Kinematics Computing for Soft Robots

Method based on Geometric Computing and Machine Learning

Fang, G.

DOI
10.4233/uuid:f8b714b2-621a-470e-86d9-ee728999d624

Publication date
2022

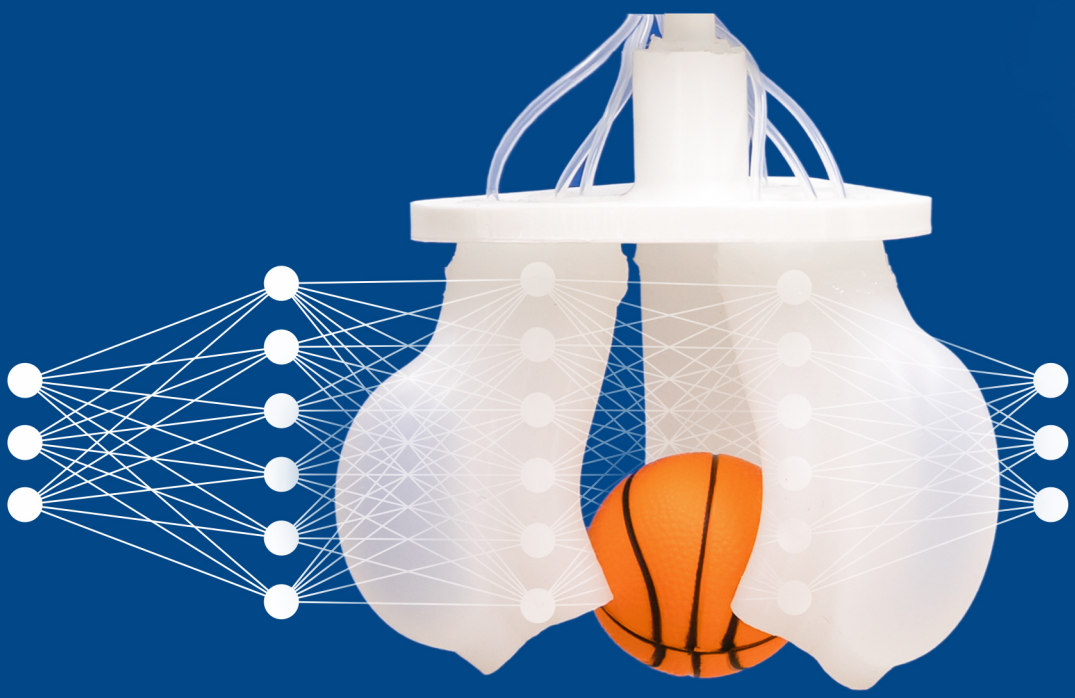
Document Version
Final published version

Citation (APA)
Fang, G. (2022). *Kinematics Computing for Soft Robots: Method based on Geometric Computing and Machine Learning*. [Dissertation (TU Delft), Industrial Design Engineering].
https://doi.org/10.4233/uuid:f8b714b2-621a-470e-86d9-ee728999d624

Kinematics Computing for Soft Robots

Method Based on Geometric Computing and Machine Learning

Guoxin Fang



Kinematics Computing for Soft Robots

Method based on Geometric Computing and Machine Learning

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on
Monday 28 November 2022 at 15:00 o'clock

by

Guoxin FANG

Bachelor of Engineering in Mechanical Engineering and Automation,
Beijing Institute of Technology, China,
born in Jingdezhen, China.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	Chairperson
Prof. dr. C. C.L. Wang	Delft University of Technology, promotor
Prof. dr. ir. J. M.P. Geraedts	Delft University of Technology, promotor
Independent members:	
Prof. dr. ir. K. M.B. Jansen	Delft University of Technology
Prof. dr. E. Eisemann	Delft University of Technology
Prof. dr. Y. Sun	University of Toronto, Canada
Prof. dr. A. Weightman	The University of Manchester, United Kingdom
Dr. Y. Song	Delft University of Technology
Prof. dr. P. Vink	Delft University of Technology, reserve member

This research was partially supported by the China Scholarship Council (CSC).

Keywords: Soft Robot, Kinematics, Geometric Computing, Machine Learning

Printed by: Proefschriftspecialist, the Netherlands

Front & Back: Designed by Guoxin Fang.

ISBN 978-94-6366-634-3

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Table of Contents

Summary

Soft robots that are built from materials with mechanical properties similar to those of living tissues can achieve tasks like never before in comparison to conventional rigid robots. Powered by the compliance of soft materials and novel structure designs, complex motion (e.g., bending, twisting, and extension) can be accomplished in robotic bodies. We now see soft robots being used to grasp fragile objects and detect confined areas. However, conventional modeling and control approaches, which rely on the rigidity of the robot body, are less effective when directly applied to soft robotic systems. Therefore, new methods and algorithms need to be developed that allow modeling and kinematics control for soft robots.

In model-based robot control, kinematics comprise the fundamental knowledge that can be used to build the mathematical connection between control parameters and robot status. Unlike rigid robots, whose kinematics are well studied and have fast (analytical) solutions, effective and general kinematics computing methods for soft robot systems are still lacking. According to the modeling perspective (i.e., *forward kinematics* (FK)), predicting the whole-body shape of soft robots under actuation is a non-trivial task since the non-linear deformation in robot bodies and the hyperplastic properties of soft materials create challenges in balancing accuracy and computational costs in existing FK models. The lack of modeling tools further brings the difficulties in developing advanced algorithms to *inverse kinematics* (IK) and (statics) control thereafter. This Ph.D. project aims to develop a general soft robot kinematics computing pipeline, that can contribute to the effective control of soft robot systems to accomplish given tasks.

A fast numerical simulator for soft robots is firstly presented in this thesis, in which the shape of the robot body is discretely represented by volumetric elements. The development of this simulator was inspired by the fact that the hard-to-model actuation input (e.g., cable force, pressure, and electronic field) in soft robot systems can be directly modeled or transformed to fit the shape change in actuation elements. An optimization pipeline was built to minimize elastic energy in the body elements and compute the deformed shape with actuation parameters as input. As a general numerical simulator, it supports the modeling of various types of actuation, and the hyperelastic soft material properties are integrated. A fast collision checking and response model was added to predict the behavior of soft robots under robot-robot collisions and robot-environment interactions. The numerical computing process of our simulator

shows good convergence, even for soft robots with large (rotational) deformation in their bodies, and can therefore balance the computational cost and model precision. In comparison to commercial *finite element analysis* (FEA) software, this geometry-based simulator demonstrates a 20-fold faster computing speed, and the simulation result can well fit the shape that was captured from the physical setup.

The IK problem of soft robots is defined as computing proper actuation parameters that drive soft robots to accomplish given tasks. In this thesis, task-specific IK objectives (which are mainly geometrically defined) are formulated, and the optimal actuation parameters are detected using gradient-based iteration. Through the developed simulator, the gradients of objective functions are estimated using numerical differences. The sequence of motion can be successfully computed using this IK solver, and its efficiency has been verified in two case studies, which include path-following and object pick-and-place.

For the final stage of this Ph.D. project, the speed and precision of the IK solver are enhanced through machine learning. Fully connected neural networks are invited to fit functions of FK and the Jacobian of IK-related objectives. With the high efficiency in the forward propagation of networks (in analytical form), the gradient-based IK solver can run in real-time. Sim-to-real transfer learning is applied to eliminate the reality gap and make the computed actuation parameters more precise in physical setups. Applying sim-to-real transfer learning can also benefit the efficiency of the data generation process. In our pipeline, massive training data is first generated in a virtual environment using a fast simulator; thereafter, a lightweight network layer is employed to map the result of the simulation to the physical hardware. As a result, the amount of physical data can be reduced by 60% to train a network that accurately computes IK solutions.

In conclusion, this dissertation presents a pipeline that computes kinematics solutions for soft robots. A fast geometry-based simulator is presented to contribute to building an iteration-based numerical IK solver. Machine learning is applied to accelerate IK computing to real-time speed with enhanced precision. Task-specific kinematics control is realized in different soft robot systems to verify the effectiveness of the proposed method. The algorithms and code presented in this Ph.D. thesis are open-sourced for researchers and designers, and have the potential to become a general tool for designing and controlling soft robots. Future studies on the design optimization and high-level control of soft robots can all benefit from the research outcomes of this project.

Samenvatting

Zachte robots, die zijn gemaakt van materialen met mechanische eigenschappen die vergelijkbaar zijn met levend weefsel, kunnen hele nieuwe taken uitvoeren in vergelijking met conventionele starre robots. Door het vermogen van zachte materialen om zich aan te passen aan externe vormen en bijbehorende nieuwe ontwerpen voor hun structuur, kunnen complexe bewegingen (bijvoorbeeld., buigen, draaien en uitzetting) uitgevoerd worden in robotlichamen. We zien nu dat zachte robots worden gebruikt om kwetsbare objecten vast te pakken en besloten ruimtes te detecteren. Conventionele modellerings- en besturingsbenaderingen, die gebaseerd zijn op de stijfheid van een robotlichaam, zijn minder effectief wanneer ze rechtstreeks worden toegepast op zachte robots. Daarom moeten nieuwe methoden en algoritmen worden ontwikkeld die modellering en kineastische controle van zachte robots mogelijk maken.

Bij de modelgebaseerde robotbesturing omvat kinematica de fundamentele kennis die de wiskundige verbinding legt tussen besturingsparameters en robotstatus. In tegenstelling tot starre robots, waarvan de kinematica goed is bestudeerd en snelle (analytische) oplossingen zijn gerealiseerd, ontbreken nog algemene en effectieve rekenmethoden voor de kinematica voor zachte robotsystemen. Vanuit het modellering oogpunt, gebaseerd op voorwaartse kinematica (FK), is het voorspellen van de vorm van het hele lichaam van aangedreven zachte robots geen triviale taak. De niet-lineaire vervormingen in zachte robotlichamen en de hyperplastische eigenschappen van de gebruikte zachte materialen zorgen voor uitdaging bij het in evenwicht brengen van de bewegingsnauwkeurigheid en de rekenkosten in bestaande voorwaartse kinematica modellen. Het huidige gebrek aan modellering hulpmiddelen geeft ook problemen bij het ontwikkelen van geavanceerde algoritmen voor inverse kinematica (IK) en de (statische) controle daarvan. Dit promotieonderzoek heeft tot doel een algemeen bruikbare pijplijn voor het berekenen van de kinematica voor zachte robots te ontwikkelen, die kan bijdragen aan een effectieve controle om specifieke taken uit te voeren.

In dit proefschrift wordt eerst een snelle numerieke simulator voor zachte robots gepresenteerd, waarbij de vorm van het robotlichaam in afzonderlijke volume elementen wordt opgedeeld. De ontwikkeling van deze simulator is geïnspireerd op het feit dat de moeilijk te modelleren aandrijvingen (bijvoorbeeld kabelkracht, druk of elektrisch veld) in zachte robotlichamen direct kan worden gemodelleerd of getransformeerd naar de vormverandering

in een volume element. Er is een zelflerende pijplijn gebouwd om de elastische energie in de volume elementen van het soft robotlichaam te minimaliseren en de vervormde vorm te berekenen met de aandrijf parameters als invoer. De ontwikkelde algemene numerieke simulator ondersteunt de modellering van de verschillende soorten aandrijving en ook zijn de hyperelastische eigenschappen van de gebruikte zachte materialen geïntegreerd. Tevens is een snel controle- en responsmodel voor botsingen toegevoegd om het gedrag bij botsingen tussen robots onderling en interacties met de omgeving, waarin de robots werken, te voorspellen.

Het numerieke rekenproces van de simulator vertoont een goede convergentie, zelfs voor zachte robotlichamen met grote (rotatie)vervorming, en brengt de bewegingsnauwkeurigheid van het model en de rekenkosten in evenwicht. De op geometrie gebaseerde simulator demonstreert, in vergelijking met de commerciële software voor eindige-elementenanalyse, een 20-maal hogere rekensnelheid, en het simulatieresultaat past goed bij de beweging die wordt gemeten in de fysieke opstelling.

Het IK-probleem van zachte robots wordt gedefinieerd als het berekenen van de juiste parameters voor de aandrijving die zachte robots ertoe aanzetten bepaalde taken uit te voeren. In dit proefschrift worden taak specifieke IK-doelen (die voornamelijk geometrisch zijn gedefinieerd) geformuleerd en worden de optimale parameters voor de aandrijving gedetecteerd met behulp van een op gradiënten gebaseerde iteratie. De ontwikkelde simulator schat de gradiënten van objectieve functies met behulp van numerieke verschillen. De bewegingsvolgorde wordt succesvol berekend met behulp van deze IK-oplosser, en de uiteindelijke efficiëntie is geverifieerd in twee praktijkstudies, gebaseerd op het volgen van bewegingspaden en het kiezen en plaatsen van objecten.

Als laatste fase van dit PhD-project worden de snelheid en nauwkeurigheid van de IK-oplosser verbeterd door middel van zelflerende algoritmen (machine learning). Volledig verbonden neurale netwerken worden gebruikt om functies van FK te fitten met de Jacobiaan van IK-gerelateerde doelen. Door de hoge efficiëntie in de voorwaartse voortplanting, van invoerlaag naar uitvoerlaag, in het neurale netwerk (in analytische vorm), kan de op gradiënten gebaseerde IK-oplosser in realtime worden uitgevoerd. Door toepassing van een leerproces, om de simulatie naar de realiteit te kunnen vertalen, worden de berekende parameters voor de aandrijving, die gebruikt worden in fysieke opstelling, nauwkeuriger. Dit leerproces komt ook de efficiëntie van de generatie van data ten goede. In de zelflerende pijplijn worden eerst in een virtuele omgeving grote hoeveelheden trainingsgegevens gegenereerd door een snelle simulator;

daarna wordt een lichtgewicht netwerklaag gebruikt om het resultaat van de simulatie op de fysieke hardware af te beelden. Met als gevolg dat de noodzakelijke hoeveelheid fysieke gegevens met 60% kan worden verminderd om een netwerk te trainen dat nauwkeurige IK-oplossingen berekent.

Tot slot, dit proefschrift presenteert een pijplijn die kinematica-oplossingen voor zachte robots berekent. Een snelle, op geometrie gebaseerde simulator wordt gepresenteerd om bij te dragen aan het bouwen van een op iteratie gebaseerde numerieke IK-oplosser. Zelflerende algoritmen worden toegepast om IK-berekeningen te versnellen tot een realtime niveau met verbeterde nauwkeurigheid. Met diverse zachte robots is de taakspecifieke controle door kinematica gerealiseerd om zo de effectiviteit van de voorgestelde methode te verifiëren. De algoritmen en code gepresenteerd in dit proefschrift zijn open source voor onderzoekers en ontwerpers en hebben het potentieel om een algemeen hulpmiddel te worden voor het ontwerpen en besturen van zachte robots. Toekomstige studies naar de optimalisatie van het ontwerp en de realisatie van een kwaliteit aansturing voor zachte robots kunnen zo profiteren van de onderzoeksresultaten van dit project.

Glossary

Abbreviations

3D	Three Dimensional
AM	Additive Manufacturing
AABB	Axis-Aligned Bounding Box
BVH	Bounding-Volume Hierarchies
DOF(s)	Degree(s) Of Freedom
DEA	Dielectric Elastomer Actuators
FK	Forward Kinematics
FNN	Feedforward Neural Network
FE	Finite Element
FEA	Finite Element Analysis
FDM	Fused Deposition Modeling
IK	Inverse Kinematics
LSTM	Long Short-Term Memory
TPU	Thermoplastic Polyurethane

Definitions

3D Printing See *Additive Manufacturing*

Additive Manufacturing (AM) A manufacturing method that creates three dimensional objects by depositing materials according to the order created by digital files, usually in layers.

Collision Checking Detecting the intersection of two or more objects.

Collision Response Models and algorithms that predict the behavior of two objects following a collision and other forms of contact.

Degrees of Freedom (DOFs) The number of independent aspects of motion that determine the configuration of a mechanical system.

Feedforward Neural Network A machine learning algorithm that consists of an artificial neural network with no feedback connections.

Geometric Modeling A branch of methods and algorithms that study the mathematical description of shapes.

Hyperelastic Material A type of constitutive material that can have extremely large elastic deformation. Meanwhile, the stress–strain relationship is nonlinear and derives from a strain energy density function.

