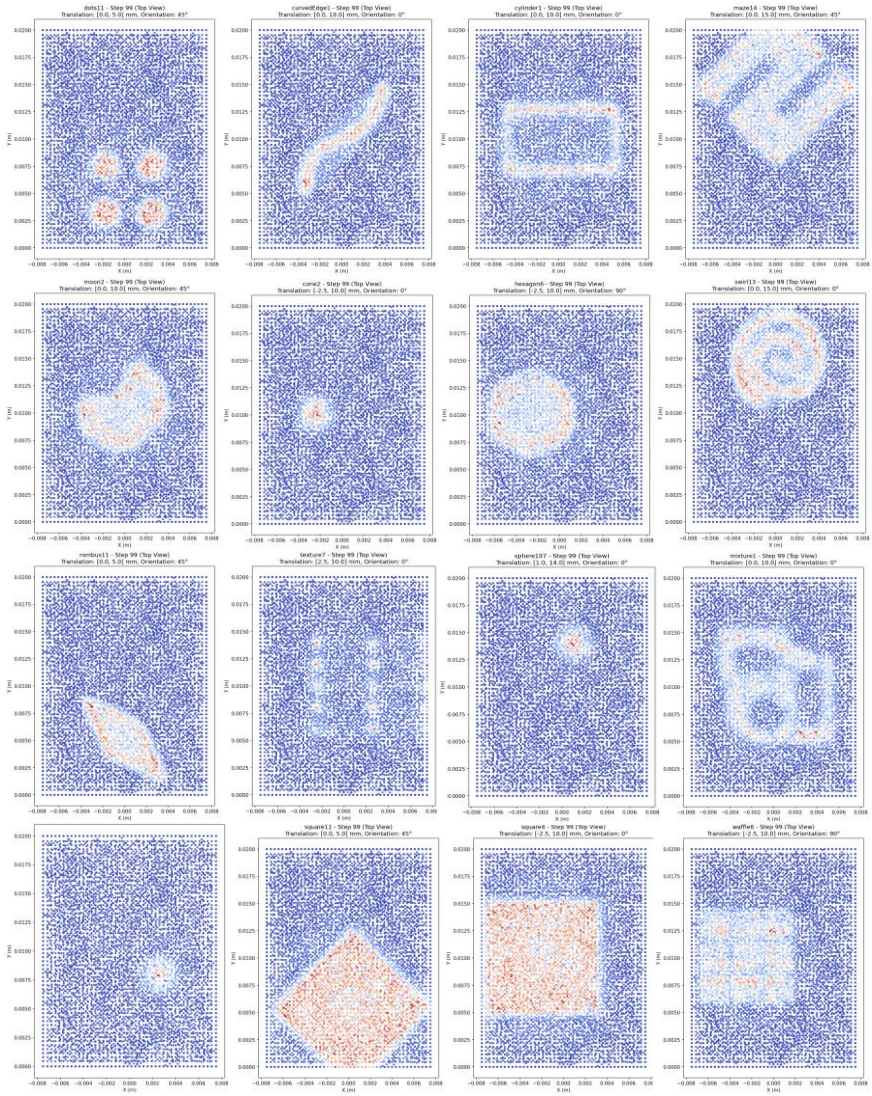


Figure 6.5: Qualitative results of the FEM surrogate model. A full-field nodal displacement predictions is generated by the model for a variety of complex indenter geometries at maximum indentation depth.

middle row of Fig. 6.6, the NPE captures distinct, asymmetric deformation patterns corresponding to the contact geometry and force distribution on each finger.

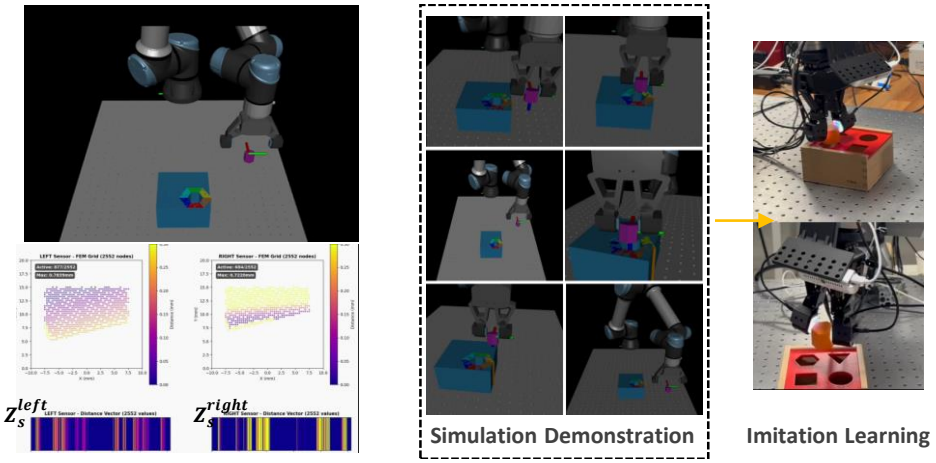


Figure 6.6: Behavior Cloning with shared tactile latent space. (Top) The simulation setup for the Peg-in-Hole task. (Middle) High-fidelity nodal displacement fields generated by the NPE for the left and right sensors ($N = 2552$). (Bottom) The corresponding latent vectors $\mathbf{Z}_S^{left}$ and $\mathbf{Z}_S^{right}$ extracted by the encoder, which serve as the input to the policy network.

These high-dimensional physical states $\mathbf{U}_S$ are subsequently compressed by the pre-trained Tactile Encoder E_ϕ into compact latent vectors, denoted as $\mathbf{Z}_S^{left}$ and $\mathbf{Z}_S^{right}$ (bottom row of Fig. 6.6). A control policy π is trained to map the combined state vector—comprising the robot's proprioception and the concatenated latent tactile features $[\mathbf{Z}_S^{left}, \mathbf{Z}_S^{right}]$—to the expert's end-effector velocity actions. The successful convergence of the BC policy demonstrates that the latent space $\mathbf{Z}$, derived from the NPE-predicted deformation $\mathbf{U}$, effectively preserves the critical geometric and contact information required for contact-rich manipulation tasks.

To validate the proposed Sim-to-Real framework, we conducted a series of physical experiments focusing on the high-precision Peg-in-Hole task. The primary objective was to assess the efficacy of the shared tactile latent space in enabling zero-shot transfer of policies trained via Behavior Cloning (BC) from simulation to the real world.

The training dataset was collected entirely within the MuJoCo simulation environment using the teleoperation interface described in Section **??**. A total

of 137 expert demonstrations were recorded. To ensure stable grasping and prevent the peg from slipping during rapid movements, the robot was controlled using Inverse Kinematics (IK) velocity commands rather than position control.

The trained policy was then deployed on a real-world UR5e manipulator equipped with dual DIGIT sensors. For the real-world rollout, we performed 100 trials for each experimental condition. It is important to note that no fine-tuning of the neural network weights was performed using real-world data. The only adaptation allowed was the calibration of the robot's joint angles to compensate for minor kinematic discrepancies between the simulated and physical robot.

6.6.3 Policy Learning via Behavior Cloning with Latent Features

To enable the control policy to operate effectively across domains, a dataset of expert demonstrations is generated within the simulation environment using teleoperation. This dataset, denoted as $\mathbb{D}_{sim}$, consists of M trajectories, where each trajectory i contains a sequence of length T_i comprising the robot's proprioceptive state $\mathbf{s}_t$, the encoded tactile latent vector $\mathbf{z}_t$, and the corresponding expert action $\mathbf{a}_t^*$. The dataset is formalized as:

$$\mathbb{D}_{sim} = \left[\{\mathbf{s}_t, \mathbf{z}_t, \mathbf{a}_t^*\}_{t=1}^{T_i} \right]_{i=1}^{M} \tag{6.18}$$

where $\mathbf{z}_t = E_\phi(\mathbf{U}_t^{sim})$ is the latent representation derived from the NPE-predicted physical state.

The control policy π^*, parameterized by ψ, is trained to map the combined state $(\mathbf{s}, \mathbf{z})$ to the optimal action. The training objective is to minimize the discrepancy between the policy's predicted action and the expert's ground-truth action. The Behavior Cloning loss function L_{BC}, which jointly optimizes the policy parameters ψ and the tactile encoder parameters ϕ, is defined as:

$$L_{BC}(\phi, \psi) = \mathbb{E}_{\mathcal{D} \sim \mathbb{D}_{sim}} \left[\|\pi^*(\mathbf{s}, \mathbf{z}) - \mathbf{a}^*\|_2^2 \right] \tag{6.19}$$

By minimizing this objective, the encoder E_ϕ learns to extract task-relevant

physical features (such as contact location and force magnitude) from the dense deformation field, while the policy π^* learns the manipulation strategy conditioned on these physical abstractions.

6.6.4 Sim-to-Real Validation via Trajectory Replay

The validation of the proposed framework was conducted through a digital twin experiment, designed to directly compare the sensory output of the NPE-enhanced simulation against data from a real-world, dynamic task. The goal is to demonstrate that the simulation can faithfully reproduce the complex tactile phenomena observed during a physical interaction.

For this experiment, a successful peg-in-hole manipulation task was first performed and recorded using a real UR5e robot equipped with a parallel gripper and two DIGIT sensors, as shown in Figure 6.7 (left). The complete robot joint trajectory and the corresponding high-resolution video stream from the real DIGIT sensors were recorded. Subsequently, the exact joint trajectory from the successful episode was replayed in the NPE-enhanced MuJoCo environment. During this simulated replay, each time a sensor's rigid proxy made contact with the virtual object, the NPE was invoked in real-time. As illustrated in the right panel of Figure 6.7, the NPE takes the sparse contact data from the simulator and generates a rich, physically-annotated data stream, including the full-field nodal displacement, the contact patch shape, and the final rendered tactile image. A qualitative comparison between the synthetic tactile image stream from the simulation and the video from the real DIGIT sensor reveals a high degree of visual and structural similarity, validating that the NPE-enhanced simulation can faithfully reproduce the sensory experience of a complex, real-world contact-rich task.

6.7. Result

The performance of the NPE was evaluated in two stages. First, a quantitative and qualitative validation of the NPE's core prediction capabilities was conducted against the ground truth dataset. Second, the entire framework

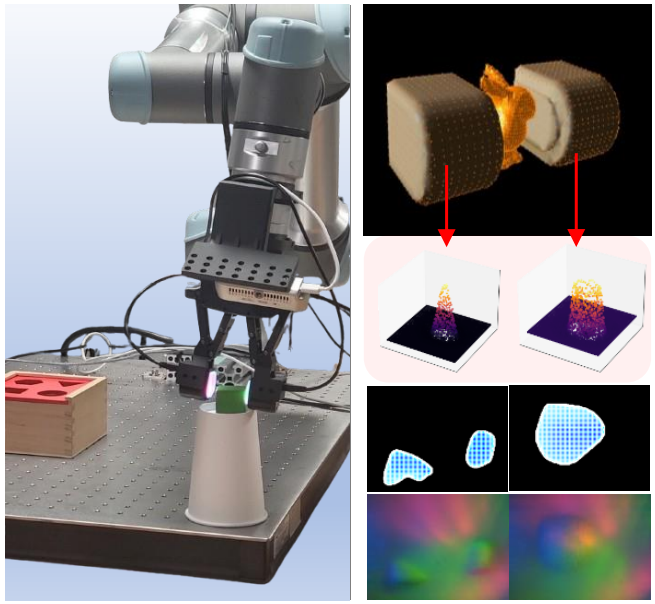


Figure 6.7: Sim-to-Real validation via trajectory replay in a peg-in-hole task. The real-world experimental setup (left), where a robot arm equipped with dual DIGIT sensors performs the contact-rich manipulation task. During the simulated replay of this task (right), the NPE is activated upon contact. From the sparse simulator data, it generates a stream of rich, physically-annotated data, including the predicted nodal displacement (top), contact patch shape (middle), and the final rendered synthetic tactile image (bottom)

was validated in a dynamic, contact-rich manipulation task via a real-to-sim trajectory replay experiment.

6.7.1 Quantitative and Qualitative Validation of the NPE

The NPE was evaluated for its accuracy in predicting physical contact properties and its utility in downstream perception tasks. The quantitative results, summarized in Table 6.1, demonstrate the high fidelity of the model. The model achieves 99.34% accuracy in identifying the correct contact nodes and predicts the full-field nodal displacement with a low RMSE of 0.53 mm compared to the ground truth FEM data.