Kinematics The relationship between the dimensions and connectivity of kinematic chains and the position, velocity, and acceleration of each of the links in the robotic system.

Long Short-Term Memory A machine learning algorithm that consists of an artificial recurrent neural network structure with feedback connections.

Machine Learning A type of artificial intelligence (AI) that builds algorithms to build models based on training data to make predictions without explicitly programming.

Open-Loop Control A control system in which feedback is not detected or the input command or control process is not influenced regardless of whether the output has achieved the desired goal.

Robot A machine that can automatically replicate certain movements and functions.

Configuration Space A complete specification of the position of every point in a robotics system.

Task Space The position and orientation of the end effector of a robot.

Soft Robot A robot that is primarily composed of materials with moduli in the range of the moduli of soft biological materials (Young's moduli between 10^4 and 10^9 Pa)

Under-Actuated System A system in which the number of actuators is less than the number of degrees of freedom of the system

1 Introduction

Soft robots have demonstrated great potential in achieving tasks that cannot be achieved by conventional rigid robots. Therefore, they broaden the applications of robotics and bring much attention to the research field. As the main focus of this PhD research, this chapter first provides a brief introduction to soft robots from the perspectives of design, actuation, and application. The definition of operation spaces for soft robots is then presented, which is followed by the definition of forward and inverse kinematics (FK and IK) computing. As essential functions for the modeling and control of soft robots, the research objective of this PhD dissertation is developing a general computational pipeline to solve the kinematics of soft robots at a fast speed with high precision. The challenges to reaching this goal are presented in four research questions which are explained in detail and solved using corresponding research cycles. This chapter ends with the organization of this dissertation.

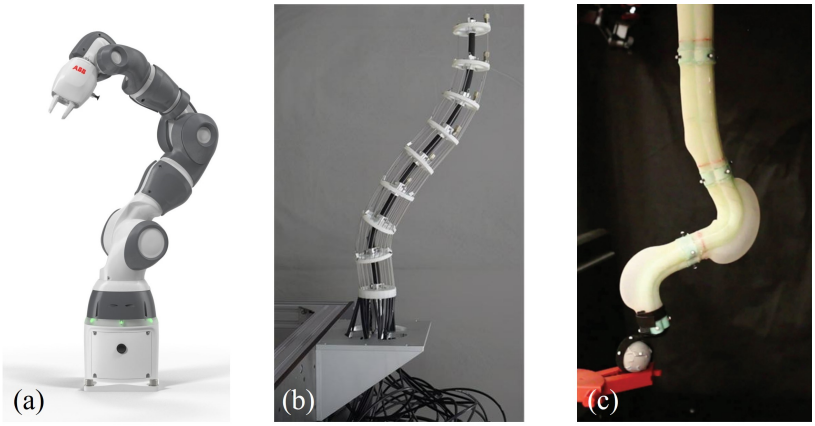


Figure 1.1: Evolution from (a) rigid-link robot arm (Image credit: ABB Inc.) to (b) continuum manipulator [1] and (c) soft robot arm [2].

1.1 Introduction of Soft Robots

In the past decades, engineers and designers have worked together to develop advanced industrial robot systems to execute specific functions and operations [3]. Constructed with rigid links and driven by motors, rigid robot manipulators (as illustrated in Fig. 1.1(a)) generally operate with high precision and can work in a well-structured working environment to repeat manually guided motions [4]. Those properties allow them to be deployed in processing lines to conduct highly repetitive operations, even in dirty and dangerous environments. For manufacturing and construction industries, robots now play an important role in replacing manual work and supporting automations [5]. However, those conventional robots demonstrate less compatibility when placed in agricultural and horticultural industries where operation spaces can be unstructured and full of (dynamic) obstacles [6]. For example, making robots that can effectively harvest fruit like human hands to support sustainability in agriculture is challenging. When grasping fruit, rigid manipulators with low *Degrees-of-Freedom* (DoFs) can easily collapse other branches, and the rigid gripper can damage the target fragile objects. Due to the rigidness exhibited by robotic bodies, conventional robots have a limited ability to work with humans in industry settings. Although external sensor/vision systems and advanced control algorithms have been developed to allow force-feedback control and a certain level of human-robot interaction, a smart co-robot working principle with guaranteed collision-free motions has not yet been achieved in the robot industry.

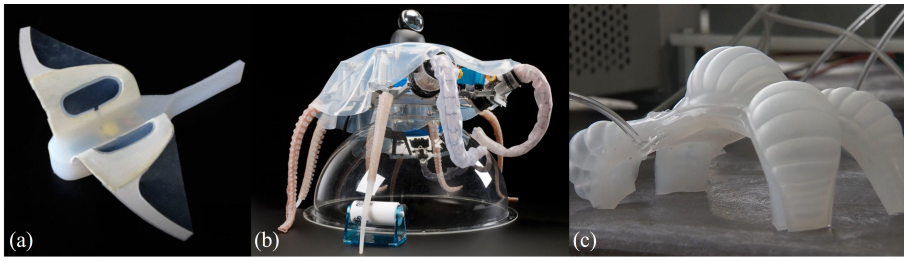


Figure 1.2: Bio-inspired soft robots designs. (a) Soft robot fish [10]. (b) Soft robot octopus [11]. (c) Soft legged robot [12]

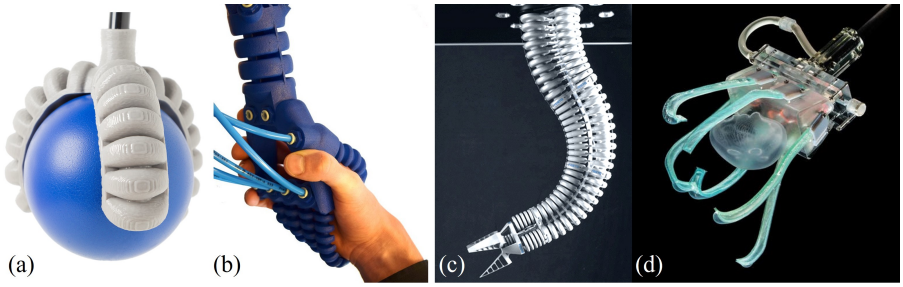


Figure 1.3: Soft robot manipulators that were developed for industry applications.

Since the early '80s, robotic engineers have incorporated compliance into the structure of robots and developed continuum robots [7]. As illustrated in Fig. 1.1(b), these robots contain a set of rigid blocks that can bend like an elephant's trunk [1]. In the automobile industry, Tesla has applied continuum robots as part of the automatic charging system [8]. Nevertheless, those manipulators are far from satisfying since they are still constructed by a large group of rigid components and have difficulty with safely interacting with environments and humans. Since the 2000s, soft robots (e.g., Fig. 1.1(c)) have been designed with maximized DoFs and have been used to explore full compliance in robot bodies. According to Rus and Tolley [9], soft robots are fabricated with materials that fit Young's modulus with less than 10^9 MPa. The usage of soft material means that such robots can absorb energy when collisions happen and, therefore, become harmless when interacting with humans.

Soft robots can achieve complex deformations including expanding, bending, and twisting with the help of novel structure designs [16, 17, 18, 19]. The designs of soft robots from an earlier stage were bio-inspired, which was natural since muscle-like robot bodies can be actuated to mimic the behavior of living creatures. Fig. 1.2(a) and Fig. 1.2(b) present the design of a soft robot

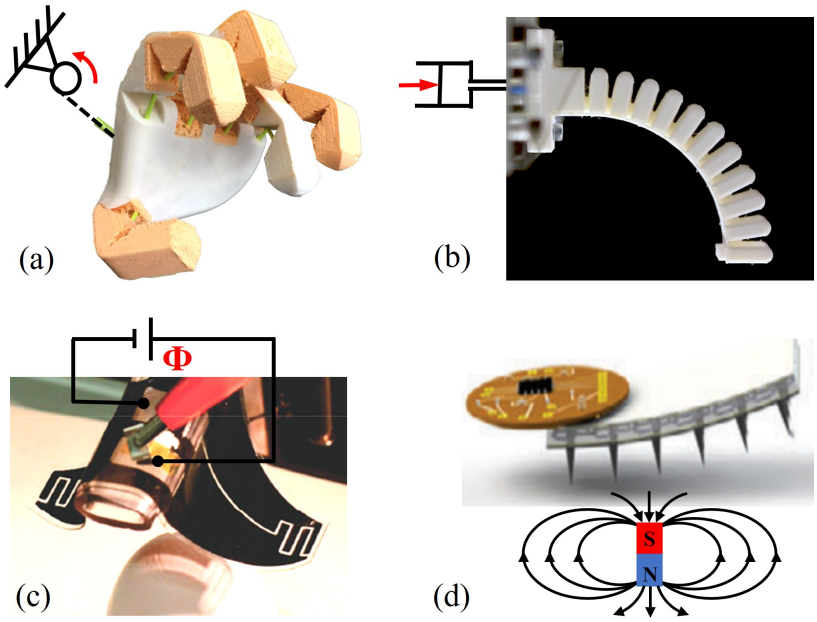


Figure 1.4: Soft robot systems with different actuation types. (a) A soft robot hand that is driven by cable force [13]. (b) A soft finger that is driven by pneumatic actuation with pressure [13]. (c) The design of a dielectric elastomer-driven soft gripper [14]. (d) A soft warping robot that is driven by magnetic fields [15].

fish and octopus [10, 11]. These robots were fabricated from silicone rubber and can explore the underwater environment. Soft-legged robots can perform locomotion in obstacle-dense environments like worms [12] (this is illustrated Fig. 1.2(c)). Reinforced by soft muscles that can be driven rapidly by electrical signals, soft robots can even mimic the motion of cheetahs [20]. However, soft manipulators have been designed for and used in the food industry for picking-and-placing fragile objects (e.g., the pneumatic-driven soft gripper and manipulators [21, 22] illustrated in Fig. 1.3(a) and Fig. 1.3(c)). For a similar application in the fishery industry, soft grippers that were fabricated with extremely soft materials have been used to cage and capture jellyfish and sea urchins [23] (illustrated in Fig. 1.3(d)). Members of the Industrial Design Engineering Faculty at TU Delft have developed a pipeline to integrate soft robot actuators, sensors, and regular rigid components to develop products that display desired behaviors [24]. They demonstrated a soft robotic prototype that can mimic the behavior of humans by engaging in behaviors such as hand

shaking (see Fig. 1.3(b)).

Soft robots can be driven through different types of actuation systems, where most of the types are provided through external power sources [25]. Fig. 1.4 presents four types of actuation: cable force, pressure, electronic voltage, and magnetic fields. Since the actuation system is separate from the robot body, soft robots can be lightweight and achieve high power-to-weight ratios [26]. Artificial exoskeletons that are powered by soft robot actuators [27, 28] have been developed by taking advantage of these properties. The usage of off-bodied actuation equipment allows soft actuators themselves to become free of metal and electronic components. Therefore, they could operate in humid or electromagnetic shielding environments. As an example, soft robotic manipulators are now being applied in intraoperative MRI-guided micro-surgeries to detect organ lesions and cut tumors [29, 30].

With numerous advantages in comparison to traditional rigid robots and as a newly developed robot family, soft robots have tremendously broadened the application of robotic systems. Studies have been conducted to identify smart soft materials [31], generate novel structure designs [32], and develop advanced fabrication processes [33]. Another key challenge is modeling and control. In this direction, existing approaches in the robotics research field assume classical levels of rigidity in the structure and, therefore, are less effective for soft robots [9]. The lack of effective modeling tools and control algorithms for soft robots not only creates challenges for industrial designers to develop novel soft robot systems but also lowers their efficiency when applied in industrial applications. To tackle the challenges of model and control, this Ph.D. dissertation focuses on solving the fundamental problem for kinematics computing of soft robots. The details of this problem are presented in the next section.

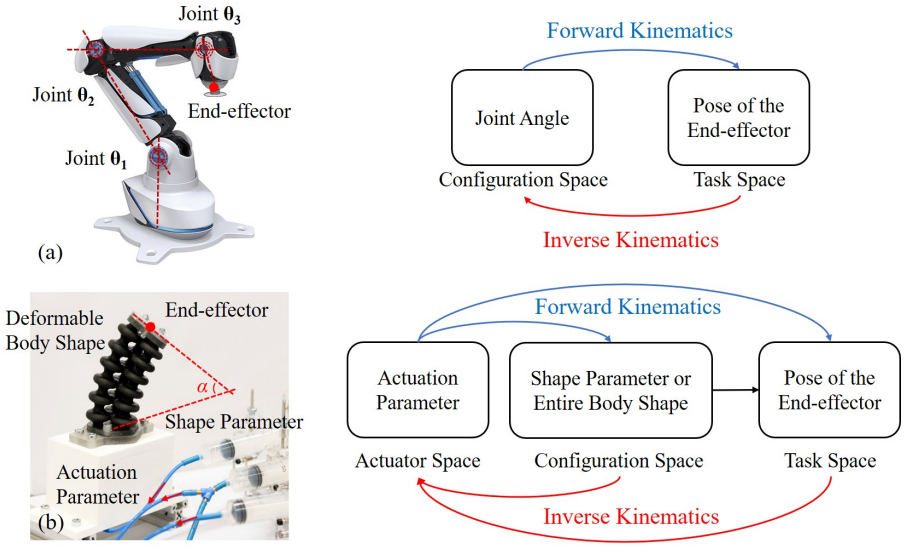


Figure 1.5: An illustration of the operation spaces for (a) conventional rigid-bodied robot arm and (b) soft robot manipulator.

1.2 Kinematics for Soft Robots

Soft robots can change their body shape with high degrees of flexibility and complexity. It is essential to develop smart algorithms in modeling and motion planning to maximize their performance and support their usage in real-world applications. Regarding the problem of robotics control, kinematics is a fundamental function that describes the connections between operating spaces. As presented in Fig. 1.5(a), the kinematics of conventional rigid-bodied robots build the mapping between robot joint angles and the pose of the end-effector. As discussed in the previous section, soft robots are driven by off-body actuation and their body configuration can contain high/infinite DoFs. In this section, we first present the definition of operation spaces (illustrated in Fig. 1.5(b)) for soft robots and then formulate the forward and inverse kinematics problems.

Actuator Space: Constructed by actuation parameters (e.g., cable force, pressure, voltage, etc.) as the main control space for soft robots. The dimension of actuator space is defined by the number of actuation parameters (e.g., chamber number for pneumatic-driven soft robots).

Configuration Space: The configuration space for soft robots refers to the

shape of the robotic body under deformation, which can be described discretely by *finite element*(FE) systems or locally defined shape parameters (e.g., the arc angle as illustrated in Fig. 1.5(b)). In this thesis, we focus on using FE systems to describe the configuration space since many soft robot applications require control of the whole body shape of soft robots. One should notice that the dimension of the configuration space can be large or even infinite.

Task Space: The task space is defined as the status of the regions of interest (ROI) that can be detected directly using the robot's configuration. In general, it contains the position/orientation information of the end-effector. In this thesis, we define a soft robot as a redundant system when the dimension of the task space is less than the actuator space.

Unlike rigid robots in which kinematics computing is comprehensively studied, soft robots still lack a general and effective pipeline to support the computing of kinematics. The *forward kinematics* (FK) of soft robots refer to predicting the shape change of the robot body in *configuration space* when actuation is applied. Building an efficient and precise FK model supports the computing of *inverse kinematics* (IK) solutions, as it allows the usage of a numerical solver with Jacobian-based iteration. When the control task focuses on the end-effector, the IK problem is defined as finding actuation parameters (or sequences) that drive the pose of the end-effector following the target motion in *task space*. When achieving tasks such as grasping and locomotion, which require the consideration of a full-body shape, IK can also be performed to compute parameters in *actuator space* with a given target shape that is defined in *configuration space*.

To tackle the problem of soft robot kinematics computing, the research objective for this PhD project is developed as:

Develop a fast and effective kinematics computing pipeline for soft robots, which is essential in modeling the behavior of soft robots and achieving task-specific kinematics control in practical applications.

The challenges of achieving this research objective are discussed in detail and transformed into four research questions in the next section.

1.3 Research Questions

The objective of this Ph.D. project contains two tasks:

1) Developing a fast and effective modeling tool to predict the behavior of soft robots under actuation;

2) Computing proper actuation parameters (or sequences) for soft robots to accomplish given tasks.

The first task refers to computing the FK of soft robots. Fig. 1.6 provides a summary of existing methods[1]. Among all the methods, numerical models can directly predict the shape of soft robots by solving a *finite element* (FE) system. With carefully selected element types (e.g., tetrahedral or hexahedral) and mesh density, novel soft robot designs with complex body structures can be precisely modeled. Together with physically calibrated material properties, soft robot FK can be precisely computed by solving the stress-strain relationship that is represented by a linear system [34]. In comparison to analytical models (e.g., the *piecewise constant curvature* (PCC) model [35] illustrated in Fig. 1.6(a)), the numerical model is more precise and has high geniality to support different types of actuation. However, numerically modeling the behavior of soft robots can suffer from high computational costs. One major issue is the highly nonlinear geometry changes to soft robot bodies (particular large rotational deformation) that make the FE-system difficult to converge. As a result, very small time steps need to be applied. This restricts the usage of commercial FEA software (e.g., Abaqus and Ansys) in a real-time model-based control. Although researchers have developed reduced FE models (e.g., SOFA framework [36] shown in Fig. 1.6(d)) with improved computing speeds, these models can lose their precision when large deformations are applied to soft robots [13]. To balance the computing speed and model accuracy to numerically solve the forward kinematics of soft robots, the first research question is defined as:

RQ1: *How can a fast and accurate modeling tool be developed to numerically compute the behavior of soft robots under actuation (i.e., solve forward kinematics)?*

Since soft robots contain compliance in their bodies, collisions can regularly happen within robot bodies (i.e., self-collision) or with environments when deployed to accomplish certain tasks. Modeling the behavior of soft robots under collision is also an essential task that can be used to plan mo-

[1] A comprehensive review of soft robot modeling methods is presented in Chapter 2.

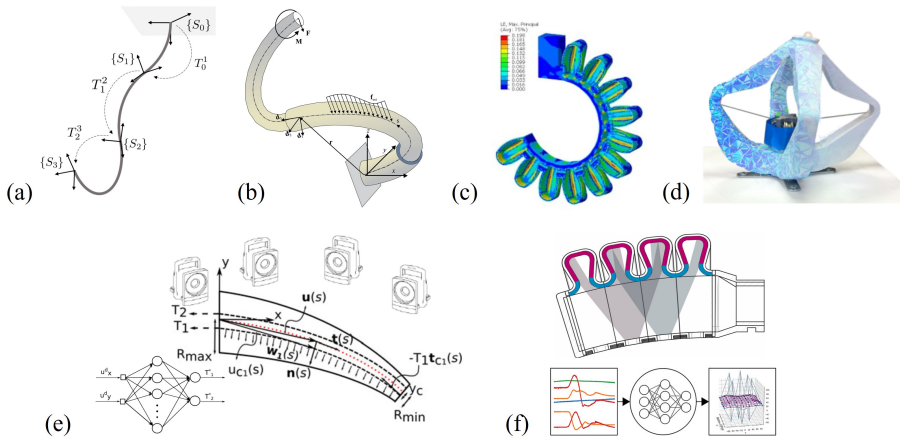


Figure 1.6: Methods for (forward) kinematics computing for soft robots. (a - d) Model-based methods where the kinematics of soft robot bodies are modeled analytically [35, 37] (a, b) or conducted using the numerical simulation [13, 38] (c, d). (e, f) Model-free methods in which machine learning technology is applied to fit the kinematics with data generated from the external vision or sensor system [39, 40].

tions in practical applications. Different from articulated robots with rigid bodies, the body shape of a soft robot significantly changes in a nonlinear way through actuation, which makes (self-)collision detection difficult to conduct efficiently. Meanwhile, it is challenging to provide a collision response model while being able to mimic the complex physical behavior of hyperelastic materials. Moreover, the collision checking and response model should be integrated with the computing process of FK and should not overload the computation speed. To tackle these challenges, the second research question is formulated as:

RQ2: *How can we model the behavior of soft robots when self-collision or environment interactions happen during actuation?*

This thesis presents the first approach to developing a fast and accurate numerical simulator that is based on geometric computing. This simulation-based FK model is conducted as a basic function to solve the IK problem. Existing analytical methods instantly solve IK only if the mapping from *task/configuration space* to *actuator space* can be analytically formulated [35]; however, this is not the case for our implementation. As FK are numerically calculated, only the iterative method could be applied [3]. The gradient-based iteration is applied to minimize task-specific kinematics objectives and to find IK solutions.

An effective solver needs to be built to enable the fast convergence speed of the optimization. Thus, the research question for soft robot IK computing is developed as:

RQ3: *How can we compute the proper control parameter of a soft robot to achieve the task-specific inverse kinematics (IK) control?*

The bottle-neck of the computational speed of an iteration-based IK solver is the costly process of evaluating objectives and their gradients since the FK function needs to be called for tremendous times. As another category of kinematics solutions, model-free methods can greatly accelerate kinematics computing speed [41, 42, 43]. Neural networks are used to fit the forward/inverse kinematics mapping. By replacing the costly numerical simulator with the analytically computed forward propagation of neural networks, a real-time computing speed for finding IK solutions can be achieved. Another advantage of learning-based methods is their high precision since the physical behavior of soft robots can be precisely captured [42, 44].

By using a data-driven method, most researchers generally collect training data from physical experiments using a vision setup [39] (illustrated in Fig. 1.6(e)) or sensor array [40] (illustrated in Fig. 1.6(f)). The effectiveness of learning-based solutions relies heavily on the quality of the training data, and the data set needs to cover the entire operation space. However, collecting massive quantities of training data from physical setups can be costly for soft robot systems because the materials used to fabricate soft robots can lose their functionality after repeating large deformations many times. To make the learning-based soft robot kinematics solution more applicable for practical applications, a method needs to be built to reduce the cost of generating training data in physical setups. To tackle this challenge, we developed the final research question regarding the solution of IK computing of soft robots by learning:

RQ4: *How can we build a machine learning pipeline to precisely solve the inverse kinematics (IK) of soft robots in real-time speed with an efficient data generation process?*

We answer these research questions by conducting related research cycles, which are explained in the next section with the structure of this Ph.D. thesis.

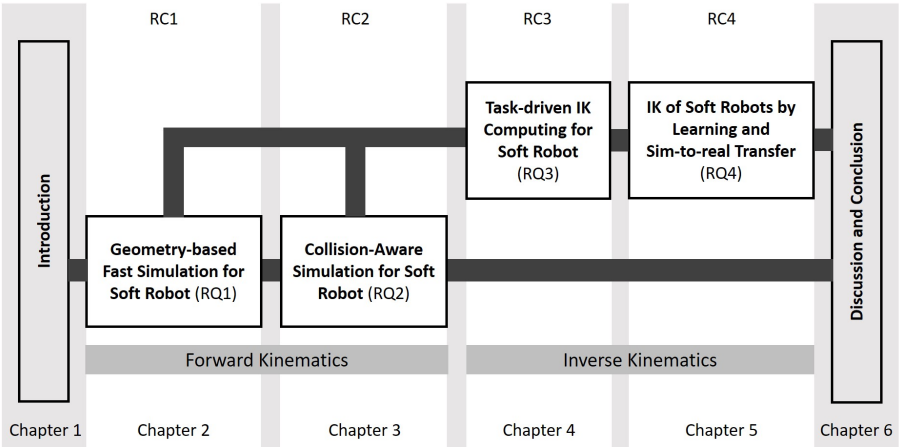


Figure 1.7: An overview of the research cycles and the organization of the thesis.

1.4 Research Cycles and Thesis Structure

The main body of this Ph.D. project contains four research cycles (RCs) that answer the corresponding research questions (RQs) that were defined in the previous section. Fig. 1.7 provides an overview of the thesis. Each research cycle is presented and discussed in a separate chapter. The thesis ends with a discussion and conclusion. Chapters 2-5 include sections that present the introduction, related work, theory, results, and discussion. Short descriptions of each research cycle are presented in the following paragraphs.

RC1: A geometry-based soft robot simulator is developed during the first research cycle. This simulator numerically computes the FK of soft robots and serves as the fundamental function that supports the rest of this research. Various types of actuation are modeled geometrically by changing the shape of (virtual) actuation elements, and the deformation of the robot body is computed by minimizing geometrically-defined elastic energy. The elasticity of soft materials are calibrated in physical tests and can model the behavior of multi-material soft robots. This simulator can balance the computing speed and model accuracy, showing a great convergence for soft robot simulation even with large rotational deformations. The behavior of this simulator is comprehensively studied using different soft robot setups, which all match with the behavior captured in physical experiments.

RC2: This research cycle enriches the functionality of the geometry-based simulator developed in **RC1** by adding the support of collision checking and

response. To realize a real-time speed of collision checking through the deformation process, the *bounding volume hierarchies* (BVH) based collision checking function is invited. Meanwhile, spring elements are utilized to perform effective collision responses through the deformation of the soft robot. The material model is also enhanced to precisely model the performance of extreme soft material with hyperelastic properties. A progressive remeshing method is invited to ensure numerical stability when simulating the performance of pneumatic-driven soft robots with large expansion in chamber regions.

RC3: This research cycle provides the solution to the IK computing of soft robots. Task-specific objectives are defined and optimized with the solver through the gradient descent. In our implementation, gradients are evaluated using the numerical difference in which the FK model developed in **RC1** and **RC2** is applied. The dual-direction line search is used to accelerate the convergence speed of the optimization process. Two case studies including path following and object grasping are used to verify the performance of the developed IK solver. The computed motion sequence can successfully drive soft robots to finish tasks in both virtual environments and physical experiments.

RC4: A learning-based pipeline is built in this research cycle to support a real-time computing speed of IK solutions. The sim-to-real transfer learning is utilized to make the data generation process more efficient and further enhance the precision of computed kinematics results. In particular, the fast simulator built in **RC1** is used to generate massive quantities of data in virtual space to train the network that approximates the mapping for forward kinematics and Jacobian of kinematics objective functions developed in **RC3**. Consequently, a reduced amount of data generated from physical setups is required to train the sim-to-real network that eliminates the reality gap. This hybrid learning method takes advantage of both numerical simulators and learning-based methods. Based on our test, it outperforms existing learning-based methods and can significantly reduce the path-following error for a parallel-structured soft manipulator. Additionally, it can help realize interactive positioning for a multi-section soft finger manipulator.

2

Geometry-Based Direct Simulation for Multi-Material Soft Robots

Robots that are fabricated using soft materials can provide higher flexibility and thus safely interact with natural objects with low stiffness such as food and human beings. However, as many more degrees of freedom are introduced, simulating the behavior of soft robots becomes cumbersome, especially when large deformations are presented. Moreover, when the actuation is defined by geometry variation, it is not easy to obtain the exact loads and material properties to be used in the conventional methods of deformation simulation. In this chapter, we present a direct approach to take the geometric actuation as input and compute the deformed shape of soft robots by numerical optimization using a geometry-based algorithm. By a simple calibration, the properties of multiple materials can be modeled geometrically in the framework. Numerical and experimental tests have been conducted to demonstrate the performance of our approach on various soft robot designs[1].

[1]This chapter is based on papers that have been published as:
Guoxin Fang, Christopher-Denny Matte, Tsz-Ho Kwok, and Charlie C.L. Wang, "Geometry-based Direct Simulation for Multi-Material Soft Robots", *IEEE international Conference on Robotics and Automation (ICRA)*, 2018, pp. 4194-4199;
Guoxin Fang, Christopher-Denny Matte, Rob B.N. Scharff, Tsz-Ho Kwok, and Charlie C.L. Wang, "Kinematics of soft robots by geometric computing", *IEEE Transactions on Robotics*, vol.36, no.4, pp.1272-1286, August 2020.
Note: a few modification and combination has been made to the original published text.

2.1 Introduction

With the excellent behavior of continuum bodies, soft robotics have attracted great attention in research. Mainly inspired by nature, designers have come up with a variety of novel designs for soft robots to achieve different tasks (see [9, 45] for a comprehensive survey). By using soft materials and specially designed structures, continuum bodies enable these robots to generate large and complex deformations with an infinite number of *Degrees-Of-Freedom* (DOFs). Highly dexterous tasks such as human-interactive grasping [46] and exploration in confined regions [47] can be realized with soft robots. In the meantime, 3D printing with multiple materials [48, 49, 50, 51] has been utilized to fabricate soft robots, providing flexibility in the complexity of the geometry as well as the material properties.

While soft matter and 3D printing open up many opportunities in developing new soft robots, these advanced designs along with the high amount of DOFs also bring challenges to develop efficient and reliable algorithms for kinematics. Unlike robots with rigid bodies for which the position and velocity of the end-effector can be directly computed with joint parameters, it is almost impossible to explicitly formulate the kinematic functions of soft manipulators. Although some reduced analytical models have been developed for specific designs, they are usually based on a particular type of soft body and, therefore, are not general enough to model robots with complicated shapes.

A numerical approach can also be used to predict the deformation of soft robots by approximating a continuum body with discretized finite elements. With precise modeling formulation of soft materials, *Finite element analysis* (FEA) has proved its effectiveness in simulating the behavior of soft robots [52, 53]. However, when dealing with large rotational deformation, the high cost of computation by using enterprise-level FEA software (e.g. Abaqus and ComSol) can hardly meet the required efficiency in kinematics applications.

Our research is inspired by the fact that many forms of actuation in soft robotic systems can be directly transformed into geometric changes (see Fig. 2.1). In this paper, we tackle the problem of kinematics computing by presenting an efficient approach where soft robots with multiple materials and their actuation are systematically modeled in a geometry-oriented formulation. In comparison to other methods, our kinematic algorithm demonstrates better convergence and keeps a good balance between the computational efficiency and the numerical accuracy. In this work, we focus on solving the modeling of

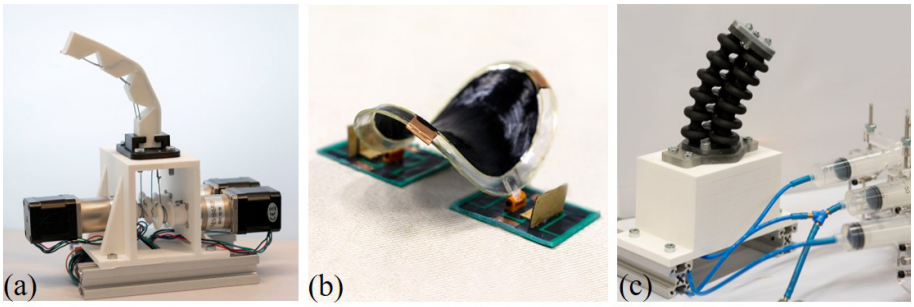


Figure 2.1: Example soft robotic systems that actuation can be represented as geometric changes: (a) a soft finger actuated by stepper motor with cable length shortening, (b) a soft crawling robot driven by *dielectric elastomer actuation* (DEA) can achieve locomotion by the area change using different voltage input [54], and (c) a pneumatic driven soft manipulator controlled by syringe actuation system with the volume change in chambers.

soft robot behavior under actuation (i.e. *forward kinematics* (FK)). Case studies with physical experiments have been conducted to demonstrate and verify the effectiveness of our soft robot modeling approach.

The technical contributions of our work are summarized as follows:

- A novel method of geometric computing is presented to predict the deformation of continuum soft bodies under geometric actuations – this results in an efficient forward kinematic computation. Physical actuations are directly transformed into geometric constraints that can be intrinsically integrated into the framework.
- An image-based calibration method is introduced to enable the simulation of multiple materials in our computational framework by learning the relationship between material properties and shape parameters.

2.2 Related Work

Efficient computation for simulating the deformation of soft robots under different types of actuation is a fundamental technique to solve the problems of kinematics, which is needed in many applications – e.g., adaptive grasping of soft objects in the food industry [55] and auxiliary systems for soft tissues in medical surgery [56]. Prior research works can be classified into three groups: 1) analytical methods, 2) numerical methods and 3) model-free methods (mainly using machine-learning and computer vision).

When dealing with soft robots having simple (particularly symmetric) structures, analytical methods based on mechanics or differential geometry have been commonly used. In the early stages, the backbone curve approach [57] and the constant curvature assumption [58] were applied to build the kinematics of multi-section soft robots. By using work-energy principle, Trivedi *et al.* [59] developed a geometric model for pneumatic-driven soft manipulators that has better accuracy than the constant-curvature model. Michele *et al.* conducted a series of work [60, 61, 42] to build FK and IK models for bio-inspired manipulators by applying the Jacobian method of statics models to compute the equilibrium status of conical shaped manipulators under cable forces. Recently, efforts have also been made to use analytical methods for soft robotic systems with high DOFs or hyper-elastic materials. For example, Panagiotis et al. [62] presented their analysis for fiber-reinforced bending pneumatic actuators. A teeth-structure soft gripper was studied by using a simplified skeleton model [63]. However, the equilibrium of a static model requires specific approximations and assumptions of shape and material properties, which can hardly be generalized to soft robots with freeform shapes fabricated by 3D printing.