The modularized NPE accurately identifies the location of the contact

Metric	Value
Contact Point Accuracy	99.34 %
Nodal Displacement RMSE	0.53 ± 0.07 mm
Orientation Estimation Error	0.3 ± 0.02 deg
Force Estimation RMSE	0.32 ± 0.03 N

Table 6.1: Quantitative Performance of the NPE

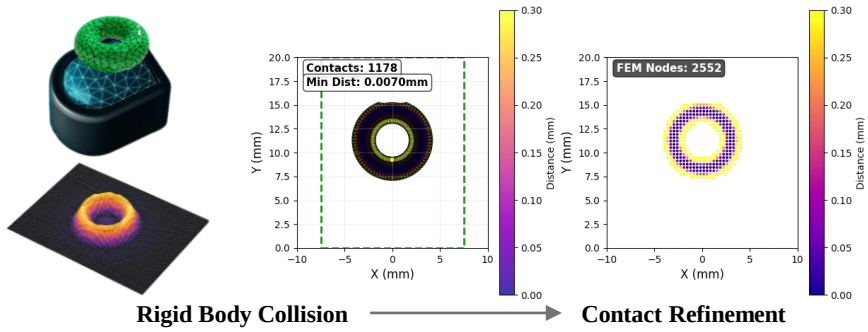


Figure 6.8: Functional demonstration results of the NPE in physics-based simulation. For a simple indentation, the model accurately localizes the contact patch and the corresponding contact force profile.

and the corresponding contact force profile, as shown in Figure 6.8 for contact localization. The complex contact patch generated by an asymmetrically grasped object contains sufficient detail to accurately estimate the object's orientation, achieving a mean error of only 0.3 degrees (Table 6.1). Furthermore, the predicted nodal deformation allows for the estimation of the total contact force with an RMSE of 0.32 N compared to the FEM ground truth. These results confirm that the NPE produces an accurate and physically rich output suitable for advanced perception tasks.

6.7.2 Sim-to-Real Transfer Performance

We compared the success rate of the policy using our proposed **Shared Latent Space** (where the policy input is $\mathbf{Z} = E_\phi(\mathbf{U})$) against a **Baseline** approach that

does not utilize the physically grounded latent alignment (i.e., mapping raw sensory features directly to actions without the intermediate nodal displacement field **U**).

The results are summarized in Table 6.2. The proposed method achieved a success rate of 89%, demonstrating robust zero-shot transfer. In contrast, the baseline method achieved only a 43% success rate. This significant performance gap highlights that without the domain-invariant physical representation **U**, the policy struggles to generalize to the real-world tactile domain, likely due to the "reality gap" in raw optical signals (e.g., lighting variations, sensor noise).

Table 6.2: Sim-to-Real Peg-in-Hole Success Rates (100 Real-World Trials)

Method	**Tactile Representation**	**Zero-Shot?**	**Success Rate**
Baseline	Direct Feature Mapping	Yes	43%
Proposed (Ours)	**Shared Latent Space (Z)**	**Yes**	**89%**

The failure cases in the baseline were primarily characterized by the robot's inability to correct for minor misalignments during the contact-rich search phase, suggesting a lack of interpretable physical feedback. The proposed method, however, successfully exhibited "wiggling" and alignment behaviors learned from the simulation experts.

6.7.3 Comparison with Related Works

To contextualize our results, we compare our success rate with recent state-of-the-art methods in tactile Sim-to-Real transfer for insertion tasks (Table 6.3).

Our method's performance (89%) is competitive with top-tier reinforcement learning approaches like GCS (91%) [66] and outperforms standard domain randomization techniques often cited in works like Tactile Gym [62]. Unlike *SplatSim* [68], which relies on heavy visual rendering fidelity, or *Chen et al.* [201], which relies on marker tracking, our approach achieves high fidelity through the abstraction of contact physics (NPE), proving that a shared

Table 6.3: Comparison with State-of-the-Art Tactile Sim-to-Real Methods

Reference	Method	Sim-to-Real Strategy	Success Rate
Tactile Gym [62]	RL (PPO)	Domain Randomization	~60-80%
Chen et al. [201]	RL + TacSL	Marker-based Feature Alignment	83%
GCS [66]	RL (PPO)	Geometric Contact Smoothing	91%
SplatSim [68]	BC (Visual)	Gaussian Splatting Rendering	86%
Ours	**BC + NPE**	**Shared Physical Latent (U)**	**89%**

latent space grounded in continuum mechanics is a highly effective bridge for the reality gap.

6.7.4 Application: Sim-to-Real Trajectory Replay

To validate the entire framework in a dynamic scenario, a digital twin experiment was conducted. As illustrated in Figure 6.9, a successful real-world peg-in-hole task, involving phases such as ‘Peg Contact‘ and ‘Hold‘, was recorded using a teleoperated robot. The recorded joint trajectory (‘Action Replay‘) was then executed in the NPE-enhanced MuJoCo simulation.

During the simulated replay, the NPE was activated at each contact event, generating a stream of physically grounded tactile data. A qualitative comparison between the synthetic data generated by the NPE in simulation and the sensory data from the real DIGIT sensor reveals a high degree of correlation, demonstrating the NPE's ability to faithfully reproduce the key events of the real-world interaction. This experiment validates the potential of the NPE-enhanced simulation to serve as a high-fidelity environment for developing and testing contact-rich manipulation policies, thereby narrowing the sim-to-real gap.

6.8. Discussion

The experimental results demonstrate that the proposed NPE can serve as a fast and accurate surrogate for high-fidelity FEM simulations of soft tactile sensors. The quantitative validation confirms that the model predicts full-

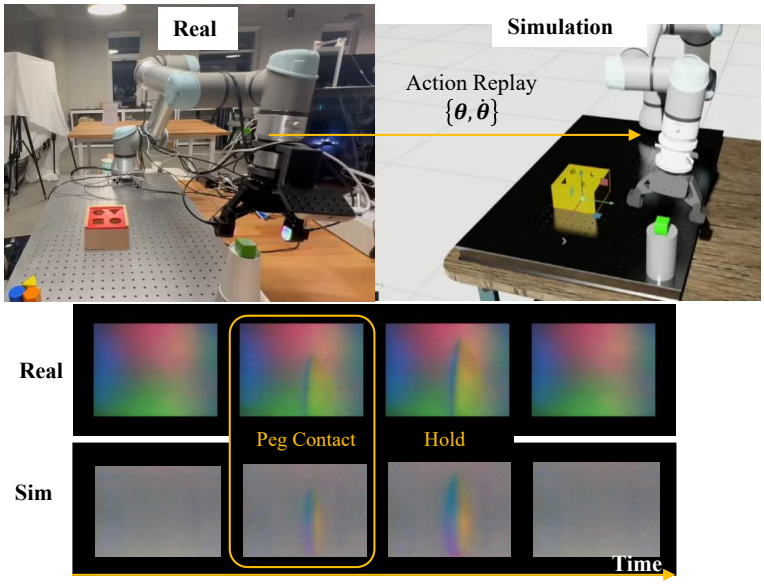


Figure 6.9: The sim-to-real trajectory replay experiment. A successful real-world manipulation sequence, involving contact and holding phases, is recorded. The robot's actions are then replayed in the simulation, where the NPE generates a corresponding stream of high-fidelity tactile data, demonstrating the framework's ability to create a digital twin of the real-world interaction.

field nodal displacements with sub-millimeter accuracy when compared to the ground truth. Crucially, the ablation studies reveal that the geometry-aware architecture, which processes the local 3D point cloud of the contacting object, is essential for achieving this high performance. Without this geometric input, even a spatially-aware GNN model fails to capture the correct shape of the deformation, confirming that both force and geometry are critical input modalities.

The success of the digital twin trajectory replay experiment is the primary validation of the entire framework. The high degree of similarity between the synthetic tactile data stream and the real sensor's output suggests that the NPE has learned a model of contact mechanics that is not only accurate with respect to the FEM data it was trained on, but is also a faithful representation of real-world physics. This result is enabled by the Rigid Proxy architecture, which

allows a standard rigid-body simulator to be augmented with a specialized, learned model for a specific physical phenomenon—in this case, soft-body contact. Instead of attempting to solve the complex contact equations with an online iterative solver, our approach leverages a significant offline training cost to achieve extremely fast and consistent online inference.

Despite these promising results, the current framework has several limitations. First, the NPE is trained on quasi-static FEM data and therefore does not explicitly model dynamic, rate-dependent effects such as viscoelasticity or high-frequency vibrations. Second, the model's accuracy is fundamentally bound by the fidelity of the original FEM simulation; any inaccuracies in the FEM calibration will be inherited by the NPE. Finally, the current model is specific to the geometry and material properties of the trained sensor. Future work should address these limitations by extending the training dataset to include dynamic events, which would enable the development of a recurrent, spatiotemporal NPE. Further research could also explore transfer learning techniques to adapt the NPE to different sensor geometries with minimal retraining, and investigate the use of the NPE's rich output to provide dense rewards for reinforcement learning agents.

Chapter 7.
Conclusion of Dissertation

7.1. Integrated Methodological Framework

This dissertation was motivated by a fundamental barrier in modern robotics: the simulation gap for deformable systems. While rigid-body physics engines have enabled the rapid progress of reinforcement learning for articulated robots, they fundamentally lack the capacity to represent the continuous compliance and complex contact mechanics essential for physical intelligence. Conversely, high-fidelity numerical methods such as Finite Element Analysis provide the necessary physical ground truth but are computationally intractable for the massive data requirements of learning algorithms. This work has addressed this dichotomy by proposing and validating a unified methodological framework for Data-Driven Physics Abstraction.

The central thesis of this research is that the complex, infinite-dimensional behavior of continuum systems can be effectively compressed into low dimensional, differentiable inference models without sacrificing essential physical fidelity. By utilizing high-fidelity FEM simulations not as runtime solvers but as offline teachers, this dissertation has developed a set of data generation engines—fast, physically grounded surrogates that replace iterative numerical solving with constant-time neural inference.

To rigorously apply this framework, the research strategically bifurcated the problem into two topologically distinct domains. For the Action domain (Part A), the methodology abstracted volumetric deformation into a kinematic surrogate model, preserving compatibility with rigid-body solvers to enable macroscopic control. For the Perception domain (Part B), the methodology abstracted boundary deformation into a neural physics engine using graph neural networks, capturing the topological propagation of force to enable microscopic

tactile sensing. Collectively, these contributions provide the essential infrastructure to bridge the reality gap, enabling the training of contact-aware and physically intelligent robots that can move, manipulate, and perceive with the richness of the real world.

7.2. Summary of Contributions

This dissertation systematically validated the proposed framework through a logical progression of four research topics, each addressing a specific facet of the simulation gap. The investigation began by establishing a theoretical baseline using classical methods. Chapter 3 derived an analytical kinematic model for a pneumatically actuated origami manipulator based on geometric first principles. This study demonstrated that while geometric abstraction can enable real-time nonlinear control for specific structures, the analytical derivation lacks the generality required for arbitrary soft bodies. The reliance on bespoke mathematical formulations confirmed that classical methods are insufficient for scaling to the diverse morphologies of soft robotics, thereby motivating the shift to data-driven approaches.

To address these limitations, Chapter 4 introduced a generalizable surrogate modeling framework for the macroscopic dynamics of soft manipulators. By combining Model Order Reduction on FEM data with a Transformer-based dynamics model, the continuum physics were abstracted into a virtual kinematic chain. This approach successfully bridged the gap between accurate offline physics and fast online inference, creating a high-fidelity simulation environment compatible with standard reinforcement learning libraries. The validation of this framework through zero-shot sim-to-real policy transfer demonstrated that abstracting continuum mechanics into a kinematic structure allows soft robots to leverage the stability and speed of rigid-body solvers while retaining their essential physical compliance.

Shifting focus to the contact interface, Chapter 5 addressed the data scarcity problem for vision-based tactile sensors. A rigorously calibrated FEM simulation was utilized to generate a paired dataset of real images and physical

states, enabling the training of bidirectional networks. This work produced an automatic annotation engine capable of inferring dense force and deformation fields from raw images, as well as a rendering engine for synthesizing photorealistic tactile data. This established a physically grounded link between the visual and physical domains, solving the critical labeling bottleneck that hinders data-driven tactile perception.