While using the numerical method, the deformation of a continuum body is usually simulated by FEA with given material properties and the boundary conditions of actuation. A deformed shape can be computed in general and this method has been used to help select the optimal design parameters of soft robot to meet specific performance (e.g., providing a faster actuation behavior [64] or making the bending curvature conformable to a design surface [65]). Conversely, the trade-off between computing time and accuracy needs to be made when applying a numerical method on real examples with more than 10k elements. Commercial FEA software like Abaqus and ComSol can generate precise calculations of forward kinematics for soft robots [52, 62]; however, small time-steps are needed when confronted with situations of large deformation. For these softwares, high computation cost and slow simulation

speed restrict its usage for further solving the IK problem. To speed up numerical methods, Allison and Okamura [66] presented a closed-loop control of a haptic jamming deformable surface by a mass-spring system. Hiller and Lipson [67] developed a multi-material simulation library for general static and dynamic analysis – called *Voxelyze*, where the voxel representation and beam theory were used. Based on a physics-based simulation engine SOFA [38], Duriez *et al.* [68] simulated the behavior of soft robots by progressively solving a quasi-static equilibrium function for every sample time. This method can achieve real-time computing speed with a reduced model [69]. However, the progressive computation accumulates numerical errors along time steps, which brings in the accuracy problem for the case with large rotational deformation (see the comparison given in Section 2.6).

In the absence of analytical and numerical models, model-free methods based on learning or vision, have been employed to solve the challenge of computational kinematics for soft robots. Machine intelligence approaches can generate forward and inverse mappings with limited samples obtained from either physical experiments [42] or precise numerical simulations [70]. The accuracy of training-based kinematic computation however mainly relies on the quality and quantity of the training datasets. Visual servoing has been used to control the manipulation by calculating the Jacobian of deformation between the control point and an unknown elastic body [71, 72]. Similarly, Li et al. [73] employed an adaptive Kalman filter to estimate the Jacobian and only required data input from the vision tracking system. Zhang et al. [74] built a closeloop tip position control strategy for specific soft robot design by combining the numerical simulator with a visual servoing system. The vision-based methods are efficient and robust after adjusting the control law. However, the requirement of vision hardware and the complex calibration process prevents the usage of this method in many scenarios.

2.3 Geometry-Based Formulation

In this section, we present the formulation of our geometry-based modeling framework. The notations used in this paper are first presented. Then, deformation energy is defined based on the shape variation of elements. After that, bodies and actuators of soft robots are modeled as two types of elements in the formulation. Lastly, the methods for computing target shapes of different elements are presented in detail.

In our method, a volumetric mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E})$ is used in our framework to represent the body of a soft robot, where $\mathcal{V}$ and $\mathcal{E}$ stand for the sets of vertices and elements in the mesh. We define the shape of each element by a $k \times 3$ matrix $\mathbf{V}_i = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_k]^T$ with k being the number of vertices on an element. In this paper, tetrahedron ($k = 4$) and prism elements ($k = 6$) are used to model soft robots with general 3D geometry and thin-shell structure respectively. Meanwhile, the status of actuation is described by a set $\mathcal{C}$ of geometric parameters:

1. Length shortening ratio s for cable actuation
2. Area stretching ratio λ for DEA
3. Volume expanding ratio α for pneumatic actuation.

In this work, the small and capital bold letters are used to present column vectors and matrices respectively, e.g., $\mathbf{v} \in \mathbb{R}^3$ and $\mathbf{N} \in \mathbb{R}^{k \times k}$. The subscript of a variable presents its order in corresponding set, meanwhile the superscript present is for the status of meshes or elements. Particularly, the superscript d denotes the deformed (or current) shape and t means the target status. The identity matrices are denoted by $\mathbf{I}_{k \times k} \in \mathbb{R}^{k \times k}$, and $\mathbf{1}_{k \times k}$ is a matrix of $k \times k$ ones.

2.3.1 Elastic Energy Function

The general purpose of an elastic deformation simulator is to determine a new shape $\mathcal{M}^d$ for a soft body that best mimics the physical behavior of deformation under the actuation of $\mathcal{C}$ with reference to the initial shape $\mathcal{M}$ and the input material distribution Ω. When different boundary conditions (or external loads) are applied to deform an object, the elastic energy is transferred by the corresponding work and distributed internally in $\mathcal{M}$. Here the elastic energy is caused by the shape deformation, which can be evaluated from the strains (i.e., local deformations throughout $\mathcal{M}$). In this sense, the total elastic energy should be minimized when the original shape is preserved as much as possible.

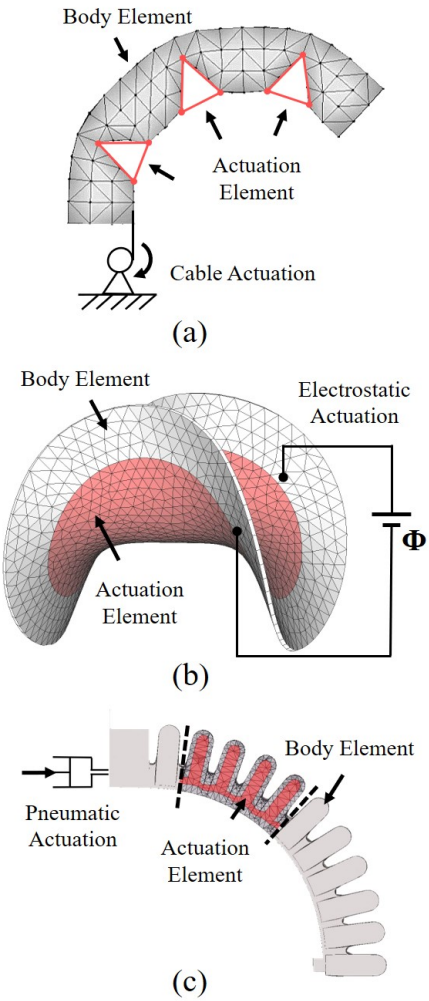


Figure 2.2: Conceptual representation of our geometry-based framework for soft robotic systems with different types of actuation. The light-gray region presents the body elements and the region in red denotes actuation elements. Three different types of actuation are transformed into the shape change of actuation elements: a) cable-drive actuation is formulated as the edge-length shortening of cable elements, b) dielectric elastomer actuation is presented as the area stretching of prism actuation elements, and c) pneumatic actuation is defined as the volume expansion of internal tetrahedral chamber elements.

To mimic this physical phenomenon, we formulate the difference between $\mathbf{V}_i^d$ (current shape under deformation) and $\mathbf{V}_i^t$ (target shape) for a single element by discretized geometry-elastic energy as

$$E_i = D(\mathbf{V}_i^d, \mathbf{V}_i^t). \tag{2.1}$$

To measure the shape difference $D(\cdot,\cdot)$ of $\mathbf{V}_i^d$ and $\mathbf{V}_i^t$, they have to be properly aligned in terms of both position and orientation. Therefore, both shapes are centered at the origin and a rotation is applied to match $\mathbf{V}_i^t$ with $\mathbf{V}_i$, such that the above energy for the i-th element can be further defined as

$$E_i = \omega_i ||\mathbf{N}_i\mathbf{V}_i^d - \mathbf{R}_i(\mathbf{N}_i\mathbf{V}_i^t)||_F^2. \tag{2.2}$$

ω_i is a weight for each element which is normally set as the element's volume (ref. [75]). $||\cdot||_F$ is the Frobenius norm, $\mathbf{R}_i$ is the pure rotational matrix between two status for the i-th element. $\mathbf{N}_i$ is used to transfer an element's center to the origin and $\mathbf{N}_i = \mathbf{I}_{k_i\times k_i} - \frac{1}{k_i}\mathbf{1}_{k_i\times k_i}$.

In this work, only elastic deformations are considered in our framework. Therefore, we can assume that every soft model will come back to its initial rest shape after releasing all the constraints (i.e. actuations and handles). The energy function defined in (2.2) consists of three sets of variables, including:

1. Vertex positions of target shape $\mathbf{V}_i^t$,
2. Rotation matrices $\mathbf{R}_i$ for individual elements and
3. Vertex positions of current shape $\mathbf{V}_i^d$ under deformation.

How to determine these variables is presented below.

We first consider the target shape, which presents the ability of a soft body to resist deformation under actuation. It is determined commonly by the initial model $\mathcal{M}$, the set of constraints $\mathcal{C}$ and the coefficients for material properties Ω. As shown in Fig. 2.2, two types of elements defined in our system – body elements and actuation elements, are modeled by using the same formulation of elastic energy. However, their target shapes are defined in different ways.

- For a body element $\mathbf{V}_i$, the target shape $\mathbf{V}_i^t$ is computed with a shape blending function by combining its initial rest shape and the coefficient R_ω reflecting its material property (i.e., the stiffness). Ideally, $\mathbf{V}_i^t$ is a blended shape between a super-elastic material and a completely rigid material, where R_ω indicates the level of blending (see Section 2.3.2 for the details of blending and Section 2.4 for coefficient calibration).
- Target shape of an actuation element $\mathbf{V}_j^t$ is determined according to the different types of actuations. All actuation elements together actually

serve as the driven handles to deform a soft body. Detailed formulation can be found in Section 2.3.3.

The final energy function is determined by integrating all the elementary elasticity together. By minimizing the integrated energy function for the whole design domain together with actuation constraints, the deformed shape of soft robots under actuation can be computed. As shown in Equation (2.2), $\mathbf{R}_i$ and $\mathbf{V}_i^d$ are unknown variables to be determined during the optimization computation, and the numerical method for solving this nonlinear optimization problem will be presented in Section 2.5. We first present the details of how to compute the target shapes for body and actuation elements below.

2.3.2 Body Elements

For the soft robots fabricated by multiple materials, regions with different materials will deform in different ways, thus the target shape should be computed disparately based on the input material distribution Ω. In this section, we propose a method to formulate soft objects with multiple materials by using linear blending method with a shape parameter.

To model the different properties of materials, a simple way is to assign different weights ω_i for each element in (2.2). The rigidity of an element will be preserved differently through the optimization when different weights are assigned. This mimics the deformation of multiple materials. However, handling the material difference in this way will lead to large approximation errors. In order to gain a better control and reinforce the physical property in large deformations, we control the deformation behavior of elements at the local region by altering their target shapes, $\mathbf{V}^t$, according to different material properties.

Its worth remark that in the simulation system, when the material of an element is extremely hard, it will be rigid during the deformation; respectively, an element with extremely soft material will deform to the shape which conforms to its neighbors while preserving its volume. Based on this remark, we came up with a method to compute two different target shapes for body element as shown in the left side of Fig. 2.3. Here the target shape of a rigid element $\mathbf{V}^r$ comes from the rigid transformation of its original shape. This method thoroughly preserves the initial shape $\mathbf{V}$ and keeps the same orientation as the current shape, which leads to

$$\mathbf{V}^r = \mathbf{R}\mathbf{V} \tag{2.3}$$

$\mathbf{R}$ is the rotation matrix between current and initial element, and can be ob-

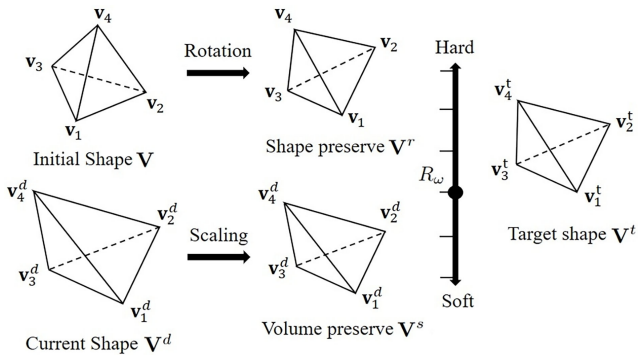


Figure 2.3: The shape blending method for controlling the material stiffness in our framework. (Top-left) The target shape for rigid material is computed by rotating the initial shape to align with the current shape. (Bottom-left) The target shape for extremely soft material is computed by scaling the current shape to preserve the volume of the initial shape. (Right) The shape blending method is applied to align the rigid and the soft materials, and merge their shapes to obtain the target shape for an intermediate material.

tained by applying SVD to the affine transformation between $\mathbf{V}^d$ and $\mathbf{V}$.

For a soft element, its target shape $\mathbf{V}^s$ comes from scaling the current shape back to its original volume (see Fig. 2.3) and we call this volume preservation. The shape comes from current element shape $\mathbf{V}^d$ and can be calculated as

$$\mathbf{V}^s = \mathbf{S}\mathbf{V}^d, \tag{2.4}$$

where $\mathbf{S} = diag(r, r, r)$ is the scaling matrix with $r = Vol(\mathbf{V})/Vol(\mathbf{V}^d)$.

For a material in-between, the rigid and soft target shapes are aligned by using a blending method with a shape parameter R_ω to get the target shape as shown in the right of Fig. 2.3. Here linear shape blending method [76] was used after centering both shapes onto the origin with $\mathbf{N}$ matrix for a general case as:

$$\mathbf{V}^t = R_\omega \mathbf{N}(\mathbf{R}\mathbf{V}) + (1 - R_\omega)\mathbf{N}(\mathbf{S}\mathbf{V}^d). \tag{2.5}$$

In this way, the target shapes of elements according to different materials can be properly controlled during the deformation.

To verify the correctness of above method for controlling the relative stiffness of materials, we have tested a variety of polymer materials widely used in 3D printing. In Section 2.4.2, we present an image-based calibration process to determine the shape parameter – the ratio R_ω – for controlling the material behavior. Our linear shape blending method works very well when the defor-

mation of each element is within the range of linear material elasticity. The correctness of our method will be verified in Section 2.4.1 with the help of FEA simulation.

2.3.3 Actuation Elements

Soft robots are deformed by applying external actuations such as cable shortening, elastomer stretching or pneumatic expansion of a chamber – these are all based on geometric metrics. When being at an equilibrium state, the geometry of an actuation must completely satisfy its given length, area or volume constraints. A straightforward method is to formulate them as hard constraints in a numerical optimization framework. However, it is hard to converge because of its high non-linearity – especially when the initial values are far away from the feasible regions.

To solve this problem of numerical computation, we formulate the deformation of an actuator as the collected function of a set of actuation elements (as shown in Fig. 2.2). The geometric constraints for an actuator are then converted into target shapes computed at each iterative step for these elements. The target shape of an actuation element is achieved by integrating it into the same elastic energy minimization framework. Larger weights are given to the actuation elements to make the actuation parameters satisfied effectively. As a result, the geometric actuation can be seamlessly integrated to our geometry-based simulation framework. Details of how we define the actuation elements and compute their target shape $\mathbf{V}^t$ according to the input actuation parameter $\mathcal{C}$ are given below. Note that, after reshaping from the rest shape, each actuation element should be transformed to a position and orientation according to its current shape – i.e., the similar step as body element. Three different types of actuation elements are considered.

Cable-driven actuation: A typical cable-driven soft gripper with design similar to [77] is as shown in Fig. 2.2(a), which has three soft 'knuckles'. A cable fixed on one side of the gripper is passed through the holes along the gripper. While pulling the cable (i.e., by shortening its length), the gripper bends towards one side. To integrate this actuation into simulation, the V-shaped 'knuckles' are modeled as a set of triangular elements. One edge of each triangle is aligning exactly with the cable, the deformation of which drive the simulations.

The total length L of a cable equals to the length of the gripper. It includes the inside portions L_R and the tooth length $\{l_i\}$ – i.e., $L = L_R + \sum_{i=1}^{k} l_i$,

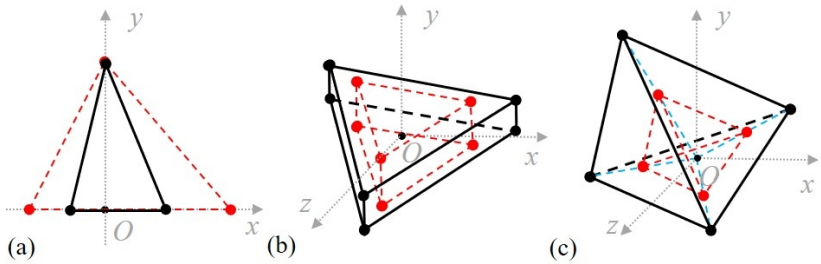


Figure 2.4: An illustration of how the target shape of actuation element is computed based on the input parameter. The initial shapes are presented by red dot lines and the target shapes are displayed by black solid lines. Notice that the number of vertices k is different for different types of actuation elements. Specifically, $k = 3$ for (a) cable actuation, $k = 6$ for (b) dielectric elastomer actuation, and $k = 4$ for (c) pneumatic actuation.

where k is the number of teeth. The shortening factor s is also given together with a cable constrain. The constraint function can be defined as

$$f_c(\mathcal{C}) = sL - (L_R + \sum_{i=1}^{k} s_i l_i) \equiv 0, \tag{2.6}$$

where s_i is a local shortening factor for the i-th tooth. Directly imposing this constraint to the optimization framework will lead to a computation very hard to converge.

It is more efficient to transform this function of constraint to a target shape for each actuation element. For a cable-driven actuation element, we place its rest shape into a position with its cable-driven face located in the xy-plane, the cable coincident with the x-axis and the opposite vertex on y-axis (see Fig. 2.4(a)). After that, the target shape can be computed by shrinking the element along x-axis by the factor s_i.

Dielectric elastomer actuation:

With voltage input, dielectric elastomers can effectively generate large deformation [78]. Driven by DEA, soft robots with specific design can perform locomotion by the areal stretching within the elastomer region. As show in Fig. 2.2(b), a thin-layer soft robot is modeled by prism elements where the inner red region is formed by the actuation elements. The total surface area A of the elastomeric region can be computed by $A = \sum_{i=1}^{k} a_i$ where a_i is the average area of the top and bottom triangles of a prism element. To satisfy the

stretching ratio λ for a DEA, the constraint function can be defined as

$$f_d(\mathcal{C}) = \lambda A - \sum_{i=1}^{k} \lambda_i a_i \equiv 0, \tag{2.7}$$

where λ_i is a local expansion ratio of an actuation element. Similar to the cable-drive actuation, this constraint should also be transformed to the target shape of DEA elements.

When computing the target shape for a prism element from its rest shape, the top and bottom triangles are scaled in their own planes with the scaling ratio $\sqrt{\lambda_i}$. The center of each triangle is chosen as the center of scaling (see Fig. 2.4(b) for an illustration). After scaling, the triangles are shifted along their normal vectors so that the "thickness" of an actuation element is scaled to $1/\lambda_i$ to preserve the original volume.

Pneumatic actuation: A pneumatic actuator usually drives soft robots by pumping pressurized air into a bellow formed by soft materials. An example is shown in Fig. 2.2(c), where the left part is fixed when pumping air along the direction of white arrow into the bellows. The internal tetrahedra that fill the chamber are modeled as the actuation elements, which have been highlighted in Fig. 2.2(c). These actuation elements are used to model the expansion of air inside the bellows.

Given the volume u_i of each pneumatic actuation element, the total volume of a bellow is then $U = \sum_{i=1}^{k} u_i$. To achieve the volume expansion ratio α for a pneumatic-driven soft robot, the geometric constraint can be described by:

$$f_p(\mathcal{C}) = \alpha U - \sum_{i=1}^{k} \alpha_i u_i \equiv 0. \tag{2.8}$$

where α_i is a local expansion ratio of each element.

The target shape for a pneumatic actuation element with the volume expansion ratio α_i can be determined by scaling its rest shape with the ratio $\sqrt[3]{\alpha_i}$. The scaling is conducted at the center of tetrahedron (see Fig. 2.4(c)). After scaling, the target shape should be transformed to a position and orientation according to the element's current shape.

We determine them proportionally to the ratios of an element's current shape w.r.t. its rest shape. The newly determined ratios must also satisfy the geometric constraints defined in (2.6), (2.7), and (2.8). In our implementation, a least-norm solution is employed to compute their values on all the actuation elements.

2.4 Material Property: Analysis and Calibration

To formulate the deformation of soft robots made with multiple materials, we proposed a reduced model based on the linear shape-blending method presented in Section 2.3.2. The effectiveness of our method mainly relies on two conjectures.

- For a variety of smart soft robot designs, the large deformation of continuum body is mainly generated by structural deformation instead of elemental deformation – i.e., the strains are relatively small.
- For many materials widely used for the fabrication of soft robots by 3D printing, the material elasticity in the range with small strain can be approximately described as a linear model.

In this section, these two assumptions are verified by both the FEM simulation and the material tests. After proving the correctness of our method, an image-based calibration process is proposed to find a shape parameter to be used in our method corresponding to the physical behavior of materials.

2.4.1 Linear Material Elasticity

Many materials used for fabricating soft robots can be largely stretched and have hyper-elastic material property, which was utilized to achieve large shape change under actuation in early years. However, recent designs of soft robots have specially designed advanced structures to realize more reliable deformation with better durability. For example, inextensible layers [51] are used to prevent the non-directional expansion so that the effectiveness of an actuator is tremendously enhanced. In these cases, extreme local stretch is no longer necessary for realizing a large global deformation. We study the range of elemental deformation on a widely applicable soft finger structure [79] and another smart design of soft manipulator [80].

Tensile tests have been conducted on two materials used in fabricating these two soft robots – Ultimaker TPU 95A and Aglius 30. The obtained stress-strain curves are shown in Fig. 2.5(c). The strain-stress relationship is nonlinear in general. However, when deformation occurs in a range with small strains, the relationship can be linearly approximated with small error. Specifically, when the strain is less than 20% for TPU and 30% for Aglius, a linear stress/strain curve can be obtained (see also the solid and dash lines shown in the zoom-view of Fig. 2.5(a) and (b)).

We conduct the FEM to further study the strains generated on these two

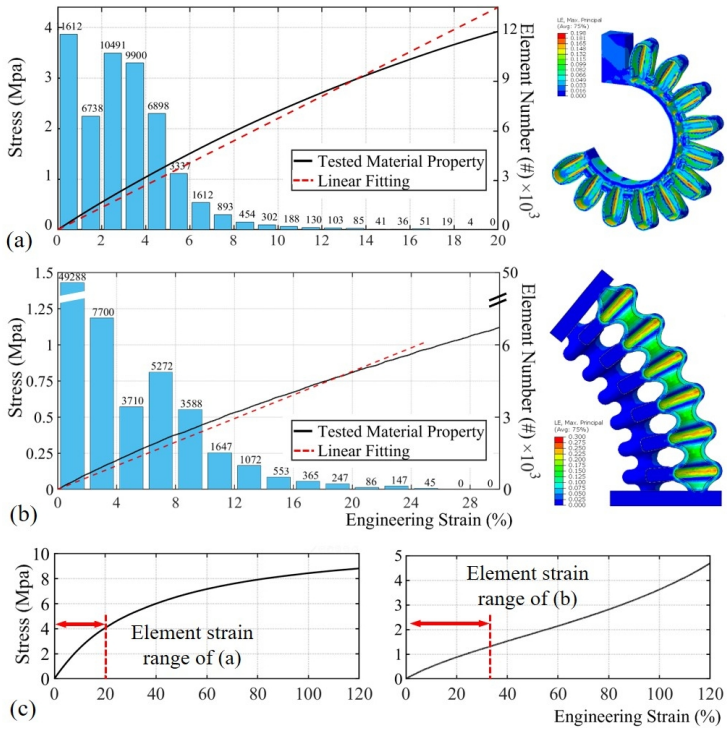


Figure 2.5: Verification of small-strain assumption on two effective designs of soft robots fabricated by soft materials: (a) Ultimaker TPU 95A and (b) Aglius 30. The strain distribution of body elements (shown in the right) is generated by FEM simulation, and the histograms (left) show the statistics of strains on these two designs under large structural deformation. (c) Stress-strain curves for Ultimaker TPU 95A (left) and Aglius 30 materials (right) obtained by physical experimental tests. It can be observed that the elemental deformation mainly occurs in the range with linear material elasticity.

designs of soft robots. Abaqus software is employed to generate the strain distribution when large structural deformation has been achieved on these two structures. In Fig. 2.5, the histograms are used to visualize the statistical distribution of strains in all elements. It can be easily found that the strains are less than 20% for most regions and all fall in the range of linear elasticity discussed above – i.e., less than 20% for TPU and 30% for Aglius.

Note that, large elemental deformation can be achieved under actuation for the materials with small Young's modulus such as silicon rubber. This material property was employed for some designs developed in early years. For these cases, the elasticity is not guaranteed to be linear for all elements,

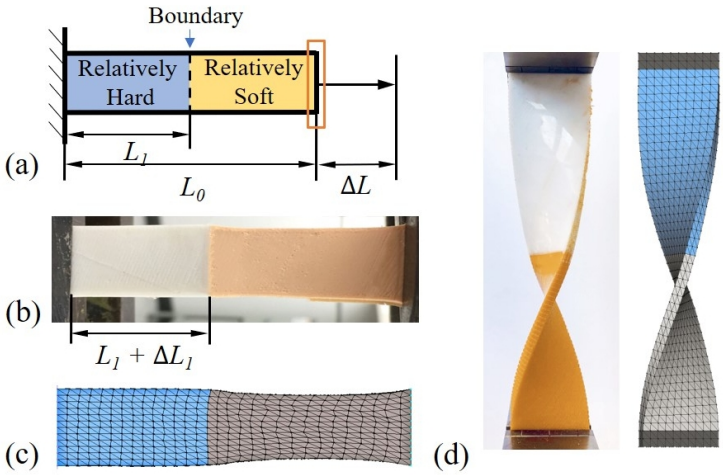


Figure 2.6: Image-based calibration of the shape parameter for simulating objects with multiple materials: (a) a multi-material bar with displacement on the right, (b) a physical elongation test on 3D printed specimen using NinjaFlex and Flexible *polylactic acid* (PLA) materials, (c) the tensile test result generated by our simulation framework after calibrating the shape parameter R_ω, and (d) the twisting test [49] is also conducted to verify the correctness of our material elasticity and the calibration method.

Table 2.1: Calibrated Parameters for Different Material Combinations

Material A	Material B	R_m	R_ω^A / R_ω^B	Actuation
TPU 95A	NinjaFlex	3.75	7.08	Cable
Tango Black	Mixed Aglius	5.68	10.30	Pneumatic

which brings modeling uncertainty although our method can still successfully predict the deformation in practice.

2.4.2 Calibration of Shape Parameter

After verifying the correctness of using the linear elasticity simplification for body elements, our shape-blending method needs to define proper shape parameters to mimic the real physical behavior. Rather than calibrating each material separately in a tensile test, an image-based method is developed to calibrate the relative properties between different materials. As shown in Fig. 2.6, we impose the displacement on a rectangular specimen at one end while fixing

another end. Without loss of generality, the specimen is fabricated with two materials A and B joined with a sharp interface. Let the length of the whole specimen be L and the distance between the interface and the fixed end be L_1, where different values of $L_1 \in (0, L)$ are used for different specimens. When imposing a displacement ΔL at the free end of the bar, the displacement of the interface will be located at $\Delta L_1 \in (0, \Delta L)$ depending on the relative material properties between A and B. The relationship of two materials can be presented by an *elasticity ratio*, which is mathematically defined as

$$R_m = \frac{\epsilon_A}{\epsilon_B} = \frac{L_1(\Delta L - \Delta L_1)}{(L - L_1)\Delta L_1}, \tag{2.9}$$

where ϵ_A and ϵ_B are the strains in the regions of two materials with A being linked to the fixed end and B locating at the free end. Note that, for linear materials, R_m also equals to the ratio of Young's modulus (i.e., a constant when materials are given). The rest of the problem is how to find the corresponding value of the shape parameter R_ω after obtaining the elasticity ratio R_m on two materials through the physical tests. The basic idea of our calibration is to apply different values of R_ω to run the elongation tests in our geometry-based simulation by the same setup. The value of R_ω is then determined by matching our simulation results with the results of physical tests, where the bisection-search method is used. With a well calibrated parameter R_ω, the position of the material interface generated by our simulator matches well with the physical experiment accurately (see Fig. 2.6(c)). To further verify the generality of this parameter, we conduct a twisting test similar to the one presented in [49]. As shown in the left of Fig. 2.6(d), the specimen with two materials gives a symmetry torsion where the relatively soft region has a larger rotation angle. By using the same calibrated material parameter R_ω, our simulation generates a similar result (see the right of Fig. 2.6(d)).

We have applied this calibration method to various materials used for fabricating soft robots. For different 3D printing systems, different combinations of materials are tested and the corresponding calibrated parameters are listed in Table 2.1. The effectiveness of our shape-blending based deformation model and the calibration method is further validated in the next section by other experimental tests taken on different robot designs.

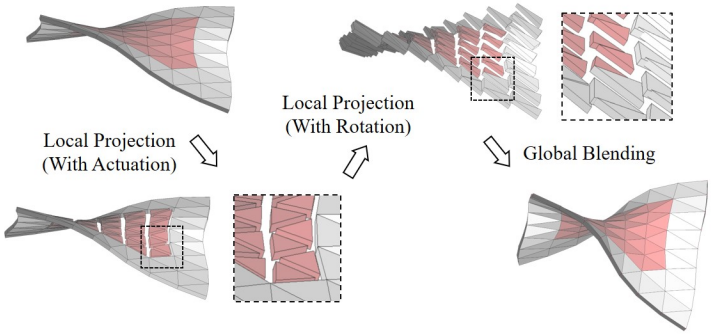


Figure 2.7: An illustration of the local-global optimization process on a simple model where the whole mesh is actuated by shrinking elements in the red region.

2.5 Algorithms for Forward Kinematics

The kinematics of soft robots are hard to be solved analytically. In this section, we present the algorithms characterized by our geometry-based formulation to solve the forward kinematics for soft robots. As a general framework, our algorithms for kinematics can intrinsically handle the different configurations of actuation with different material-distributions as long as the actuation can be converted into geometric inputs.

The forward kinematics for the soft robots can simply be described as the computation of the deformed shape $\mathcal{V}^D$ from the initial shape $\mathcal{V}$ given the actuation in the form of constraints, $\mathcal{C}$. As formulated in Section 2.3, the deformed shape of a soft body can be computed by minimizing the elastic energy after converting the actuation constraints into a set of target shapes for the actuation elements. This leads to a solution of forward kinematics in our framework by solving the unconstrained optimization problem below.

$$\min_{\mathcal{V}} E(\mathcal{M}, \mathcal{C}) = \sum_{i=1}^{m+n} w_i \, Vol(\mathbf{V}_i) \|\mathbf{N}_i \mathbf{V}_i^d - \mathbf{R}_i (\mathbf{N}_i \mathbf{V}_i^t)\|_F^2 \tag{2.10}$$

where the variables of optimization are the vertices $\{\mathbf{V}_i^d\}$ of a deformed shape. In this framework, the final shape of a soft body under actuation is determined by the initial shape of n body elements and the target shape of m actuation elements.

In the formulation of (2.10), both the local rotation $\mathbf{R}_i$ and the vertex positions $\mathbf{V}_i^d$ are unknowns to be determined. As a result, the objective function of optimization is highly non-linear which may lead to a very slow convergence

Algorithm 1: ForwardKinematicComp

Input: The initial shape $\mathcal{V}$, the actuation $\mathcal{C}$ and the material distribution Ω.

Output: The deformed shape $\mathcal{V}^d$

1 Initialize the weights $\{w_i\}$ and volumes $Vol(\mathbf{V}_i)$ for all elements;
2 Apply factorization to the normal equation of (2.11);
3 **repeat**
    `/* Local / global optimization */`
4 Compute the target shape for each element;
5 Applying SVD to obtain the rotation matrix $\{\mathbf{R}_i\}$;
6 Determine $\{\mathbf{v}_j\}$ by solving the linear system where the factorization can be re-used;
7 Update $\mathcal{V}^d$ by the new positions of vertices $\{\mathbf{v}_j\}$;
8 **until** *the position change is less than* 10^{-5} *on all vertices*;
9 **return** $\mathcal{V}^d$;

and high computation time. In order to solve it efficiently, a local / global scheme akin to [81] is employed. In the step of *local projection*, the initial shapes of actuation elements are first deformed according to the actuation parameters (as introduced in Section 2.3.3). After that, the target shapes of all elements are independently transformed by applying the rigid transformation determined between their target shapes and the current positions (i.e., the rotation matrices as discussed in Section 2.3.2). Then, the new positions of vertices can be computed in the *global blending* step by minimizing the energy. Letting

$$\frac{\partial E(\mathcal{M}, \mathcal{C})}{\partial \mathbf{v}_j} = 0 \quad (\forall \mathbf{v}_j \in \mathcal{V}) \tag{2.11}$$

leads to a least-square problem that can be solved efficiently.