Finally, Chapter 6 integrated these methods to create a runtime contact simulator. A Geometry-Aware Graph Neural Network was trained to predict nodal deformation based on local contact geometry, effectively replacing the iterative FEM solver. The resulting Neural Physics Engine was integrated into a rigid-body simulator via a rigid proxy architecture, achieving real-time performance with sub-millimeter accuracy on unseen objects. By enforcing a topological inductive bias, the model demonstrated zero-shot generalization to novel geometries, validating that learned physics engines can provide the high-fidelity contact mechanics required for contact-rich manipulation tasks.

7.3. Limitations

Despite the successful validation of the proposed framework, several limitations inherent to the methodologies and the scope of validation remain.

In the domain of body dynamics (Part A), both the geometric analysis and the surrogate modeling framework encounter difficulties under conditions of extreme deformation. While the surrogate model captures the primary modes of motion, the approximation of a continuum body as a kinematic chain breaks down when the material undergoes severe twisting or buckling that exceeds the training distribution. Furthermore, the applicability of these pipelines is sensitive to the robot's specific actuation method and structural morphology. Although the abstraction methodology is theoretically applicable to various forms of deformable bodies, this dissertation primarily validated it on pneumatic origami structures. The generalization of this framework to other soft actuation types, such as tendon-driven or dielectric elastomer actuators, requires further demonstration to confirm its universality.

In the domain of contact perception (Part B), a fundamental challenge remains in identifying the true latent information required for general contact tasks. While Chapter 5 demonstrated that force and deformation fields can be inferred from images, the question of exactly which physical features are essential for general-purpose manipulation remains open. This work validated the utility of the extracted latent information in the specific context of a peg-in-hole task in Chapter 6, but this represents only a single piece of validation. A broader range of tasks is necessary to determine if the current set of abstracted features is sufficient for diverse manipulation scenarios.

Furthermore, a significant scope limitation of this research is the focus on the robot's own deformation, leaving the simulation of deformable objects largely unaddressed. Real-world interaction frequently involves rigid robots manipulating soft objects or soft robots interacting with soft environments. While the Neural Physics Engine theoretically models the relative action-reaction at the interface—implying that the deformation of an object could be modeled similarly to the sensor—this extension was not explicitly validated. Validating such interactions requires ground truth data for the object's deformation, which is exceptionally challenging to acquire. Unlike the sensor, which can be instrumented, obtaining precise force and shape data from a passive deforming object would require invasive instrumentation (such as internal load cells) or complex external tracking systems. Consequently, the validation of this framework for deformable object manipulation remains a critical gap to be addressed.

7.4. Future Directions and Long-Term Vision

The methodologies established in this dissertation lay the groundwork for several transformative research directions in the field of robot learning.

Self-Supervised Physics Learning. A significant bottleneck remains the reliance on FEM for ground truth generation. Future work could leverage the bidirectional pipeline developed in Chapter 5 to implement a self-supervised learning paradigm. By utilizing the rendering network as a differentiable de-

coder, the perception model could be trained directly on vast amounts of unlabeled real-world tactile videos using a photometric reconstruction loss. This would allow the physics abstraction to refine itself against real-world data without explicit FEM supervision, effectively closing the sim-to-real loop autonomously.

Force-Aware Foundation Models. Current robot foundation models primarily process visual and linguistic tokens, lacking a deep understanding of physical interaction. The data generation engines developed in this work provide the capability to generate massive, physically annotated datasets of contact and deformation. Future research should explore tokenization strategies for this high-dimensional physical data. By integrating these physics tokens into large-scale transformer training, it becomes possible to imbue foundation models with a true physical intuition, enabling them to reason about force, compliance, and material properties alongside vision and language.

Unified Differentiable Simulation. While this work utilized a rigid proxy to integrate with standard simulators, the ultimate goal is a fully differentiable, end-to-end simulation pipeline. Future work should focus on integrating the differentiable inference models directly into the computation graph of differentiable physics engines. This would enable end-to-end gradient optimization for hardware design, control policy, and perception algorithms simultaneously, realizing the vision of co-designing the robot's body and brain for physical intelligence.

Bibliography

[1] R. McCarthy, D. C. Tan, D. Schmidt, F. Acero, N. Herr, Y. Du, T. G. Thuruthel, and Z. Li, "Towards generalist robot learning from internet video: A survey," *Journal of Artificial Intelligence Research*, vol. 83, 2025.

[2] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, *et al.*, "Vision-language foundation models as effective robot imitators," *arXiv preprint arXiv:2311.01378*, 2023.

[3] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 701–739, 2025.

[4] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference*, pp. 304–313, IEEE, 1984.

[5] A. Bicchi, "Hands for dexterous manipulation and powerful grasping: A difficult road towards simplicity," in *International Symposium of Robotics Research*, vol. 7, pp. 2–15, Springer, 1996.

[6] E. F. Camacho and C. Bordons, "Constrained model predictive control," in *Model predictive control*, pp. 177–216, Springer, 2007.

[7] D. A. Haggerty, M. J. Banks, E. Kamenar, A. B. Cao, P. C. Curtis, I. Mezić, and E. W. Hawkes, "Control of soft robots with inertial dynamics," *Science robotics*, vol. 8, no. 81, p. eadd6864, 2023.

[8] T. Kim, S. Lee, S. Chang, S. Hwang, and Y.-L. Park, "Environmental adaptability of legged robots with cutaneous inflation and sensation," *Advanced Intelligent Systems*, vol. 5, no. 1, p. 2370050, 2023.

[9] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 121–127, IEEE, 2015.

[10] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.

[11] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands," *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[12] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[13] D. Marcheseandrew, D. Onalcagdas, *et al.*, "Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators," *Soft robotics*, 2014.

[14] F. Corucci, N. Cheney, S. Kriegman, J. Bongard, and C. Laschi, "Evolutionary developmental soft robotics as a framework to study intelligence and adaptive behavior in animals and plants," *Frontiers in Robotics and AI*, vol. 4, p. 34, 2017.

[15] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE transactions on robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[16] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[17] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: a bioinspired evolution in robotics," *Trends in biotechnology*, vol. 31, no. 5, pp. 287–294, 2013.

[18] K.-J. Bathe, *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[19] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.

[20] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *2013 IEEE international conference on robotics and automation*, pp. 3982–3987, IEEE, 2013.

[21] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[22] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[23] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[24] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979, IEEE, 2019.

[25] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[26] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*, pp. 737–744, IEEE, 2020.

[27] L. R. Treloar, "The elasticity of a network of long-chain molecules—ii," *Transactions of the Faraday Society*, vol. 39, pp. 241–246, 1943.

[28] M. Mooney, "A theory of large elastic deformation," *Journal of applied physics*, vol. 11, no. 9, pp. 582–592, 1940.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[30] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, Ieee, 2004.

[32] C. E. Lemke, "Bimatrix equilibrium points and mathematical programming," *Management science*, vol. 11, no. 7, pp. 681–689, 1965.

[33] G. B. Dantzig and R. Cottle, "Complementary pivot theory of. mathematical programming," *Mathematics of the decision sciences, part*, vol. 1, pp. 115–136, 1968.

[34] B. M. Kwak, "Complementarity problem formulation of three-dimensional frictional contact," *Journal of Applied Mechanics*, vol. 58, no. 1, pp. 134–140, 1991.

[35] T. Hong, J. Yang, and Y.-L. Park, "Model-based control of proprioceptive origami actuators for pneumatic manipulation," *The International Journal of Robotics Research*, p. 02783649251366323, 2025.

[36] T. Hong, J. Lee, B.-H. Song, and Y.-L. Park, "Bridging high-fidelity simulations and physics-based learning using a surrogate model for soft robot control," *Advanced Intelligent Systems*, p. e202500696, 2025.

[37] T. Hong and Y.-L. Park, "Bidirectional mapping between physical contacts and visual tactile images for physics-based simulation," in *2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, pp. 515–522, IEEE, 2025.

[38] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, vol. 82, pp. 1013–1024, 1977.

[39] M. Raous, P. Chabrand, and F. Lebon, "Numerical methods for frictional contact problems and applications," *Journal de mecanique theorique et appliquee*, vol. 7, no. 1, pp. 111–128, 1988.

[40] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, *et al.*, "Sofa: A multi-model framework for interactive physical simulation," *Soft tissue biomechanical modeling for computer assisted surgery*, pp. 283–321, 2012.

[41] O. Goury, B. Carrez, and C. Duriez, "Real-time simulation for control of soft robots with self-collisions using model order reduction for contact forces," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3752–3759, 2021.

[42] D. F. Gomes, P. Paoletti, and S. Luo, "Beyond flat gelsight sensors: Simulation of optical tactile sensors of complex morphologies for sim2real learning," *arXiv preprint arXiv:2305.12605*, 2023.

[43] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "Difftaichi: Differentiable programming for physical simulation," *arXiv preprint arXiv:1910.00935*, 2019.

[44] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.

[45] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "Sofagym: An open platform for reinforcement learning based on soft robot simulations," *Soft Robotics*, vol. 10, no. 2, pp. 410–430, 2023.

[46] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[47] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.

[48] Z. Chen, F. Renda, A. Le Gall, L. Mocellin, M. Bernabei, T. Dangel, G. Ciuti, M. Cianchetti, and C. Stefanini, "Data-driven methods applied to soft robot modeling and control: A review," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2241–2256, 2024.

[49] C. Schaff, A. Sedal, S. Ni, and M. R. Walter, "Sim-to-real transfer of co-optimized soft robot crawlers," *Autonomous Robots*, vol. 47, no. 8, pp. 1195–1211, 2023.

[50] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, "Using first principles for deep learning and model-based control of soft robots," *Frontiers in Robotics and AI*, vol. 8, p. 654398, 2021.

[51] H. Wang, M. Totaro, and L. Beccai, "Toward perceptive soft robots: Progress and challenges," *Advanced science*, vol. 5, no. 9, p. 1800541, 2018.

[52] L. Mullins, "Softening of rubber by deformation," *Rubber chemistry and technology*, vol. 42, no. 1, pp. 339–362, 1969.

[53] J. C. Case, E. L. White, and R. K. Kramer, "Soft material characterization for robotic applications," *Soft Robotics*, vol. 2, no. 2, pp. 80–87, 2015.

[54] J. Walker, T. Zidek, C. Harbel, S. Yoon, F. S. Strickland, S. Kumar, and M. Shin, "Soft robotics: A review of recent developments of pneumatic soft actuators," in *Actuators*, vol. 9, p. 3, MDPI, 2020.

[55] R. W. Ogden, *Non-linear elastic deformations*. Courier Corporation, 1997.

[56] F. Janabi-Sharifi, A. Jalali, and I. D. Walker, "Cosserat rod-based dynamic modeling of tendon-driven continuum robots: A tutorial," *IEEE Access*, vol. 9, pp. 68703–68719, 2021.

[57] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[58] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, and A. Handa, "Gavriel state. 2021. isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[59] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[60] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, *et al.*, "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.

[61] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3930–3937, 2022.

[62] Y. Lin, J. Lloyd, A. Church, and N. F. Lepora, "Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10754–10761, 2022.

[63] Y. Zhao, K. Qian, B. Duan, and S. Luo, "Fots: A fast optical tactile simulator for sim2real learning of tactile-motor robot manipulation skills," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5647–5654, 2024.

[64] Z. Si and W. Yuan, "Taxim: An example-based simulation model for gelsight tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2361–2368, 2022.

[65] I. Akinola, J. Xu, J. Carius, D. Fox, and Y. Narang, "Tacsl: A library for visuotactile sensor simulation and learning," *arXiv preprint arXiv:2408.06506*, 2024.

[66] Z. Si, G. Zhang, Q. Ben, B. Romero, Z. Xian, C. Liu, and C. Gan, "Difftactile: A physics-based differentiable tactile simulator for contact-rich robotic manipulation," *arXiv preprint arXiv:2403.08716*, 2024.

[67] X. Huang, Z. Xu, and C. Xiao, "Twintac: A wide-range, highly sensitive tactile sensor with real-to-sim digital twin sensor model," *arXiv preprint arXiv:2509.10063*, 2025.

[68] S. Suresh, H. Qi, T. Wu, T. Fan, L. Pineda, M. Lambeta, J. Malik, M. Kalakrishnan, R. Calandra, M. Kaess, *et al.*, "Neuralfeels with neural fields: Visuotactile perception for in-hand manipulation," *Science Robotics*, vol. 9, no. 96, p. eadl0628, 2024.

[69] G. M. Caddeo, A. Maracani, P. D. Alfano, N. A. Piga, L. Rosasco, and L. Natale, "Sim2surf: A sim2real surface classifier for vision-based tactile sensors with a bilevel adaptation pipeline," *IEEE Sensors Journal*, 2025.

[70] D. Rus and M. T. Tolley, "Design, fabrication and control of origami robots," *Nature Reviews Materials*, vol. 3, no. 6, pp. 101–112, 2018.

[71] E. A. Peraza-Hernandez, D. J. Hartl, R. J. Malak Jr, and D. C. Lagoudas, "Origami-inspired active structures: a synthesis and review," *Smart Materials and Structures*, vol. 23, no. 9, p. 094001, 2014.

[72] N. Turner, B. Goodwine, and M. Sen, "A review of origami applications in mechanical engineering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, no. 14, pp. 2345–2362, 2016.

[73] M. Meloni, J. Cai, Q. Zhang, D. Sang-Hoon Lee, M. Li, R. Ma, T. E. Parashkevov, and J. Feng, "Engineering origami: A comprehensive review of recent applications, design methods, and tools," *Advanced Science*, vol. 8, no. 13, p. 2000636, 2021.

[74] M. Johnson, Y. Chen, S. Hovet, S. Xu, B. Wood, H. Ren, J. Tokuda, and Z. T. H. Tse, "Fabricating biomedical origami: a state-of-the-art review," *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, pp. 2023–2032, 2017.

[75] Y. Nishiyama, "Miura folding: Applying origami to space exploration," *International Journal of Pure and Applied Mathematics*, vol. 79, no. 2, pp. 269–279, 2012.

[76] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, p. eaah3690, 2016.

[77] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, *et al.*, "Review of machine learning methods in soft robotics," *PLOS One*, vol. 16, no. 2, p. e0246102, 2021.

[78] T. Tachi, "Geometric considerations for the design of rigid origami structures," in *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium*, vol. 12, pp. 458–460, Elsevier Ltd Shanghai, China, 2010.

[79] J. Ji, Q. Luo, and K. Ye, "Vibration control based metamaterials and origami structures: A state-of-the-art review," *Mechanical Systems and Signal Processing*, vol. 161, p. 107945, 2021.

[80] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A geometrically exact model for soft continuum robots: The finite element deformation space formulation," *Soft Robotics*, vol. 6, no. 6, pp. 790–811, 2019.

[81] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 490–513, 2020.

[82] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2018.

[83] Q. Zhang, H. Fang, and J. Xu, "Yoshimura-origami based earthworm-like robot with 3-dimensional locomotion capability," *Frontiers in Robotics and AI*, p. 271, 2021.

[84] J. Kaufmann, P. Bhovad, and S. Li, "Harnessing the multistability of kresling origami for reconfigurable articulation in soft robotic arms," *Soft Robotics*, vol. 9, no. 2, pp. 212–223, 2022.

[85] Q. Zhang, K. Tan, Z. He, H. Pang, Y. Wang, and H. Fang, "An origami continuum manipulator with modularized design and hybrid actuation: accurate kinematic modeling and experiments," *Advanced Intelligent Systems*, vol. 6, no. 4, p. 2300468, 2024.

[86] J. Santoso and C. D. Onal, "An origami continuum robot capable of precise motion through torsionally stiff body and smooth inverse kinematics," *Soft Robotics*, vol. 8, no. 4, pp. 371–386, 2021.

[87] J. Wang and A. Chortos, "Control strategies for soft robot systems," *Advanced Intelligent Systems*, vol. 4, no. 5, p. 2100165, 2022.

[88] M. Li, A. Pal, A. Aghakhani, A. Pena-Francesch, and M. Sitti, "Soft actuators for real-world applications," *Nature Reviews Materials*, vol. 7, no. 3, pp. 235–249, 2022.

[89] M. Luo, E. H. Skorina, W. Tao, F. Chen, S. Ozel, Y. Sun, and C. D. Onal, "Toward modular soft robotics: Proprioceptive curvature sensing and sliding-mode control of soft bidirectional bending modules," *Soft Robotics*, vol. 4, no. 2, pp. 117–125, 2017.

[90] S. Ku, B.-H. Song, T. Park, Y. Lee, and Y.-L. Park, "Soft modularized robotic arm for safe human–robot interaction based on visual and proprioceptive feedback," *The International Journal of Robotics Research*, 2024.

[91] R. L. Truby, C. Della Santina, and D. Rus, "Distributed proprioception of 3D configuration in soft, sensorized robots via deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3299–3306, 2020.

[92] J. Zhou, Y. Chen, X. Chen, Z. Wang, Y. Li, and Y. Liu, "A proprioceptive bellows (PB) actuator with position feedback and force estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1867–1874, 2020.

[93] J. Wirekoh, L. Valle, N. Pol, and Y.-L. Park, "Sensorized flat pneumatic artificial muscles (sfpam) embedded with biomimetic microfluidic sensors for proprioceptive feedback," *Soft Robotics*, vol. 6, no. 6, pp. 768–777, 2019.

[94] A. B. Dawood, H. Godaba, A. Ataka, and K. Althoefer, "Silicone-based capacitive e-skin for exteroception and proprioception," in *Proceedings of the IEEE/ASME International Conference on Intelligent Robots and Systems (IROS)*, pp. 8951–8956, 2020.

[95] L. Scimeca, J. Hughes, P. Maiolino, and F. Iida, "Model-free soft-structure reconstruction for proprioception using tactile arrays," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2479–2484, 2019.

[96] I. Choi, S. J. Yoon, and Y.-L. Park, "Linear electrostatic actuators with moiré-effect optical proprioceptive sensing and electroadhesive braking," *The International Journal of Robotics Research*, 2023.

[97] J. Kang, S. Lee, and Y.-L. Park, "Soft bending actuator with fiber-jamming variable stiffness and fiber-optic proprioception," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7344–7351, 2023.

[98] J. Jung, M. Park, D. Kim, and Y.-L. Park, "Optically sensorized elastomer air chamber for proprioceptive sensing of soft pneumatic actuators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2333–2340, 2020.

[99] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 26, p. eaav1488, 2019.

[100] E. Vander Hoff, D. Jeong, and K. Lee, "Origamibot-i: A thread-actuated origami robot for manipulation and locomotion," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1421–1426, 2014.

[101] Q. Zhang, H. Fang, and J. Xu, "Tunable dynamics in yoshimura origami by harnessing pneumatic pressure," *Journal of Sound and Vibration*, vol. 544, p. 117407, 2023.

[102] K. Westra, F. Dunne, S. Kulsa, M. Hunt, and J. Leachman, "Compliant polymer origami bellows in cryogenics," *Cryogenics*, vol. 114, p. 103226, 2021.

[103] E. T. Filipov, T. Tachi, and G. H. Paulino, "Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials," *Proceedings of the National Academy of Sciences*, vol. 112, no. 40, pp. 12321–12326, 2015.

[104] T. H. Hong, S.-H. Park, J.-H. Park, N.-J. Paik, and Y.-L. Park, "Design of pneumatic origami muscle actuators (POMAs) for a soft robotic hand orthosis for grasping assistance," in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 627–632, 2020.

[105] S. Li, J. J. Stampfli, H. J. Xu, E. Malkin, E. V. Diaz, D. Rus, and R. J. Wood, "A vacuum-driven origami "magic-ball" soft gripper," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7401–7408, 2019.

[106] B. Chen, Z. Shao, Z. Xie, J. Liu, F. Pan, L. He, L. Zhang, Y. Zhang, X. Ling, F. Peng, *et al.*, "Soft origami gripper with variable effective length," *Advanced Intelligent Systems*, vol. 3, no. 10, p. 2000251, 2021.

[107] H. Suzuki and R. J. Wood, "Origami-inspired miniature manipulator for teleoperated microsurgery," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 437–446, 2020.

[108] M. Sivaperuman Kalairaj, C. J. Cai, and H. Ren, "Untethered origami worm robot with diverse multi-leg attachments and responsive motions under magnetic actuation," *Robotics*, vol. 10, no. 4, p. 118, 2021.

[109] M. Yu, W. Yang, Y. Yu, X. Cheng, and Z. Jiao, "A crawling soft robot driven by pneumatic foldable actuators based on miura-ori," in *Actuators*, vol. 9, p. 26, MDPI, 2020.

[110] A. Zaghloul and G. M. Bone, "Origami-inspired soft pneumatic actuators: Generalization and design optimization," in *Actuators*, vol. 12, p. 72, MDPI, 2023.

[111] J.-G. Lee and H. Rodrigue, "Origami-based vacuum pneumatic artificial muscles with large contraction ratios," *Soft Robotics*, vol. 6, no. 1, pp. 109–117, 2019.

[112] X. Zou, T. Liang, M. Yang, C. LoPresti, S. Shukla, M. Akin, B. T. Weil, S. Hoque, E. Gruber, and A. D. Mazzeo, "Based robotics with stackable pneumatic actuators," *Soft Robotics*, 2021.

[113] K. Liu and G. Paulino, "Nonlinear mechanics of non-rigid origami: an efficient computational approach," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2206, p. 20170348, 2017.

[114] G. W. Hunt and I. Ario, "Twist buckling and the foldable cylinder: an exercise in origami," *International Journal of Non-Linear Mechanics*, vol. 40, no. 6, pp. 833–843, 2005.

[115] K. Liu and G. H. Paulino, "Merlin: A matlab implementation to capture highly nonlinear behavior of non-rigid origami," in *Proceedings of IASS Annual Symposia*, vol. 2016, pp. 1–10, International Association for Shell and Spatial Structures (IASS), 2016.

[116] Y. Zhu, M. Schenk, and E. T. Filipov, "A review on origami simulations: From kinematics, to mechanics, toward multiphysics," *Applied Mechanics Reviews*, vol. 74, no. 3, p. 030801, 2022.

[117] M. Russo, S. M. H. Sadati, X. Dong, A. Mohammad, I. D. Walker, C. Bergeles, K. Xu, and D. A. Axinte, "Continuum robots: An overview," *Advanced Intelligent Systems*, vol. 5, no. 5, p. 2200367, 2023.

[118] L. A. Al Abeach, S. Nefti-Meziani, and S. Davis, "Design of a variable stiffness soft dexterous gripper," *Soft Robotics*, vol. 4, no. 3, pp. 274–284, 2017.

[119] J.-E. Suh, Y. Miyazawa, J. Yang, and J.-H. Han, "Self-reconfiguring and stiffening origami tube," *Advanced Engineering Materials*, vol. 24, no. 5, p. 2101202, 2022.

[120] Z. Zhang, G. Chen, W. Fan, W. Yan, L. Kong, and H. Wang, "A stiffness variable passive compliance device with reconfigurable elastic inner skeleton and origami shell," *Chinese Journal of Mechanical Engineering*, vol. 33, pp. 1–13, 2020.

[121] R. V. Martinez, C. R. Fish, X. Chen, and G. M. Whitesides, "Elastomeric origami: programmable paper-elastomer composites as pneumatic actuators," *Advanced Functional Materials*, vol. 22, no. 7, pp. 1376–1384, 2012.

[122] J. Cai, X. Deng, Y. Xu, and J. Feng, "Motion analysis of a foldable barrel vault based on regular and irregular yoshimura origami," *Journal of Mechanisms and Robotics*, vol. 8, no. 2, p. 021017, 2016.

[123] C.-P. Chou and B. Hannaford, "Measurement and modeling of mckibben pneumatic artificial muscles," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 90–102, 1996.

[124] M. A. E. Kshad, C. Popinigis, and H. E. Naguib, "3D printing of ronresch-like origami cores for compression and impact load damping," *Smart Materials and Structures*, vol. 28, no. 1, p. 015027, 2018.

[125] K. Ho-Le, "Finite element mesh generation methods: a review and classification," *Computer-aided Design*, vol. 20, no. 1, pp. 27–38, 1988.

[126] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006.

[127] H. Yasuda, T. Yein, T. Tachi, K. Miura, and M. Taya, "Folding behaviour of tachi–miura polyhedron bellows," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 469, no. 2159, p. 20130351, 2013.