Through this global blending step, the incompatible positions of a vertex in different elements are "glued" together. An illustration of this local-global computation can be found in Fig. 2.7. Notice that, w_i, $Vol(\mathbf{V}_i)$ and $\mathbf{N}_i$ are constant during the iterations for minimizing $E(\mathcal{M}, \mathcal{C})$, factorization of the normal equation defined by (2.11) can be pre-computed and reused to accelerate the computation of optimization. In order to well-preserve the constraints of actuation, a larger weight as $w_i = 5.0$ are employed for the actuation elements while keeping $w_i = 1.0$ for all other body elements. The pseudo-code of our algorithm can be found in **Algorithm 1**.

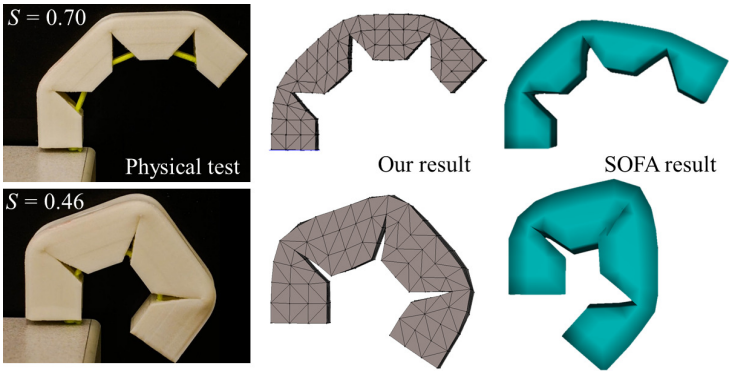


Figure 2.8: Comparisons of a cable-driven gripper among the physical test (left), our simulation (middle), and the simulation by the SoftRobots plug-in for SOFA [68] (right).

2.6 Implementation on Different Robots

We have implemented our geometric computing based kinematic algorithms for soft robots in C++ and tested on a standard PC with an Intel E5-1653 3.5GHz CPU and 16GB RAM. With the help of parallelization on multi-core CPU on the numerical solver Eigen [82], our system can support the computation of forward kinematics for models with up to 50k tetrahedra in real-time (i.e., 25 fps). In this section, the results of forward kinematic computation for soft robots will be first presented and compared with existing numerical modeling methods.

The results generated by our forward kinematics algorithm on a deformed soft body are validated by physical tests. Moreover, our method is also compared with different simulation techniques. The models of soft robot are digitally represented by tetrahedral meshes, and their corresponding physical objects are fabricated by multi-material 3D printer (e.g., Ultimaker 3 and Object 350 Connex3). The properties of soft materials are evaluated on a Zwick Roell static testing machine.

2.6.1 Cable-driven Soft Finger

The first test is conducted on a cable-driven gripper with single material (Flexible PLA) as shown in Fig. 2.8. The top and bottom rows show two sequences of deformations at different time instants, where from left to right show the results of physical test, our simulation and SOFA [68]. Due to the reason that

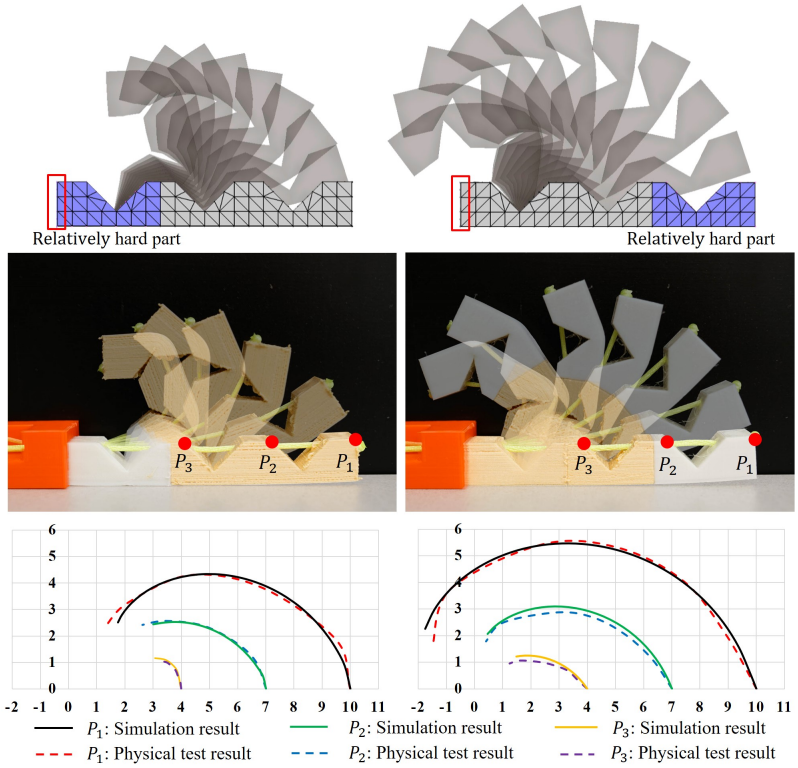


Figure 2.9: Two cable-driven soft grippers (left and right) with different material distributions have different behaviors under actuation. Locations of markers determined by our simulation are well-matched with theirs in physical test.

the ‘deformations are progressively computed for each time step and the accuracy is traded off for computational speed in SOFA, its results do not match with the physical tests in large deformation. Specifically, simulation starts to vary from reality when cable length change is larger than 45%. Meanwhile, to verify the result of our forward kinematic computation for multiple materials, we test two cable-driven grippers with different material compositions. The simulation and physical results are compared visually with its dynamics in Fig. 2.9. The deformations are also compared quantitatively by the trajectory of three corresponding markers located on the boundary of the grippers. It can be seen that both results match with the physical experiments very well.

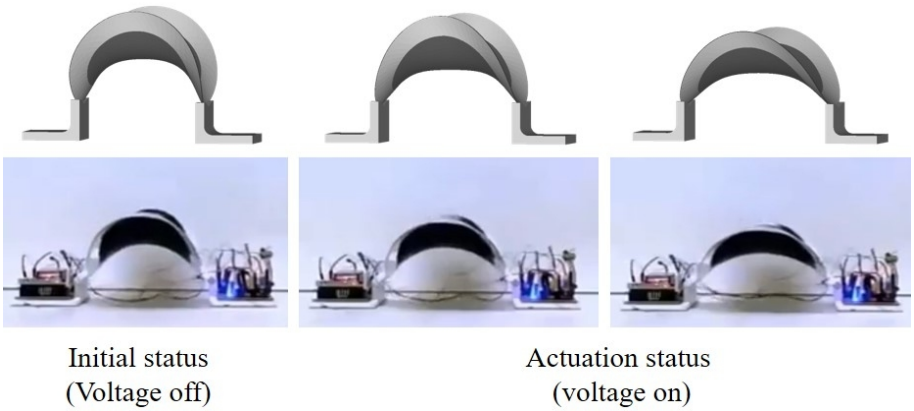


Figure 2.10: Simulation result for a soft crawling robot by geometry modeling the electrostic-driven stretching behavior of DEA. (Top) The results of our forward kinematic computing. (Bottom) The locomotion behavior of a real robot [83].

2.6.2 DEA Soft Robot

A design of soft crawling robot [83] is studied to validate our approach on the DEA in FK computing. The actuator is fabricated by attaching bendable PET frames to a pre-stretched elastomer membrane. After releasing the constraint, the elastomer layer will shrink and drive the soft body deforming to its initial status (as shown in the left of Fig. 2.10). By applying the voltage to the electrodes, the elastomer region will elongate the soft body. Two rigid legs are attached to the robot and always kept on the x-y plane during simulation. In our simulation, an initial stretching ratio $\lambda_{init} = 0.7$ is used on actuation elements to first deform the planner model and get the initial shape. We use the voltage and stretch relationship in [84] to determine the parameter λ for the actuated status.

2.6.3 Pneumatic-driven Soft Robot

We test the behavior of proposed simulator on a pneumatic soft gripper by increasing the pressure of the air pumped into the chamber to control bending of the gripper. We quantitatively present the accuracy of our method by tracking the tip position of a soft gripper. As shown in Fig. 2.11, our result matches well with the analysis conducted by advanced FEA software as well as the physical experiment. When using the similar number of tetrahedra in the computation (i.e., around 45k), the computation of our framework is much faster – with

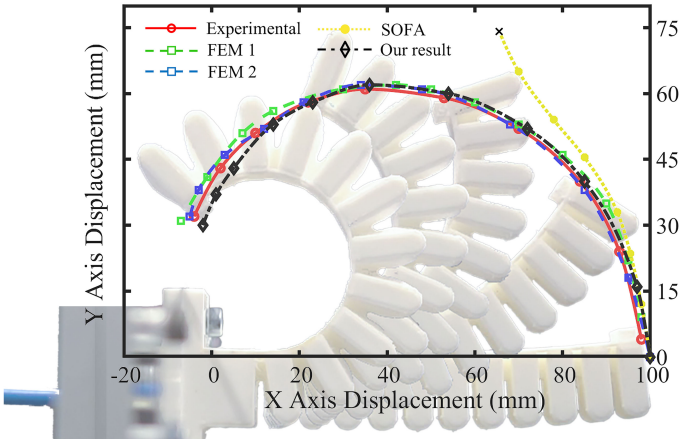


Figure 2.11: Trajectories for a soft finger's tip under pneumatic actuation. The background image shows the bending results in real physical test. The results of three different numerical simulators are presented: 1) finite element analysis with linear (FEM 1) and non-linear material properties (FEM 2), 2) the SOFA simulator and 3) our method. The FEM results are generated by Abaqus.

23.2 seconds vs. 13.6 minutes required to complete the simulation for bending the soft actuator up to 240 degrees by the Abaqus software. Meanwhile, we test this soft model on the SOFA platform with similar mesh size. The computing time has been reported in Table 2.2. Noticed that the simulation speed of SOFA is faster than our method; however, the result begins to become unrealistic after being bent for more than 90 degrees (the yellow trajectory shown in Fig. 2.11). In contrast, our simulation can produce very realistic results for large deformation case while still having a fast speed in FK computing.

Table 2.2: Computational Costs for Different Methods of Simulation

Method	Element #	t_{90° (sec.)	t_{180° (sec.)	t_{240° (sec.)
FEM 1†	44774	240	529	820
FEM 2	44774	288	636	1068
Our Method	45802	8.5	15.2	23.2
SOFA [68]	44900	3.8	-	-

†Simulations of finite element analysis use approximated linear material propriety in FEM 1 and nonlinear model in FEM 2 – both by the Abaqus software.

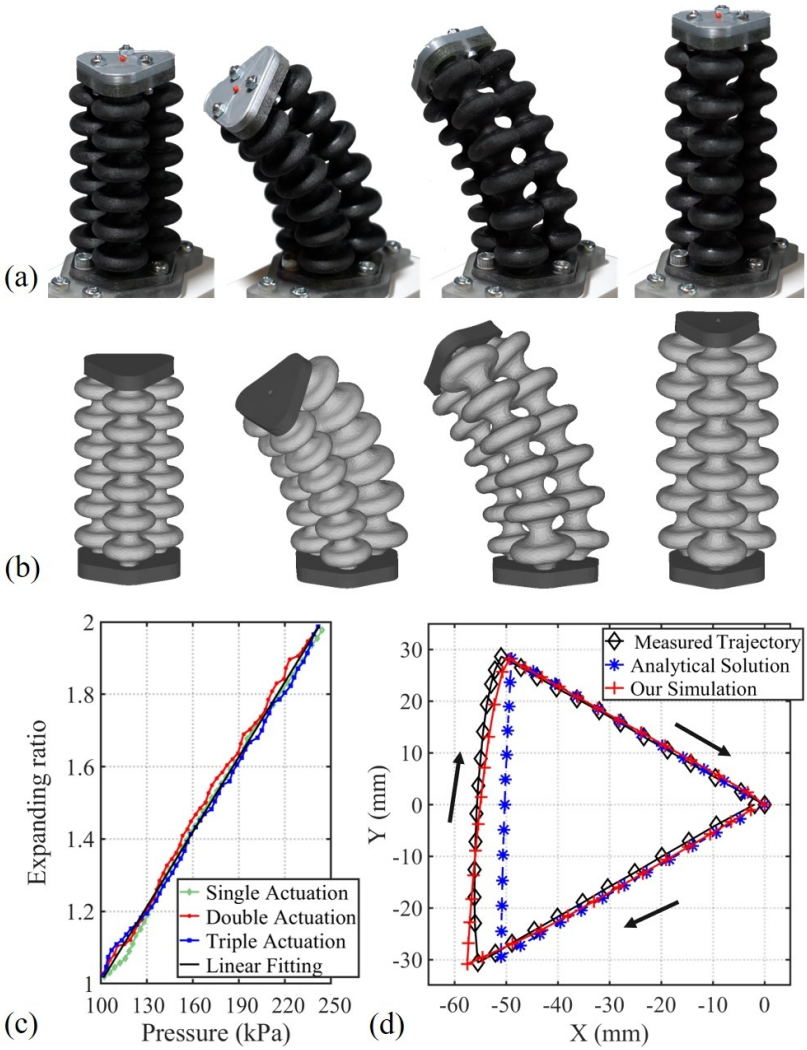


Figure 2.12: The experiments taken on a multi-chamber 3D printed pneumatic-driven soft actuators, which is reproduced from [85]. (a) The physical behavior under actuation – from left to right: rest shape, expanding one chamber, expanding two chambers and expanding all chambers with the same volume-change ratio. (b) The results computed by our forward kinematic algorithm. (c) Calibrated ratio for the relationship between pressures and expanding ratios under different actuation statuses. (d) Study the trajectories for the tip point (shown as the red dot in (a)) by comparing the results of physical tests, analytic computation [80] and our forward kinematic algorithm, where the black arrow shows the direction of tip moving.

Our simulator has also proves its effectiveness in modeling the behavior of a pneumatic-driven three-chamber soft robot. This design was originally presented in [85], and the robot has three chambers that can be actuated individually to bend its body in 3D space. The soft part of the actuator is fabricated using the Object 350 Connex3 printer with the mixture between a rigid (Vero-Magenta) and a soft material (Agilus 30 Black), which has a rigidity of shore 70A hardness. In our experiments, the three chambers are pressurized one after another by $P = 100 \sim 240\ \mathrm{kPa}$, and the related hysteresis curve $\alpha(P)$ can be found in Fig. 2.12(c).

The forward kinematic computation for actuating multiple chambers is presented in Fig. 2.12(b) and compared with analytic computation [80] and physical tests. The trajectories of the tip's movements are plotted in Fig. 2.12(d). It is easy to note that our algorithm can generate results more accurately than the analytic prediction method presented in [80], which determines the position of an investigated point by simply combining the prediction results of individual chambers. The maximum tracking error (on every waypoint) observed in the results of forward kinematic computation is less than $3\mathrm{mm}$ throughout the whole trajectory. The dimension of this soft actuator is $48\mathrm{mm} \times 48\mathrm{mm} \times 136\mathrm{mm}$ and the model used for kinematic computing contains $135k$ tetrahedra.

2.7 Conclusion and Discussion

In this chapter, we present a fast simulator for soft robots based on geometric computing. In our method, both the soft body of robots and different types of actuation are modeled as geometric elements that are integrated into an energy optimization formulation. Meanwhile, the distribution of multiple materials on the body of a soft robot is formulated by giving different stiffnesses to different elements, where the stiffness is represented by a calibrated shape parameter in our framework. We have proposed an efficient optimization-based algorithm for solving forward kinematics. Our method is fast and adaptable for various actuation types and can handle soft robots with complex designs. In comparison to existing soft robot modeling solutions, our method balances efficiency and accuracy in computing. In particular, it demonstrates excellent performance in convergence and robustness when dealing with large rotational deformation. We have conducted several physical experiments to validate the accuracy of our framework.