[128] J. D. Greer, T. K. Morimoto, A. M. Okamura, and E. W. Hawkes, "Series pneumatic artificial muscles (spams) and application to a soft continuum robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5503–5510, 2017.

[129] L. Hao, C. Xiang, M. E. Giannaccini, H. Cheng, Y. Zhang, S. Nefti-Meziani, and S. Davis, "Design and control of a novel variable stiffness soft arm," *Advanced Robotics*, vol. 32, no. 11, pp. 605–622, 2018.

[130] Z. Zhang, X. Wang, S. Wang, D. Meng, and B. Liang, "Design and modeling of a parallel-pipe-crawling pneumatic soft robot," *IEEE access*, vol. 7, pp. 134301–134317, 2019.

[131] J. Huang, J. Zhou, Z. Wang, J. Law, H. Cao, Y. Li, H. Wang, and Y. Liu, "Modular origami soft robot with the perception of interaction force and body configuration," *Advanced Intelligent Systems*, vol. 4, no. 9, p. 2200081, 2022.

[132] Y. Toshimitsu, K. W. Wong, T. Buchner, and R. Katzschmann, "Sopra: Fabrication & dynamical modeling of a scalable soft continuum robotic arm with integrated proprioceptive sensing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 653–660, IEEE, 2021.

[133] M. A. Robertson, O. C. Kara, and J. Paik, "Soft pneumatic actuator-driven origami-inspired modular robotic "pneumagami"," *The International Journal of Robotics Research*, vol. 40, no. 1, pp. 72–85, 2021.

[134] Z. Shen, Y. Zhao, H. Zhong, K. Tang, Y. Chen, Y. Xiao, J. Yi, S. Liu, and Z. Wang, "Soft origami optical-sensing actuator for underwater manipulation," *Frontiers in Robotics and AI*, vol. 7, p. 616128, 2021.

[135] S. Liu, J. Liu, K. Zou, X. Wang, Z. Fang, J. Yi, and Z. Wang, "A Six Degrees-of-Freedom Soft Robotic Joint With Tilt-Arranged Origami Actuator," *Journal of Mechanisms and Robotics*, vol. 14, p. 060912, 06 2022.

[136] W. Fan, J. Wang, Z. Zhang, G. Chen, and H. Wang, "Vacuum-driven parallel continuum robots with self-sensing origami linkages," *IEEE/ASME Transactions on Mechatronics*, 2024.

[137] Y. X. Mak, A. Dijkshoorn, and M. Abayazid, "Design methodology for a 3d printable multi-degree of freedom soft actuator using geometric origami patterns," *Advanced Intelligent Systems*, p. 2300666, 2024.

[138] N. Kidambi and K. Wang, "Dynamics of kresling origami deployment," *Physical Review E*, vol. 101, no. 6, p. 063003, 2020.

[139] J. Yang, D. Tang, J. Ao, T. Ghosh, T. V. Neumann, D. Zhang, Y. Piskarev, T. Yu, V. K. Truong, K. Xie, Y.-C. Lai, Y. Li, and M. D. Dickey, "Ultrasoft liquid metal elastomer foams with positive and negative piezopermittivity for tactile sensing," *Advanced Functional Materials*, vol. 30, no. 36, p. 2002611, 2020.

[140] Y.-L. Park, B.-r. Chen, and R. J. Wood, "Design and fabriication of soft artificial skin using embedded microchannels and liquid conductors," *IEEE Sensors Journal*, vol. 12, no. 8, pp. 2711–2718, 2012.

[141] J. Jung, E. Lee, J. Kim, and Y.-L. Park, "Ultra-thin multi-modal soft sensor using liquid-metal thin-film deposition for enhanced human-robot interaction," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5269–5275, 2024.

[142] J. Weichart, C. Roman, and C. Hierold, "Tactile sensing with scalable capacitive sensor arrays on flexible substrates," *Journal of Microelectromechanical Systems*, vol. 30, no. 6, pp. 915–929, 2021.

[143] J. Yi, B. Kim, K.-J. Cho, , and Y.-L. Park, "Underactuated robotic gripper with fiber-optic force sensing tendons," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7607–7614, 2023.

[144] D. Kim, S. Lee, T. Hong, and Y.-L. Park, "Exploration-based model learning with self-attention for risk-sensitive robot control," *npj Robotics*, vol. 1, no. 7, 2023.

[145] G. M. Whitesides, "Soft robotics," *Angewandte Chemie International Edition*, vol. 57, no. 16, pp. 4258–4273, 2018.

[146] J. Shintake, V. Cacucciolo, D. Floreano, and H. Shea, "Soft robotic grippers," *Advanced materials*, vol. 30, no. 29, p. 1707035, 2018.

[147] C. Majidi, "Soft robotics: a perspective—current trends and prospects for the future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, 2014.

[148] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced robotics*, vol. 26, no. 7, pp. 709–727, 2012.

[149] C. Della Santina, C. Duriez, and D. Rus, "Model-based control of soft robots: A survey of the state of the art and open challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, pp. 30–65, 2023.

[150] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, and F. Renda, "Soft robots modeling: A structured overview," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1728–1748, 2023.

[151] M. Cianchetti, C. Laschi, A. Menciassi, and P. Dario, "Biomedical applications of soft robotics," *Nature Reviews Materials*, vol. 3, no. 6, pp. 143–153, 2018.

[152] P. Polygerinos, N. Correll, S. A. Morin, B. Mosadegh, C. D. Onal, K. Petersen, M. Cianchetti, M. T. Tolley, and R. F. Shepherd, "Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human-robot interaction," *Advanced engineering materials*, vol. 19, no. 12, p. 1700016, 2017.

[153] M. S. Nazeer, C. Laschi, and E. Falotico, "Rl-based adaptive controller for high precision reaching in a soft robot arm," *IEEE Transactions on Robotics*, 2024.

[154] Q. Wang, Z. Wu, J. Huang, Z. Du, Y. Yue, D. Chen, D. Li, and B. Su, "Integration of sensing and shape-deforming capabilities for a bioinspired soft robot," *Composites Part B: Engineering*, vol. 223, p. 109116, 2021.

[155] Y. Choi, G. Shin, S. J. Yoon, and Y.-L. Park, "Soft electromagnetic sliding actuators for highly compliant planar motions using microfluidic conductive coil array," *Soft Robotics*, 2024.

[156] L. Ding, L. Niu, Y. Su, H. Yang, G. Liu, H. Gao, and Z. Deng, "Dynamic finite element modeling and simulation of soft robots," *Chinese journal of mechanical engineering*, vol. 35, no. 1, p. 24, 2022.

[157] J. M. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion.," in *Robotics: Science and Systems*, vol. 1, 2019.

[158] K. Chin, T. Hellebrekers, and C. Majidi, "Machine learning for soft robotic sensing and control," *Advanced Intelligent Systems*, vol. 2, no. 6, p. 1900171, 2020.

[159] B. Jumet, M. D. Bell, V. Sanchez, and D. J. Preston, "A data-driven review of soft robotics," *Advanced Intelligent Systems*, vol. 4, no. 4, p. 2100163, 2022.

[160] H. Zhang, R. Cao, S. Zilberstein, F. Wu, and X. Chen, "Toward effective soft robot control via reinforcement learning," in *Intelligent Robotics and Applications: 10th International Conference, ICIRA 2017, Wuhan, China, August 16–18, 2017, Proceedings, Part I 10*, pp. 173–184, Springer, 2017.

[161] M. Raeisinezhad, N. Pagliocca, B. Koohbor, and M. Trkov, "Design optimization of a pneumatic soft robotic actuator using model-based optimization and deep reinforcement learning," *Frontiers in Robotics and AI*, vol. 8, p. 639102, 2021.

[162] D. Kim, S. Lee, T. H. Hong, and Y.-L. Park, "Exploration-based model learning with self-attention for risk-sensitive robot control," *npj Robotics*, vol. 1, no. 1, p. 7, 2023.

[163] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, p. 4, 2019.

[164] T. Yang, Y. Xiao, Z. Zhang, Y. Liang, G. Li, M. Zhang, S. Li, T.-W. Wong, Y. Wang, T. Li, *et al.*, "A soft artificial muscle driven robot with reinforcement learning," *Scientific reports*, vol. 8, no. 1, p. 14518, 2018.

[165] C. Della Santina, R. K. Katzschmann, A. Biechi, and D. Rus, "Dynamic control of soft robots interacting with the environment," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 46–53, IEEE, 2018.

[166] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, "Dynamic simulation of articulated soft robots," *Nature communications*, vol. 11, no. 1, p. 2233, 2020.

[167] Z. Wan, Y. Sun, Y. Qin, E. H. Skorina, R. Gasoto, M. Luo, J. Fu, and C. D. Onal, "Design, analysis, and real-time simulation of a 3d soft robotic snake," *Soft Robotics*, vol. 10, no. 2, pp. 258–268, 2023.

[168] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2550–2555, IEEE, 2015.

[169] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, *et al.*, "Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, no. 22, pp. 1208–1224, 2017.

[170] M. A. Graule, T. P. McCarthy, C. B. Teeple, J. Werfel, and R. J. Wood, "Somogym: A toolkit for developing and evaluating controllers and reinforcement learning algorithms for soft robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4071–4078, 2022.

[171] L. Qin, H. Peng, X. Huang, M. Liu, and W. Huang, "Modeling and simulation of dynamics in soft robotics: A review of numerical approaches," *Current Robotics Reports*, vol. 5, no. 1, pp. 1–13, 2024.

[172] S. Ku, B.-H. Song, T. Park, Y. Lee, and Y.-L. Park, "Soft modularized robotic arm for safe human–robot interaction based on visual and proprioceptive feedback," *The International Journal of Robotics Research*, p. 02783649241227249, 2024.

[173] W. Dou, G. Zhong, J. Cao, Z. Shi, B. Peng, and L. Jiang, "Soft robotic manipulators: Designs, actuation, stiffness tuning, and sensing," *Advanced Materials Technologies*, vol. 6, no. 9, p. 2100018, 2021.

[174] Z. Gong, B. Chen, J. Liu, X. Fang, Z. Liu, T. Wang, and L. Wen, "An opposite-bending-and-extension soft robotic manipulator for delicate grasping in shallow water," *Frontiers in Robotics and AI*, vol. 6, p. 26, 2019.

[175] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, "Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer," in *2019 2nd IEEE international conference on soft robotics (RoboSoft)*, pp. 717–724, IEEE, 2019.

[176] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[177] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[178] Z. Zhao, X. Ding, and B. A. Prakash, "Pinnsformer: A transformer-based framework for physics-informed neural networks," *arXiv preprint arXiv:2307.11833*, 2023.

[179] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[180] G. Brockman, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[181] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[182] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[183] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[184] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[185] X. Liu, C. D. Onal, and J. Fu, "Reinforcement learning of cpg-regulated locomotion controller for a soft snake robot," *IEEE Transactions on Robotics*, 2023.

[186] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, IEEE, 2024.

[187] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[188] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[189] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine, "The feeling of success: Does touch sensing help predict grasp outcomes?," *arXiv preprint arXiv:1710.05512*, 2017.

[190] H. Yousef, M. Boukallel, and K. Althoefer, "Tactile sensing for dexterous in-hand manipulation in robotics—a review," *Sensors and Actuators A: physical*, vol. 167, no. 2, pp. 171–187, 2011.

[191] C. Y. Wong and W. Suleiman, "Sensor observability index: Evaluating sensor alignment for task-space observability in robotic manipulators," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1276–1282, IEEE, 2022.

[192] I. Sorrentino, G. Romualdi, and D. Pucci, "Ukf-based sensor fusion for joint-torque sensorless humanoid robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13150–13156, IEEE, 2024.

[193] A. C. Abad and A. Ranasinghe, "Visuotactile sensors with emphasis on gelsight sensor: A review," *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7628–7638, 2020.

[194] S. Zhang, Z. Chen, Y. Gao, W. Wan, J. Shan, H. Xue, F. Sun, Y. Yang, and B. Fang, "Hardware technology of vision-based tactile sensor: A review," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21410–21427, 2022.

[195] V. Kakani, X. Cui, M. Ma, and H. Kim, "Vision-based tactile sensor mechanism for the estimation of contact position and force distribution using deep learning," *Sensors*, vol. 21, no. 5, p. 1920, 2021.

[196] S. Wang, Y. She, B. Romero, and E. Adelson, "Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6468–6475, IEEE, 2021.

[197] C. Higuera, A. Sharma, C. K. Bodduluri, T. Fan, P. Lancaster, M. Kalakrishnan, M. Kaess, B. Boots, M. Lambeta, T. Wu, *et al.*, "Sparsh: Self-supervised touch representations for vision-based tactile sensing," *arXiv preprint arXiv:2410.24090*, 2024.

[198] R. Sui, L. Zhang, T. Li, and Y. Jiang, "Incipient slip detection method for soft objects with vision-based tactile sensor," *Measurement*, vol. 203, p. 111906, 2022.

[199] F. Yang, C. Ma, J. Zhang, J. Zhu, W. Yuan, and A. Owens, "Touch and go: Learning from human-collected vision and touch," *arXiv preprint arXiv:2211.12498*, 2022.

[200] Y. Li, J.-Y. Zhu, R. Tedrake, and A. Torralba, "Connecting touch and vision via cross-modal prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10609–10618, 2019.

[201] Z. Chen, N. Ou, X. Zhang, Z. Wu, Y. Zhao, Y. Wang, N. Lepora, L. Jamone, J. Deng, and S. Luo, "General force sensation for tactile robot," *arXiv preprint arXiv:2503.01058*, 2025.

[202] E. Su, C. Jia, Y. Qin, W. Zhou, A. Macaluso, B. Huang, and X. Wang, "Sim2real manipulation on unknown objects with tactile-based reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9234–9241, IEEE, 2024.

[203] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-rl for insertion: Generalization to objects of unknown geometry," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6437–6443, IEEE, 2021.

[204] Q. Liu, Y. Cui, Z. Sun, G. Li, J. Chen, and Q. Ye, "VTDexmanip: A dataset and benchmark for visual-tactile pretraining and dexterous manipulation with reinforcement learning," in *The Thirteenth International Conference on Learning Representations*, 2025.

[205] D. H. Nguyen, T. Schneider, G. Duret, A. Kshirsagar, B. Belousov, and J. Peters, "Tacex: Gelsight tactile simulation in isaac

sim–combining soft-body and visuotactile simulators," *arXiv preprint arXiv:2411.04776*, 2024.

[206] Q. K. Luu, N. H. Nguyen, *et al.*, "Simulation, learning, and application of vision-based tactile sensing at large scale," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2003–2019, 2023.

[207] L. Van Duong *et al.*, "Large-scale vision-based tactile sensing for robot links: Design, modeling, and evaluation," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 390–403, 2020.

[208] L. Zhang, T. Li, and Y. Jiang, "Improving the force reconstruction performance of vision-based tactile sensors by optimizing the elastic body," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1109–1116, 2023.

[209] G. M. Caddeo, A. Maracani, P. D. Alfano, N. A. Piga, L. Rosasco, and L. Natale, "Sim2real bilevel adaptation for object surface classification using vision-based tactile sensors," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15128–15134, IEEE, 2024.

[210] C. Higuera, B. Boots, and M. Mukadam, "Learning to read braille: Bridging the tactile reality gap with diffusion models," *arXiv preprint arXiv:2304.01182*, 2023.

[211] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 351–358, 1995.

[212] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3d gaussian splatting as new era: A survey," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[213] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[214] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*, pp. 510–517, IEEE, 2015.

[215] Y. Wu, Z. Chen, F. Wu, L. Chen, L. Zhang, Z. Bing, A. Swikir, S. Haddadin, and A. Knoll, "Tacdiffusion: Force-domain diffusion policy for precise tactile manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11831–11837, IEEE, 2025.

[216] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[217] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[218] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *Advanced Robotics*, vol. 38, no. 18, pp. 1232–1254, 2024.

[219] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[220] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[221] M. Lambeta, H. Xu, J. Xu, P.-W. Chou, S. Wang, T. Darrell, and R. Calandra, "Pytouch: A machine learning library for touch processing," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13208–13214, IEEE, 2021.

[222] X.-P. Wang, C. J. Garcıa-Cervera, *et al.*, "A gauss–seidel projection method for micromagnetics simulations," *Journal of Computational Physics*, vol. 171, no. 1, pp. 357–372, 2001.

[223] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[224] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, (New York, NY, USA), p. 43–54, Association for Computing Machinery, 1998.

[225] T. Kipf, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[226] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

Appendix A.

Data-Driven Lookup Table (LUT) Method for OCM Mapping

The mapping functions $\mathbb{F}_A$ and $\mathbb{F}_B$ for the OCM were computed numerically using a data-driven approach, as it is challenging to express these functions explicitly due to the complexity of the origami structure. The LUT method is used to store and efficiently search the forward and the inverse mapping relationships between strain (ϵ), pressure (P), force (F) and moment (M).

A.1. Forward Mapping Function Derivation

The forward mapping functions $\mathbb{F}_A$ and $\mathbb{F}_B$ are numerically calculated and stored in a structured data format. These functions are derived by discretizing the strain and bending angle relationships and evaluating the force and moment through numerical differentiation. The function $\mathbb{F}_A$ defines the axial force as a function of strain (ϵ) and internal pressure (P) as shown in

$$F_z = \mathbb{F}_{\mathbb{A}}(\epsilon, P), \tag{A.1}$$

which the same equation as in the main text. Since the force is derived from the strain response, numerical differentiation is applied:

$$F_z(\epsilon, P) = \frac{\partial \Pi(\epsilon, P)}{\partial \epsilon} \approx \frac{\Pi(\epsilon + \Delta\epsilon, P) - \Pi(\epsilon, P)}{\Delta\epsilon}, \tag{A.2}$$

where $\Pi(\epsilon, P)$ represents the potential energy stored in the system. The differentiation step $\Delta\epsilon$ is set to a small value to ensure high resolution in the LUT.

The function $\mathbb{F}_B$ defines the moment as a function of the bending angle

θ_C and the arc length S_C:

$$M = \mathbb{F}_{\mathbb{B}}(\theta_C, S_C). \tag{A.3}$$

The moment is computed through numerical differentiation based on the strain energy in multiple dihedral angles within the OCM:

$$M(\theta_C, S_C) = \frac{\partial \Pi(\theta_C, S_C)}{\partial \theta_C} \approx \frac{\Pi(\theta_C + \Delta\theta, S_C) - \Pi(\theta_C, S_C)}{\Delta\theta}. \tag{A.4}$$

By applying these numerical differentiation methods, fine-step LUT data is generated for both $\mathbb{F}_A$ and $\mathbb{F}_B$, providing high-resolution mappings.

A.2. Inverse Mapping Function Derivation

Inverse mapping functions are constructed by inferring the target variable from precomputed LUT data. Given the difficulty in explicitly solving these relationships, the inverse function is inferred by grouping related values and performing numerical interpolation.

The inverse function ${\mathbb{F}_{\mathbb{A}}}^{\dagger}$ estimates strain given a known force and pressure:

$$\epsilon = {\mathbb{F}_{\mathbb{A}}}^{\dagger}(F_z, P). \tag{A.5}$$

The inference process is based on searching for the closest entry in the LUT:

$$\epsilon^* = \arg\min_{\epsilon} \left| \mathbb{F}_{\mathbb{A}}(\epsilon, P) - F_z \right|. \tag{A.6}$$

A search algorithm such as binary search was applied to efficiently retrieve the inverse-mapped value.

Similarly, the inverse function ${\mathbb{F}_{\mathbb{B}}}^{\dagger}$ estimates the bending angle given a known moment and arc length (so called back bone length):

$$\theta_C = {\mathbb{F}_{\mathbb{B}}}^{\dagger}(M, S_C). \tag{A.7}$$

The inference is performed by searching for the best match in the precomputed

LUT:

$$\theta_C^* = \arg\min_{\theta_C} \left| \mathbb{F}_{\mathbb{B}}(\theta_C, S_C) - M \right|. \tag{A.8}$$

A.3. Efficient LUT Computation and Deployment

The LUT computation was performed on a high-performance workstation to enable high-resolution numerical differentiation and efficient inverse function inference. The LUT was structured as a multi-dimensional table with finely spaced intervals of strain, pressure, bending angle, and moment values.

To ensure efficient retrieval of values, the LUT was implemented using binary search interpolation, which allowed a logarithmic search approach for rapid inverse mapping retrieval. Additionally, spline interpolation was used to smooth the approximation method to estimate values between LUT entries. The optimized search allows rapid inference of inverse function outputs, reducing computational overhead for real-time applications.

These functions were set in the Python script, which did not require the memory storage in the MCU board. The final optimized LUT was deployed to the MCU by compressing the table into fixed-point format for efficient search. The target pressure values were computed on the workstation and transmitted to the MCU using a structured communication protocol:

$$P = f_{OCM}^{\dagger\dagger}(\tau_{\text{ext}}, C). \tag{A.9}$$

The MCU received the optimized pressure setpoints and executed real-time control adjustments based on the precomputed inverse function mappings.

Appendix B.

Kinematics and Force Analysis of a Two-Segment Origami Manipulator

The two-segment origami manipulator consists of two modular sections connected in series, each comprising multiple extended origami cylinder modules (extended OCMs in Figure 1). This continuum-type robot was built to demonstrate the application of the developed manipulator. The estimation of the 3D configuration is explained in this section, which is the extended version of modeling introduced in the main text. Unlike a single-section manipulator, the deformation of each segment is influenced not only by the external force applied at the tool center point (TCP) but also by the cumulative weight of the upper segment. Figure .B.1-(A) illustrates the segmented structure, where each segment is characterized by its arc length (S_{Ck}), bending angle (θ_{Ck}), and bending direction (ϕ_{Ck}). Each extended OCM within a given segment shares the same internal pressure, resulting in uniform deformation across the segment under the piecewise constant curvature (PCC) assumption. For the manipulation task, the continuum robot was fixed at the frame making the robot upside down as shown in Figure 8 and the Figure S.B.1-(A).

The two-segment origami manipulator undergoes deformations based on the applied pressures(P) and external forces (F) at the TCP. The force and moment equilibrium conditions are established to compute the internal forces, bending moments, and interaction effects between the two segments. The final TCP position and orientation are then estimated using measured actuator lengths from Hall sensors and orientation data from the IMU.

B.1. Force Equilibrium Equations

Each extended OCM consists of three actuators positioned at 120° intervals. The pressures in these actuators generate internal forces that contribute to the general deformation of the manipulator. Given the input pressures ($P_1, P_2, P_3, P_4, P_5,$ and P_6) from the pressure regulators and the force applied at the TCP ($F_x, F_y,$ andF_z). The axial force in each OCM is determined using the force-strain mapping function:

$$F_{z,i} = \mathbb{F}_{\mathbb{A}}(\epsilon_i, P_i), \quad i = 1, 2, 3, 4, 5, 6. \tag{B.1}$$

The external force applied at the TCP propagates downward, influencing both segments. The force equilibrium equation for the first segment is written as

$$F_{z1,1} + F_{z2,1} + F_{z3,1} = F_z + W_1, \text{ where} \tag{B.2}$$

F_z is the z-direction component of the τ_{ext}. The second segment must support both its own structure and the reaction force from the first segment. The force equilibrium equation for the second segment is

$$F_{z1,2} + F_{z2,2} + F_{z3,2} = R_{z1,1} + R_{z2,1} + R_{z3,1} + W_2. \tag{B.3}$$

where $R_{z1,1}, R_{z2,1}, R_{z3,1}$ are the reaction forces of the interface at the connection point between the two segments. The horizontal force equilibrium conditions are:

$$F_{x1,1} + F_{x2,1} + F_{x3,1} = F_x, \quad F_{y1,1} + F_{y2,1} + F_{y3,1} = F_y. \tag{B.4}$$

The moment equilibrium for the first segment must now include the unknown reaction moment from Segment 2 at the interface as:

$$M_{x,1} - M_{\text{int},x} = d(F_{z2,1} \sin 120^\circ + F_{z3,1} \sin 240^\circ) \text{ and} \tag{B.5}$$

$$M_{y,1} - M_{\text{int},y} = d(F_{z1,1} - F_{z2,1} \cos 120^\circ - F_{z3,1} \cos 240^\circ), \tag{B.6}$$

where $M_{\text{int},x}$ and $M_{\text{int},y}$ are the unknown interactions at the interface. Similarly, the moment equilibrium for the second segment must incorporate the interaction moment:

$$M_{x,2} + M_{\text{int},x} = d(F_{z2,2} \sin 120^\circ + F_{z3,2} \sin 240^\circ) \text{ and} \tag{B.7}$$

$$M_{y,2} + M_{\text{int},y} = d(F_{z1,2} - F_{z2,2} \cos 120^\circ - F_{z3,2} \cos 240^\circ). \tag{B.8}$$