As a numerical method, it represents soft robots in a discrete form as vol-

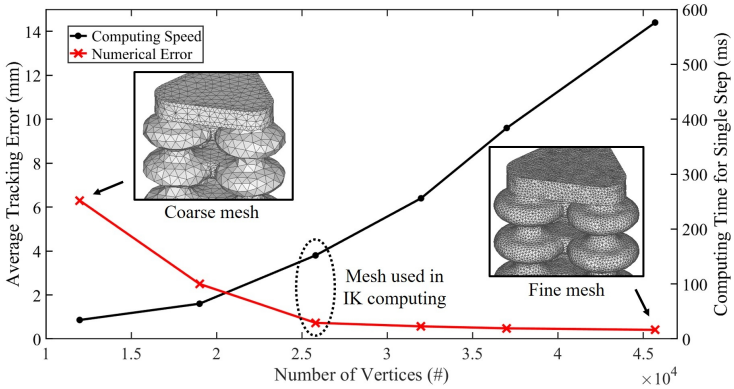


Figure 2.13: The performance of our kinematic computing method on meshes with different resolutions from coarse to fine. As can be expected, a finer mesh takes more time on each iteration but can generate more accurate results. In our practice, we use a relatively fine mesh that can also give very accurate prediction (i.e., a mesh with $25.8k$ vertices as shown in the circled dash line).

umetric meshes. Our framework supports different types of elements that can precisely describe the model (e.g. tetrahedron for general 3D shape and prism for thin-shell structure). The average time used to compute forward kinematics for a single step of iteration keeps a nearly linear relationship with the number of vertices in a mesh (see Fig. 2.13). Simultaneously, we also observe that accurate results could be achieved when a relatively fine mesh is employed to conduct the simulation. Specifically, the average tracking error is less than 1mm for the three-chamber soft manipulator when a relatively fine mesh is employed. In real applications, we always adjust the mesh density and compare the results of FK computing to seek a good balance between accuracy and efficiency. In the next chapter, the progressive remeshing process is discussed to precisely model the behavior of pneumatic-driven soft robot where the chamber region is extremely expanded.

The effectiveness of the simulation proposed in this chapter relies on the small strain assumption of elements during the deformation. Therefore, the current material model needs to be further extended to support cases with large local stretch – e.g., the soft robots fabricated with silicon rubber. In the next chapter, the simulator is enhanced to support the modeling of soft material with hyperelastic properties. Meanwhile, the support of collision checking and response is added to the simulator.

3

Collision-Aware Fast Simulation for Soft Robots by Optimization-Based Geometry Computing

Soft robots can safely interact with environments because of their mechanical compliance. Self-collision is also employed in the modern design of soft robots to enhance their performance in different tasks. However, developing an efficient and reliable simulator that can handle the collision response well is still a challenging task. In this chapter, we enhance the fast geometry-based simulator developed in Chapter 2 to support the modeling of soft robots under collision. A highly efficient and realistic collision checking / response model is developed, and the hyperelastic material property is incorporated. Both actuated deformation and collision response for soft robots are formulated as geometry-based objectives. The collision-free body of a soft robot can be obtained by minimizing the geometry-based objective function. Unlike the FEA-based physical simulation, the proposed pipeline performs a much lower computational cost. Moreover, adaptive remeshing is applied to achieve the improvement of the convergence when dealing with soft robots that have large volume variations. Experimental tests are conducted on different soft robots to verify the performance of our approach[1].

[1]This chapter is based on the paper that has been published as:
Guoxin Fang, Yingjun Tian, Andrew Weightman, and Charlie C.L. Wang, "Collision-Aware Fast Simulation for Soft Robots by Optimization-Based Geometric Computing", *The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, accepted.
Note: a few modification and combination has been made to the original published text.

3.1 Introduction

Powered by the flexibility of soft materials and novel structure designs, soft robots can perform complex deformation in their body shapes and safely interact with other objects [86]. Thus, they have many advantages in certain tasks, such as grasping fragile objects [87, 88] and exploring confined environments [89, 16], which are challenging for conventional robots that have rigid bodies. When controlling rigid robots to complete tasks, robot-robot collisions and robot-environment collisions are generally prohibited. A simplified solution is to avoid contact. However, this is not the case for soft robots since the phenomenon of contact caused by collision plays an important role in allowing them to safely interact with environments and their own bodies. One should note that self-collision has been employed in modern design to achieve advanced functionality [90, 91]. For example, the soft gripper presented in Fig. 3.1 has the ability to interact and grasp objects with its soft body. Meanwhile, the self-collision that happens in the gap region between chambers can enhance the stiffness of the grasping action and provide faster responses in bending deformation (ref. [92, 93]).

3.1.1 Challenges in Collision-Aware Simulation for Soft Robots

At present, the design process of soft robots heavily relies on the experience of engineers. Building effective computational tools is essential for accelerating the process. Studies have been conducted on using FEA in the loop of design optimization (e.g., [94, 95, 96]). Meanwhile, predicting the behavior of soft robot under actuation also plays an important role in building fast model-based controllers [97].

Efficiently simulating the deformation of soft robots while considering self-collisions and interactions with obstacles is still a challenging problem. Incorporating the collision response into the simulation of soft robots requires the information of whole body shapes. In this case, FEA-based simulator is more general and effective than the analytical models (e.g., [98]). Nevertheless, the hyperelastic material properties of soft materials and the highly nonlinear deformation presented on soft robots [89, 99] (e.g., expanding, twisting, and bending) make obtaining an accurate result from simulations quiet time-consuming.

The challenges of building collision-aware simulation for soft robots are summarized below.

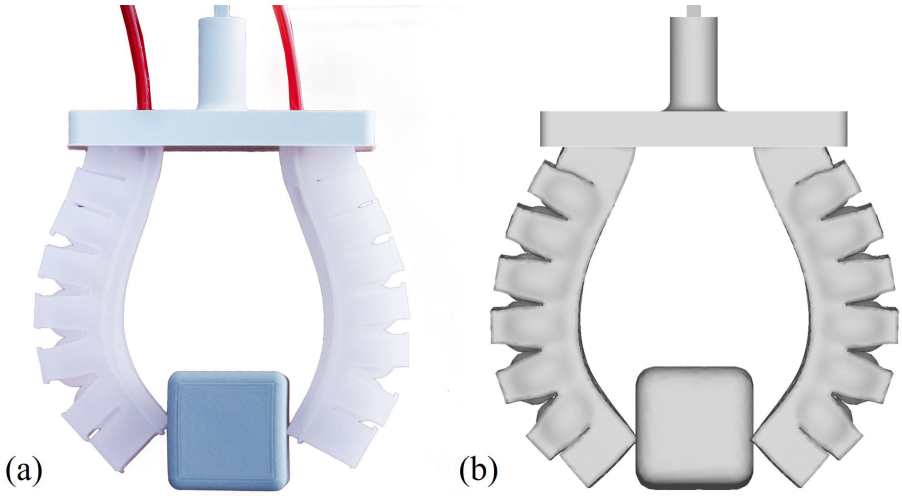


Figure 3.1: Pneumatically actuated soft gripper that can effectively grasp objects by large inflation of chambers and the self-collision between neighboring chambers: (a) the physical result on a soft gripper made by silicone casting and (b) our simulation result that can well predict the shape of the soft gripper by considering both the robot-robot and the robot-obstacle collisions.

1. *Complexity in deformation:* The involvement of highly nonlinear deformation makes FEA-based simulations difficult to converge in numerical iterations [100]. For example, the largely inflated chambers for pneumatic-driven soft robots can lead to highly distorted elements for FEA, which introduces numerical instability.
2. *Fast collision checking through actuation:* Unlike articulated robots with rigid bodies, the freeform deformable body shape of a soft robot makes collision detection and self-collision detection more difficult to conducted efficiently.
3. *Efficient collision response with hyperelastic material properties:* Once (self-)collisions are detected, a response algorithm needs to modify a soft robot's current shape to eliminate collided regions. Providing effective collision response while being able to mimic the complex physical behavior of hyperelastic materials is challenging.

In this work, we propose a method of optimization-based geometric computing to tackle the mentioned challenges.

3.1.2 Related Work

To model the behavior of soft robot with interaction, full FEA-based simulations are most widely used due to their proof of high accuracy [101, 94]. However, the computational cost is high since a small time step needs to be used to solve the nonlinear geometry change (mainly rotational) of a collision response. Reduced FEA-based numerical models that are based on voxel [102] or particle [103] approximation can achieve simulation with real-time speed. Based on off-the-shelf physics engine of rigid-body simulation, Graule *et al.* [104] built a model to predict the behavior of a continuum robot finger when interacting with a complex environment. The self-collision issue was not considered in these frameworks. A general simulator that was based on force-equilibrium function was developed by Duriez *et al.* [105], and it can achieve the speed of real-time simulation with the help of model reduction [106]. This work was recently enhanced by the self-collision response and has improved its accuracy when controlling cable-driven soft robots [93]. However, the (self-)collision region was defined during the pre-processing step and was fixed through actuation. Furthermore, their method, which is based on integration along time steps, accumulates the error; therefore, the simulation could become non-realistic for pneumatic-driven soft robots with large expanding and complex rotational deformation, which has been discussed in [13].

For applications such as cloth simulation and animation, research has been conducted in computer graphics (e.g., [107, 108, 109]) to tackle the problem of collision detection and response for deformable objects. Methods based on *bounding-volume hierarchies* (BVHs), distance fields, and spatial partitioning were invited for fast collision detection (see [110] for a comprehensive review). In this work, BVH-based acceleration is applied to support real-time (self-)collision detection for soft robots. For collision response, the method presented in [111] was based on using constrained optimization to minimize the collided region between two objects. Penalty force-based method was also utilized to accelerate the collision-response process with time integration [112, 113], which however suffers from the difficulty of tuning parameters to avoid unrealistic results in simulation. Meanwhile, none of these works incorporates the hyperelastic properties of materials that are commonly used to fabricate soft robots. In this paper, we bring the idea from spring-mass systems [114] and build the 'virtual' spring elements for collision response. When the spring energy for these elements is minimized, collision-free is achieved on the bodies of soft robots.

3.1.3 Our Method

In this work, we present an optimization-based simulator that can efficiently predict complicated deformation on soft robots and effectively provide realistic collision response by incorporating hyperelastic material properties. Our method extends the pipeline of geometry computing that is presented in Chapter 2. The elastic energy in soft body is minimized to compute its deformed shape, and the actuation is formulated as constraints in terms of length, area or volume variation applied to some elements. After using axis-aligned bounding boxes (AABB) as BVH to support real-time (self-)collision detection, we introduce a group of 'virtual' spring elements to link collided points with their collision-free corresponding vertices. The hard-to-solve collision response constraint is then be reformulated as a quadratic energy term to be minimized. A local-global iterative solver is employed to find the collision-free shape under certain actuation. The geometry-based hyperelastic material model [115] is adopted in our approach, where the material parameters are obtained from physical calibration.

The technical contributions of our work include the following:

- A geometry-based simulator that can effectively predict the shape of soft robots with large and complex deformation by incorporating adaptive remeshing;
- Applying a BVH-based fast collision checking method to detect both robot-robot and robot-obstacle interactions through the deformation of soft robots;
- A spring-element-assisted collision response model that incorporates hyperelastic material properties.

The generality and performance of our collision-aware soft robot simulator have been tested and verified on different soft robots fabricated from different materials. In all experimental tests, our simulator can successfully mimic the physical behaviors of soft robots with high accuracy. When compared with FEA-based methods, our simulator shows better performance in its efficiency and capability of numerical convergence.

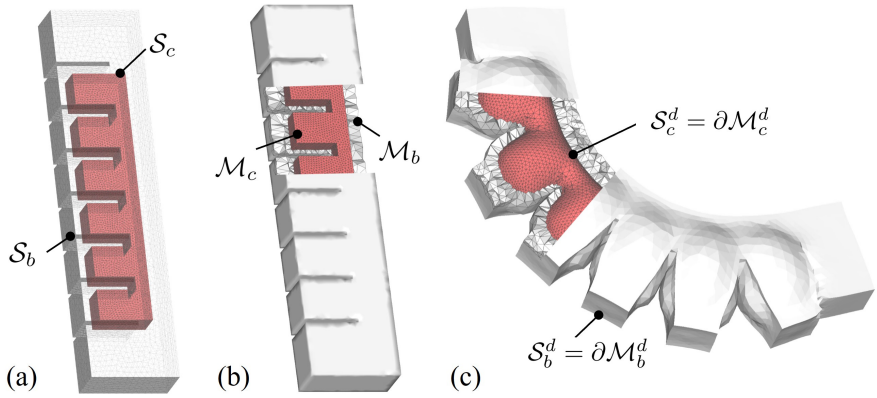


Figure 3.2: (a) The input shape of a soft robot is represented by two surface meshes $\mathcal{S}_c$ (as chamber) and $\mathcal{S}_b$ (as body). (b) Volume tessellation is applied to generate tetrahedral meshes $\mathcal{M}_c$ and $\mathcal{M}_b$. (c) Inflation in the chamber region drives the soft robot being deformed to $\mathcal{M}^d = \{\mathcal{M}_c^d, \mathcal{M}_b^d\}$. Adaptive remeshing is applied to update $\mathcal{M}_c$ during the deformation.

3.2 Simulation for Soft Robot with Collisions

This section first presents how to formulate the problem of predicting the deformed shape of soft robots under certain actuations as a constrained optimization problem that can be solved by geometric computing. The fast collision detection is enabled by BVH based search. The collision response model will be formulated by adding 'virtual' spring elements into the constrained optimization framework.

3.2.1 Geometry-Based Optimization for Modeling Soft Robots

As illustrated in Fig.3.2, we require an input of two surface meshes $\mathcal{S}_c$ and $\mathcal{S}_b$ that represent the initial shape of a soft robot in the form of its chamber and body, respectively. Volumetric tessellation (function remarked as $\mathcal{T}(\cdot)$) is applied to generate a tetrahedral mesh that interpolated these two surface meshes. Note that the tetrahedral elements are generated between $\mathcal{S}_c$ and $\mathcal{S}_b$ and also inside $\mathcal{S}_c$. $\mathcal{M}$ is segmented into two topologically connected parts $\mathcal{M}_c$ and $\mathcal{M}_b$. Elements in $\mathcal{M}$ are categorized as *chamber element* ($e \in \mathcal{M}_c$, shows in red) and *body element* ($e \in \mathcal{M}_b$, shows in gray). We remark the shape of each element by matrix $\mathbf{V}_e = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4] \in \mathbb{R}^{3\times 4}$, where $\mathbf{v}_i$ gives the position of the i-th vertex.

Without loss of generality, when pneumatic actuation is applied, the deformation of soft robot is driven by the volume expansion in chamber elements. Meanwhile, the shape of body region will be changed as the elements are topologically connected. Here a target volume expansion ratio α is given as the input actuation parameter. In the geometry-based simulation pipeline [100], shapes of the body elements were computed by minimizing the elastic energy E_{ela}. This energy is evaluated by the shape difference between the element's *deformed shape* $\mathbf{V}^d$ and its *target shape* $\mathbf{V}^t$ (details are presented in Chapter 2 and Sec. 3.3.1). Meanwhile, collisions could occur either between different bodies of the soft robot (i.e., self-collision) or between the soft robot and external obstacles.

The optimization problem that computes deformed shape $\mathcal{M}^d$ is formulated as

$$\arg\min_{\mathcal{M}^d} \quad E_{ela} = \sum_{e \in \mathcal{M}_b} Vol(e) \|\mathbf{N}\mathbf{V}^d - \mathbf{R}(\mathbf{N}\mathbf{V}^t)\|^2 \tag{3.1}$$

$$s.t. \quad Vol(\mathcal{M}_c^d) = \alpha \sum_{e \in \mathcal{M}_c} Vol(e) \tag{3.2}$$

$$\mathbf{v} \cap e = \emptyset \ (\forall \mathbf{v} \in \mathcal{S}_c, e \in \mathcal{M}) \tag{3.3}$$

$$\mathcal{M} \cap \mathcal{O}_{obs} = \emptyset \tag{3.4}$$

In the definition of elastic energy (Eq.(3.1)), the matrix $\mathbf{N} = \mathbf{I}_{4\times4} - \frac{1}{4}\mathbf{1}_{4\times4}$ is used to transform the element to the coordinate origin. The rotation matrix $\mathbf{R}$ is computed by *single value decomposition* (SVD) and aligns $\mathbf{V}^d$ and $\mathbf{V}^t$ to the same pose. The operator $Vol(\cdot)$ in the actuation constraint (Eq.(3.2)) computes the volume of a given element. The self-collision-free requirements are formulated as hard constraints (Eq.(3.3)) where each vertex located at the boundary of $\mathcal{M}_b$ (remark as $\mathbf{v} \in \partial\mathcal{M}_b = \mathcal{S}_c$) should not go inside any of the other tetrahedral elements on $\mathcal{M}$. Interaction with the obstacles are handled by the last hard constraint (Eq.(3.4)).