Due to the high torsional stiffness of the structure, torsional moments remain negligible:

$$M_{z,1} = 0, \quad M_{z,2} = 0. \tag{B.9}$$

For the second segment, the reaction moment at the base supports the external force at the TCP:

$$M_{\text{int},x} = d(F_{z2,2} \sin 120^\circ + F_{z3,2} \sin 240^\circ), \tag{B.10}$$

$$M_{\text{int},y} = d(F_{z1,2} - F_{z2,2} \cos 120^\circ - F_{z3,2} \cos 240^\circ). \tag{B.11}$$

The interaction moment is computed using the bending response of the second segment:

$$M_{\text{int},x} = \mathbb{F}_{\mathbb{B}}{}^{\dagger}(\theta_{C2}, S_{C2}), \tag{B.12}$$

$$M_{\text{int},y} = \mathbb{F}_{\mathbb{B}}{}^{\dagger}(\theta_{C2}, S_{C2}). \tag{B.13}$$

Solving for the actuator forces in the first segment are

$$\begin{aligned} F_{z1,1} &= \frac{F_z + W_1 + \frac{M_{y,1} - M_{\text{int},y}}{d} + \frac{M_{x,1} - M_{\text{int},x}}{\sqrt{3}d}}{3}, \\ F_{z2,1} &= \frac{F_z + W_1 - \frac{2(M_{x,1} - M_{\text{int},x})}{\sqrt{3}d}}{3} \text{ and} \\ F_{z3,1} &= \frac{F_z + W_1 - \frac{M_{y,1} - M_{\text{int},y}}{d} + \frac{M_{x,1} - M_{\text{int},x}}{\sqrt{3}d}}{3}. \end{aligned} \tag{B.14}$$

Solving for the actuator forces in the second segment are

$$
\begin{aligned}
F_{z1,2} &= \frac{R_{z1,1} + R_{z2,1} + R_{z3,1} + W_2 + \frac{M_{y,2}+M_{\text{int},y}}{d} + \frac{M_{x,2}+M_{\text{int},x}}{\sqrt{3}d}}{3}, \\
F_{z2,2} &= \frac{R_{z1,1} + R_{z2,1} + R_{z3,1} + W_2 - \frac{2(M_{x,2}+M_{\text{int},x})}{\sqrt{3}d}}{3} \text{ and} \\
F_{z3,2} &= \frac{R_{z1,1} + R_{z2,1} + R_{z3,1} + W_2 - \frac{M_{y,2}+M_{\text{int},y}}{d} + \frac{M_{x,2}+M_{\text{int},x}}{\sqrt{3}d}}{3}.
\end{aligned}
\tag{B.15}
$$

B.2. TCP Position and Orientation Estimation

The arc lengths for each segment are determined from the Hall sensor measurements:

$$
S_{C1} = \frac{el_1 + el_2 + el_3}{3}, \quad S_{C2} = \frac{el_4 + el_5 + el_6}{3}. \tag{B.16}
$$

The segment-wise bending angles are estimated using the inverse moment-curvature mapping:

$$
\theta_{C1} = \mathbb{F}_{\mathbb{B}}{}^{\dagger}(M_{x,1}, S_{C1}), \quad \theta_{C2} = \mathbb{F}_{\mathbb{B}}{}^{\dagger}(M_{x,2}, S_{C2}). \tag{B.17}
$$

The bending directions are given by:

$$
\phi_{C1} = \tan^{-1}\left(\frac{\theta_y}{\theta_x}\right), \quad \phi_{C2} = \tan^{-1}\left(\frac{\theta_y}{\theta_x}\right). \tag{B.18}
$$

Using successive transformations, the TCP position is obtained as:

$$
T^B_{\text{TCP}} = T^B_1 T^1_2. \tag{B.19}
$$

where

$$
p_{\text{TCP}} = R_1 p_2 + p_1. \tag{B.20}
$$

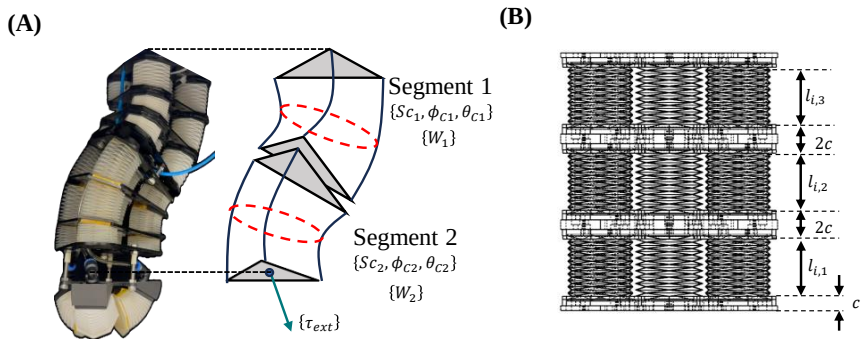


Figure B.1: (A) Continuum robot composed of two segments of the origami manipulator. (B) Length information in the manipulator segment.

Expanding the transformation equations:

$$\begin{bmatrix} X_{\text{TCP}} \\ Y_{\text{TCP}} \\ Z_{\text{TCP}} \end{bmatrix} = R_1 \begin{bmatrix} S_{C2} \sin\theta_{C2} \cos\phi_{C2} \\ S_{C2} \sin\theta_{C2} \sin\phi_{C2} \\ S_{C2} \cos\theta_{C2} + 2c \end{bmatrix} + \begin{bmatrix} S_{C1} \sin\theta_{C1} \cos\phi_{C1} \\ S_{C1} \sin\theta_{C1} \sin\phi_{C1} \\ S_{C1} \cos\theta_{C1} \end{bmatrix}. \tag{B.21}$$

The force and moment equilibrium equations couple the deformation responses of the two segments. The interaction moment is derived using the inverse moment-curvature function, ensuring that force transmission between segments is accurately modeled. The TCP position and orientation are computed using Hall sensor measurements for segment lengths and IMU measurements for orientation tracking. The use of SE(3) transformations provides a structured approach to estimating the pose of the end effector.

Appendix C.

Calibration of the Hall-effect sensor

The actuator is equipped with a Hall effect sensor (SS41, Honeywell) embedded within it to measure the magnitude of the surrounding magnetic field, as shown in Figure S.C.1-(A). The distance to the magnet can be estimated by placing it near the sensor and measuring the output signal. This output measurement is then used to estimate the entire length of the actuator, as explained in the modeling section. A highly linear mapping between the measured voltage and the distance value is shown in Figure S.C.1-(B). The sensor is connected to an analog-digital converter (ADS1115, Texas Instruments, USA), reading 16-bit digits to measure the distance between the neodymium magnet and the sensor. The mapping voltage ranges from 0 V to 5 V, with a measurement range of at least 1.9 V to 2.1 V. To ensure that the voltage matches the distance data, the actual distance measurement is carried out using a linear encoder of the tensile tester machine, which is the same machine introduced in the main text. The mean error of the Hall effect sensor was 0.231 mm average, and it exhibits a high level of linearity, up to R^2=0.96 in distance and voltage. In cases where the neodymium magnet is not aligned with the Hall effect sensor, such as during bending, we tested whether the sensor could still provide accurate linear distance output. The first assumption was that the length (s) could be measured directly, even when the origami configuration was bent, as shown in Figure S.C.1-(A). The experimental setup to collect Hall effect sensor signals during bending deformation is shown in Figure S.C.1-(C), where the OCM was moved through all possible orientations (q) with a fixed backbone length (s). The error was defined as the difference between the initial length of the backbone (s) and the measured distance value. The error is shown in Figure S.C.1-(D), with the mean error (red box) and

standard deviation boundaries (purple box). The average measurement error was 0.131 mm, which was acceptable for measuring the actuator length.

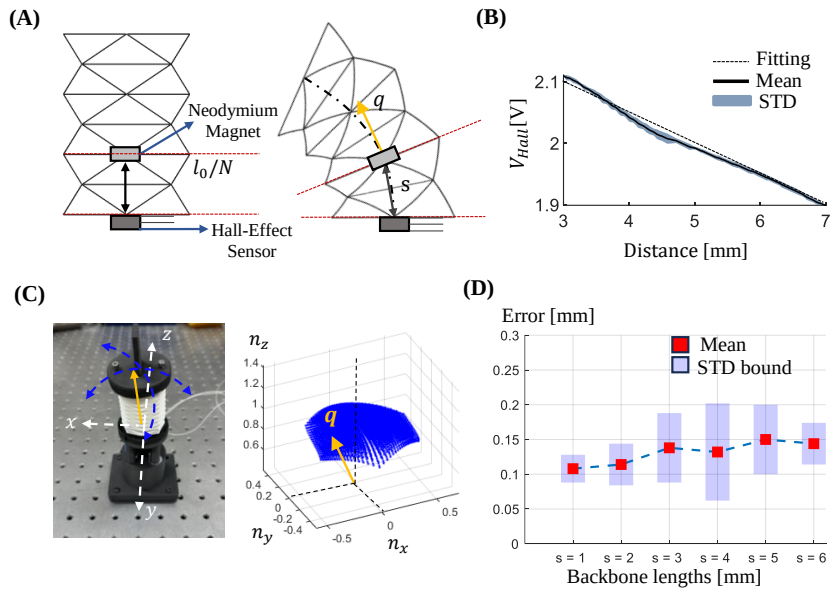


Figure C.1: (A) Length measurements during both linear and bending deformations. The axial height of a single layer (l_0/N) is shown for the axial deformation, while the curved length of the neutral axis (s) represents the bending deformation. (B) Sensor signal output from the Hall effect sensor, mapped against the predefined displacement of the single layer. A linear fitting function was derived based on the calibration results. (C) The bending test setup for the OCM, measuring the Hall-effect sensor signal within the ROM and checking the tilted configuration of the magnet relative to the sensor. (D) Error analysis of the Hall effect sensor length measurements across various orientations at each backbone length.

Appendix D.

Action Space Mapping Between Real and Simulation

Two sources of discrepancy between the pressure input levels in the real robot and the SOFA FEM simulation were addressed using two MLP-based mapping functions: $f_{\text{MLP,RS}}$ and $f_{\text{MLP,SP}}$. The objective of both mappings was to align the configuration state $\mathbf{X}$ across different domains by estimating the calibrated actuation inputs. Two mapping functions were defined as below.

- f_{RS} maps the real pressure input $P_R \in \mathbb{R}^3$ to the SOFA-calibrated pressure $P_S \in \mathbb{R}^3$, so that the resulting TCP position X_S in simulation matches the real robot TCP position X_R.
- f_{SP} maps the FEM simulation pressure $P_S \in \mathbb{R}^3$ to the joint angles $\theta_P \in \mathbb{R}^{n_a}$ in the surrogate model, where n_a is the number of controllable DOFs.

In both cases, the goal is to minimize the RMSE between matched state variables:

$$\text{RMSE}(\mathbf{X}_{\text{ref}}, \widehat{\mathbf{X}}) = \sqrt{\frac{1}{N_n} \sum_{i=1}^{N_n} \left\| \mathbf{X}_{\text{ref}}^i - \widehat{\mathbf{X}}^i \right\|^2},$$

where N_n is the number of samples and $\mathbf{X}_{\text{ref}}$, $\widehat{\mathbf{X}}$ are the target and predicted configuration states, respectively.

Both f_{RS} and f_{SP} shared the same network design and training procedures:

- **Architecture:** 3-layer fully connected MLP with [64, 64, 64] neurons
- **Activation Function:** ReLU for hidden layers, linear for output

- **Input/Output:**
  - $f_{\text{RS}} : P_R \mapsto P_S$
  - $f_{\text{SP}} : P_S \mapsto \theta_P$
- **Loss Function:** Mean squared error (MSE) on configuration states
- **Optimizer:** Adam
- **Learning Rate:** 1×10^{-3}
- **Batch Size:** 32
- **Epochs:** 200, with early stopping after 20 stagnant epochs
- **Framework:** PyTorch, single NVIDIA RTX 4070 12G

The dataset for f_{RS} was obtained by applying static pressure combinations to both the real robot and the SOFA model, using $\tau = 0$. The output $\mathbf{X}_R$ and $\mathbf{X}_S$ were used to generate training targets. For f_{SP}, the input was the pressure from the FEM domain and the target was the optimized joint angle vector θ_P, derived via L-BFGS-B optimization.

Once trained, the networks were used in two ways. The function f_{RS} was used to infer calibrated FEM pressure inputs P_S that produce configuration states that match the real robot given a real pressure P_R. The other f_{SP} enabled the control of the surrogate model using the joint angle predictions learned from the pressure-based input.

Appendix E.

Model Order Reduction by Proper Orthogonal Decomposition

The model order reduction (MOR) using proper orthogonal decomposition (POD) was applied to accelerate soft body simulation while retaining dominant deformation behavior [41,44]. The vector of full-order nodal displacement in each time step is denoted:

$$\mathbf{p}(t) \in \mathbb{R}^{3N},$$

where N is the number of nodes and each entry contains the displacements x, y, z. The initial number of nodes N was 4327. A snapshot matrix $Sn \in \mathbb{R}^{3N\times M}$ was formed by stacking time-series nodal states:

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

These snapshots were obtained from SOFA simulations with randomized pressure inputs and external forces to sufficiently explore the configuration space, which was applied M = 20K samples in singular value decomposition (SVD):

$$Sn = U\Sigma V^T,$$

where:

- $U \in \mathbb{R}^{3N\times 3N}$: spatial POD modes,
- $\Sigma \in \mathbb{R}^{3N\times M}$: singular values,
- $V^T \in \mathbb{R}^{M\times M}$: temporal coefficients.

On a workstation with an AMD Ryzen 9 processor and 128 GB RAM, full singular value decomposition of the matrix $Sn \in \mathbb{R}^{12981\times 20000}$ was completed

in approximately 20 minutes using multithreaded CPU execution. The first $r = 1223$ dominant modes were retained to construct the reduced basis:

$$\Phi = U_r \in \mathbb{R}^{3N\times 1223}, \quad \mathbf{p}(t) \approx \Phi\, \mathbf{a}(t),$$

where $\mathbf{a}(t) \in \mathbb{R}^{1223}$ represents the generalized coordinates in the reduced space.

The full-order dynamics of the deformable manipulator are governed by the equation

$$\mathbb{M}(\mathbf{p})\, \ddot{\mathbf{p}} = \mathbb{P}(t) - \mathbb{F}(\mathbf{p}, \dot{\mathbf{p}}) - \mathbb{H}^T \boldsymbol{\lambda}(t),$$

where $\mathbf{p}(t) \in \mathbb{R}^{3N}$ is the nodal displacement vector, and $\mathbb{M}$, $\mathbb{P}$, $\mathbb{F}$, $\mathbb{H}$, and $\boldsymbol{\lambda}$ denote the mass matrix, external pressure-induced forces, internal elastic and damping forces, constraint Jacobian, and Lagrange multipliers, respectively.

To reduce the system, the nodal displacement vector was approximated using a linear projection onto a reduced basis: $\mathbf{p}(t) \approx \Phi\mathbf{a}(t)$, where $\Phi \in \mathbb{R}^{3N\times 1223}$ is the POD basis containing the first $r = 1223$ dominant spatial modes obtained via singular value decomposition, and $\mathbf{a}(t) \in \mathbb{R}^{1223}$ are the generalized coordinates. Substituting this into the full-order dynamics and applying Galerkin projection yields the reduced-order system

$$\mathbb{M}_r \ddot{\mathbf{a}} = \mathbb{P}_r(t) - \mathbb{F}_r(\mathbf{a}, \dot{\mathbf{a}}) - \mathbb{H}_r^T \boldsymbol{\lambda}_r(t),$$

where the reduced matrices are defined as

$$\begin{aligned} \mathbb{M}_r &= \Phi^T \mathbb{M}(\mathbf{p})\Phi, \\ \mathbb{P}_r(t) &= \Phi^T \mathbb{P}(t), \\ \mathbb{F}_r(\mathbf{a}, \dot{\mathbf{a}}) &= \Phi^T \mathbb{F}(\Phi\mathbf{a}, \Phi\dot{\mathbf{a}}), \\ \mathbb{H}_r &= \Phi^T \mathbb{H}, \quad \boldsymbol{\lambda}_r = \boldsymbol{\lambda}. \end{aligned}$$

The mass matrix $\mathbb{M}(\mathbf{p})$ was precomputed using the SOFA FEM engine, assuming a constant material density of $1040\,\mathrm{kg/m^3}$. The external force vector $\mathbb{P}(t)$ was generated using SOFA's `SurfacePressureConstraint` applied to

three surface cavities. The internal force term $\mathbb{F}(\mathbf{p}, \dot{\mathbf{p}})$ includes both elastic and damping components, computed using a corotational FEM model with Rayleigh damping coefficients $\alpha = 0.05$ and $\beta = 0.005$. The constraint Jacobian $\mathbb{H}$ captures the influence of fixed supports and contact constraints, and was extracted from the simulation using `FixedConstraint` and contact handling modules. The Lagrange multipliers $\boldsymbol{\lambda}(t)$ were solved internally by SOFA's constraint solver and retained during projection. The reduced system captures the dominant dynamics of the manipulator with significantly lower computational cost.

The reduced coordinates $\mathbf{a}(t)$ were further used to construct a compact state vector $\mathbf{X}_S(t) \in \mathbb{R}^{N_s}$, containing physically meaningful quantities required for learning and control. This vector included the Cartesian position of the tool center point $\mathbf{X}_{\text{TCP}}(t)$, the curvature $\kappa(t)$, the arc length $l(t)$, and the orientation matrix $\mathbf{R}(t)$ of the end-effector. These reduced-order representations served as input features for inverse models, forward dynamics networks, and reinforcement learning policies.

Appendix F.

Grouped Joint Constraints

To ensure that the surrogate model exhibits physically realistic deformation behavior, joint angle constraints were applied to adjacent joint groups. The action space of the surrogate model includes multiple rotational and prismatic joints, denoted by $\theta_{x_i}, \theta_{y_i}, \theta_{d_i}$, corresponding to each segment i. To prevent abrupt bending or unrealistic shape transitions, the following inequality constraints were enforced as

$$\begin{aligned} |\theta_{x_i} - \theta_{x_{i+1}}| &< \delta_x, \\ |\theta_{y_i} - \theta_{y_{i+1}}| &< \delta_y, \text{ and} \\ |\theta_{d_i} - \theta_{d_{i+1}}| &< \delta_d, \end{aligned} \tag{F.1}$$

where $\delta_x = 0.01$, $\delta_y = 0.01$, and $\delta_d = 0.005$ are constants that limit inter-joint variations. These bounds were chosen to reflect the smooth, continuous curvature observed in the real robot under actuation and external loads. The absolute joint limits were also constrained to reflect the physical range of motion:

$$\theta_x, \theta_y \in [-0.2, 0.2] \quad \text{and} \quad \theta_d \in [0, 0.02]. \tag{F.2}$$

The constraints in Equation F.1 were applied within the control environment and enforced during optimization and policy rollout. The lower and upper bounds were embedded into the control step function and the optimizer (Algorithm 1) to restrict the exploration space to feasible regions. In addition, constraints enforce spatial smoothness across joint segments, consistent with the continuous curvature observed in physical continuum robots. In the absence of such constraints, the surrogate model may exhibit abrupt curvature reversals across layers, corresponding to high-frequency deformation modes that are not physically realizable due to material and pneumatic coupling. The constraint values $\delta_{\theta_X}, \delta_{\theta_Y}, \delta_d$ effectively limit the second-order variation

in curvature, promoting globally feasible shapes. Without such constraints, the surrogate model can exhibit unrealistic behavior, such as large curvature changes between links, as shown in Supplementary Figure F.1-(a). These configurations are not observed in the real system, even under dynamic loading. In contrast, applying the grouped constraints results in smooth and physically plausible postures, as shown in Supplementary Figure F.1-(b).

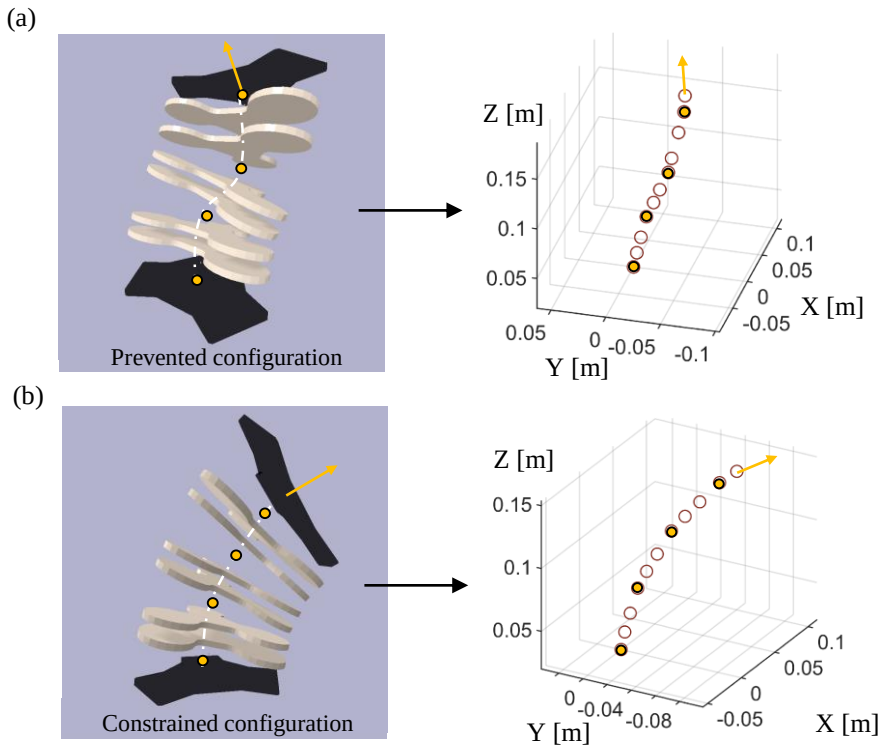


Figure F.1: (a) Example of an unrealistic configuration in the surrogate model, where multiple curvatures appear across joint segments, deviating from the physical behavior of the real robot. (b) Example of a feasible configuration generated by applying joint constraints, ensuring smooth deformation.

국문 초록

소프트 로보틱스 분야에서 학습 기반 제어의 발전은 연산 속도와 물리적 충실도 사이의 임계적 괴리인 시뮬레이션과 현실 사이의 격차에 의해 근본적으로 제약받고 있다. 본 학위논문은 기존 모델링 패러다임을 엄밀히 분석하여 이 격차를 수학적으로 정의하고, 데이터 기반 추상화의 당위성을 입증한다.

본 연구는 해석적 근사가 외력 하의 필수적인 비선형 거동을 포착하지 못하는 반면, 유한요소법과 같은 수치 해석법은 복잡도가 세제곱으로 증가하여 강화 학습에 요구되는 대규모 데이터 처리가 불가능하다는 점을 밝힌다. 이러한 불일치를 해결하기 위해, 본 논문은 물리적 참값 생성 과정과 런타임 실행을 분리하는 통합 파이프라인을 제안한다.

일련의 연구를 통해 다음 두 가지 핵심 추상화 전략에 대한 이론적 토대를 확립하였다. 첫째, 대체 모델링 (Surrogate modeling)을 통한 모델 차수 축소를 통해 거시적 물체 역학을 고속 강체 호환 형식으로 압축한다. 둘째로 신경망 물리 엔진(Neural physics engines)을 이용하여 데이터 기반 접촉 모델 근사를 활용하여 접촉 계면의 복잡하고 고차원적인 역학을 포착한다.

이 방법론은 강건한 시뮬레이션 전이학습에 필수적인 연속체 물리학적 특성을 유지하면서도 강체 엔진 수준의 연산 효율성을 달성하여 현실 격차를 해소하는 것을 목표로 한다. 궁극적으로 본 연구는 고차원 유한요소 데이터를 저차원의 필수정보로 추상화하는 일반화된 파이프라인을 제시하며, 이를 로봇 공학 내 변형이 수반되는 주요 시스템에 적용함으로써 확장 가능하고 물리적으로 충실한 학습을 구현한다.