To effectively solve this nonlinear system, we introduce a projection-based local-global solver to **Algorithm 1** in Chapter 2; however, we need to deal with the newly added collision constraints (i.e., Eqs.(3.3) and (3.4)). The details of this solver that incorporates collision response are presented in Sec. 3.3. We first discuss the method for fast collision detection and our collision response model in the following section.

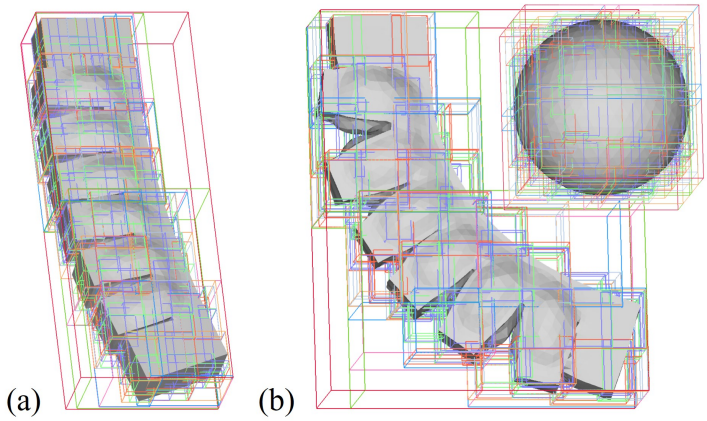


Figure 3.3: (a) AABB trees are built for both soft robots and obstacles for fast collision detection. (b) The structure of BVHs can be re-used when the deformation is relatively small (i.e., only the dimensions of the bounding-boxes need to be updated).

3.2.2 Fast (Self-)Collision Detection with BVHs

For the purpose of collision detection, a brute-force solution is to check all tetrahedra for every vertex on the surface of a soft robot to determine whether any intersection happens. This method, however, introduces a high computational cost when dense meshes are involved. An algorithm based on BVHs is ultilized to accelerate the search by avoiding unnecessary checks of collision pairs during the computation.

First, the *axis-aligned bounding box* (AABB) trees [116] are constructed for both $\mathcal{M}$ and $\mathcal{O}_{obs}$. The bounding-box of a whole model is set as the tree's root, and the nodes in different levels are built to split elements into smaller and smaller bounding boxes until reaching the leaf nodes that only contain one tetrahedron (illustrated in Fig. 3.3). Both the self-collision and obstacle interactions can be efficiently detected using the traversal on the AABB trees, which has the complexity of $\mathcal{O}(n \log(n))$ with n being the number of elements.

AABB trees are employed in our framework instead of other BVHs since the structure of AABB trees can be kept through the deformation while only the geometry of bounding boxes is updated by the new positions of the tetrahedra. The tightness of bounding will become loose after presenting large deformations on the soft bodies. The trees can be re-built when necessary. In all our examples that are presented in this paper, we only rebuild the AABB trees after remeshing. The collision detection can always be efficiently conducted on

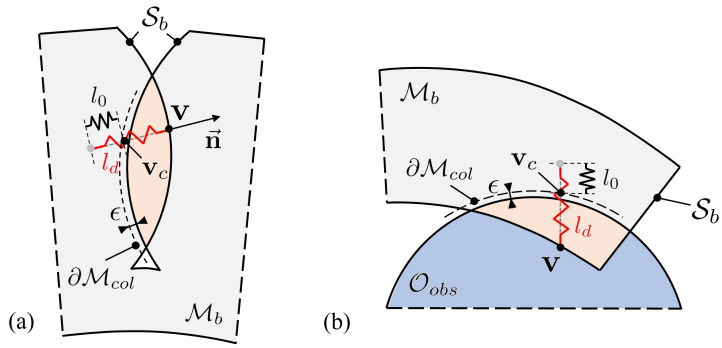


Figure 3.4: An illustration of building 'virtual' spring elements for collision response. The starting point of a spring can be conceptually considered as a point through the direction of $\mathbf{v}\vec{\mathbf{v}}_c$ inside the model. (a) Self-collision happens between two chamber regions that both belong to the body mesh $\mathcal{M}_b$. (b) Interaction between $\mathcal{M}_b$ and $\mathcal{O}_{obs}$.

AABB trees without re-building. Based on our test, (self)-collision detection can be completed at 24fps for a mesh that contains 40k tetrahedra. In each iteration, we stored all collided vertices in the set $\mathcal{V}_{col}$. Statistics regarding the computational time for the collision detection are listed in Sec. 3.4.

3.2.3 Collision Response by Spring Elements

After detecting the collided regions, we built virtual springs on every vertex $\mathbf{v} \in \mathcal{V}_{col}$ for the purpose of collision response. As illustrated in Fig. 3.4, when a vertex $\mathbf{v}$ goes inside the collided region (depicted in orange), we could consider adding a virtual spring between $\mathbf{v}$ and the predicted position $\mathbf{v}_c$ where it should contact on the interface after removing collision. Without loss of generality, we could conceptually assume the spring's length was changed from its resting length l_0 to the deformed status l_d. To compute the collision-free shape using geometric optimization, we allows the 'virtual' spring to drag the collided vertex $\mathbf{v}$ to a collision-free position $\mathbf{v}_c$. This collision response can be formulated as minimizing the spring energy:

$$E_{col} = k(l_d - l_0)^2 = k\|\mathbf{v} - \mathbf{v}_c\|^2. \tag{3.5}$$

When E_{col} is minimized close to zero, the vertex will no longer collide with the model itself or other obstacles.

For the case of self-collision as illustrated in Fig. 3.4(a), the corresponding position $\mathbf{v}_c$ is computed by searching along the inverse normal direction

$-\vec{n}(\mathbf{v})$ from $\mathbf{v}$ until it intersects with the boundary of the collided region (remarked as $\partial\mathcal{M}_{col}$). In the case of robot-obstacle interaction, $\mathbf{v}_c$ is found to be the closest point on $\partial\mathcal{M}_{col}$ (see Fig. 3.4(b)). To avoid numerical errors, $\partial\mathcal{M}_{coll}$ is computed by giving a small offset ϵ to the boundaries of robots and obstacles. The closest point can be efficiently computed with the help of the PQP library [117].

During the iteration of deformation, corresponding points need to be updated accordingly. The spring elements also need to be added or removed throughout the iteration to avoid unrealistic adhesion behavior. In the next section, we will present the details of the adaptive collision response and a local-global solver that minimizes $E_{ela} + E_{col}$ with the actuation constraints (i.e., Eq.(3.2)).

3.3 Projection-Based Solver with Remeshing

By incorporating collision constraints as the spring energy to be minimized, the collision-aware simulation for soft robots can be reformulated as an optimization problem. A local-global iterative solver is used to solve this system. In the local step, the target shapes for chamber elements, body elements, and spring elements are computed using local projection. These target shapes are set to satisfy the corresponding objectives or constraints. In this local step, collision detection is also applied to update spring elements accordingly. After that, a global blending step is applied to assemble all elements together according to their topological connections. Repeatedly applying the local projection and the global blending steps results in the collision-free shape of a soft robot. Meanwhile, progressive remeshing is conducted to ensure a stable and realistic result when simulating soft robots with very large chamber deformation.

3.3.1 Computing Target Shapes by Local Projection

We introduce how the target shapes are computed for different types of elements. For chamber elements with volume variation, their target shape are computed using an average expansion in all verities as illustrated in Fig. 3.5(a) and is the same as presented in Chapter 2. With actuation parameter α as input, the target position for the i-th vertex in the element is determined as $\mathbf{v}_i^t = \sqrt[3]{\alpha}\mathbf{v}_i$. This update ensures that the volume of the chamber region is updated to α times its initial status to satisfy constraint (Eq.(3.2)).

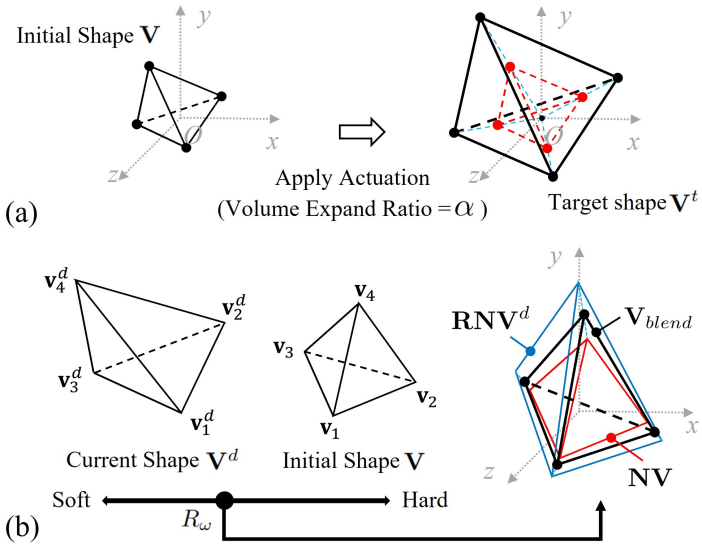


Figure 3.5: Computing the target shapes for (a) a chamber element with given actuation parameter α and (b) a body element ($\mathbf{v} \in \mathcal{M}_b$) by incorporating material parameter R_ω.

Body elements with hyperelastic material properties

The property of soft material needs to be integrated when computing the target shape for body elements. We modify the scheme of shape blending presented in Chapter 2.3 by using the strain-related stiffness parameter to mimic the behavior of hyperelasticity in soft material. As illustrated in Fig. 3.5(b), the projected shape is first computed by combing the initial shape and the current shape of an element. The target shape is then computed by

$$\mathbf{V}_{blend} = (1 - R_\omega(\varepsilon))\mathbf{RNV}^d + R_\omega(\varepsilon)\mathbf{NV}, \tag{3.6}$$

$$\mathbf{V}^t = \frac{Vol(\mathbf{V}^t)}{Vol(\mathbf{V}_{blend})}\mathbf{V}_{blend}. \tag{3.7}$$

Here, the parameter R_ω is a function of the strain ε (detail presented in [115]) and can be calibrated from physical test. Meanwhile, to ensure the incompressibility of hyperelastic materials (i.e., with Poisson's ratio around 0.5), volume scaling is applied to this element. In our experiments, two silicon rubbers – Ecoflex 00-30 and Dargon Skin 20A – are modeled since they are widely used in fabricating soft robots [90, 89, 99]. Detailed data for these materials is provided in Sec. 3.4.

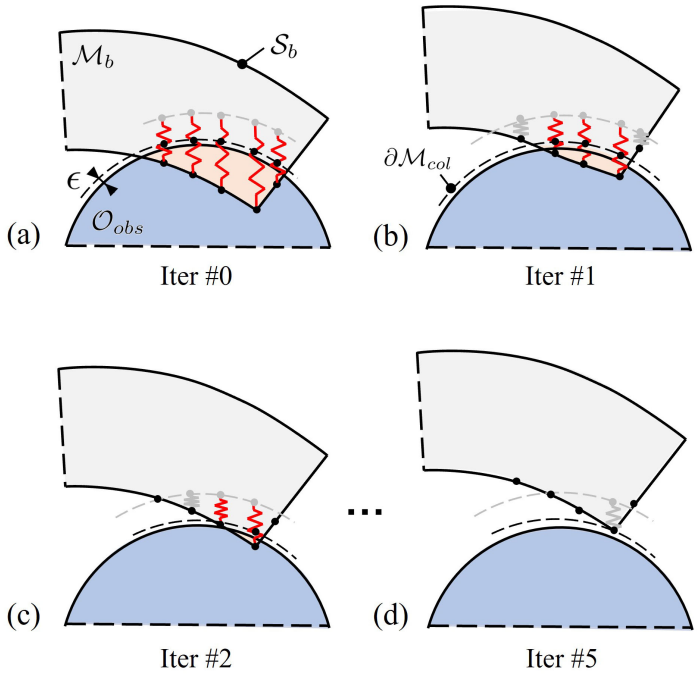


Figure 3.6: Iterative updating of spring elements and their correspondences $\mathbf{v}_c$ in collision response. (a) The initial status contains spring elements that are added to detected collision vertices. (b, c) Through the iteration, the spring elements are deactivated (depicted in gray) when $\mathbf{v}$ has been pulled out of the collided region $\mathcal{M}_{col}$. (d) The final collision-free status with all spring elements released.

Spring elements with updated correspondences

As explained in Sec. 3.2.3, the target shape of a spring element is set as the spring's initial status (i.e., letting the collided point $\mathbf{v}$ move to its corresponding point $\mathbf{v}_c$). In each iteration, the target shape for a spring element needs to be updated together with collision detection. This process is explained and illustrated in Fig. 3.6. An existing spring element will be released if the vertex $\mathbf{v}$ has been outside the collided region and is away from the boundary of $\mathcal{M}_{col}$:

$$\mathbf{v} \notin \mathcal{O}_{obs}, \ ||\vec{\mathbf{n}} \cdot (\mathbf{v} - \mathbf{v}_c)|| > \epsilon. \tag{3.8}$$

For example, springs depicted in Fig. 3.6(b) with a gray color are to be removed in the next iteration since they have satisfied the above mentioned condition (Eq.(3.8)). The iteration of collision response would terminate when all the springs are released (i.e., $\mathcal{V}_{col} = \emptyset$; see Fig. 3.6(d) for an example).

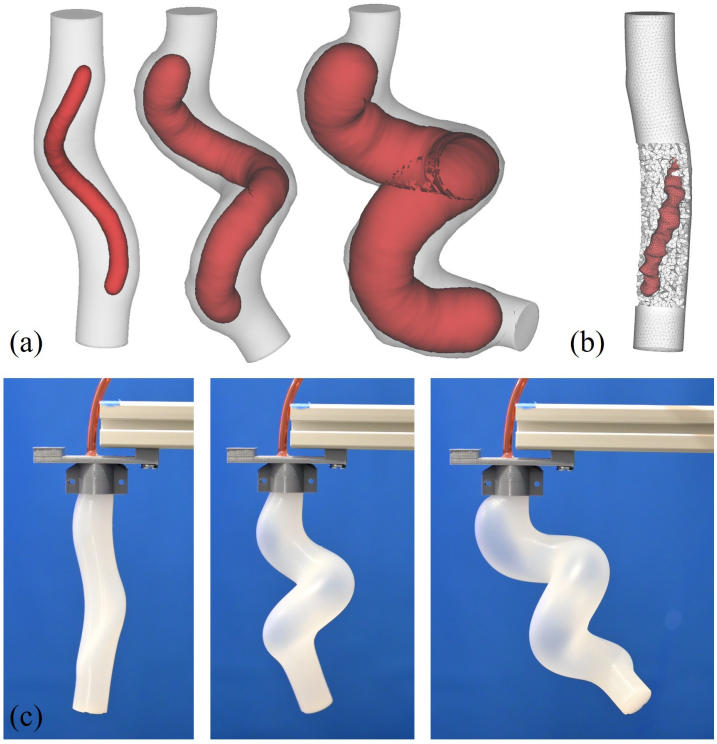


Figure 3.7: The simulation results for a soft robot with twisting deformation under actuation. (a) By progressively remeshing the chamber region $\mathcal{M}_c$ (depicted in red), our method can generate the results that mimic the physical behavior portrayed in (c) as applying different volume-based actuation parameters (from left to right): $\alpha = 5, 28$, and 115. (b) Directly applying a large actuation parameter $\alpha = 28$ leads to an unrealistic result in simulation.

3.3.2 Local-Global Solver with Progressive Remeshing

After defining the target shapes for all elements, the optimization problem is transformed into an augmented form that minimizes the energy

$$\sum_{e \in \mathcal{M}_b, \mathcal{M}_c} \omega_e ||\mathbf{N}\mathbf{V}^d - \mathbf{R}(\mathbf{N}\mathbf{V}^t)||^2 + \sum_{\mathbf{v} \in \mathcal{V}_{col}} ||\mathbf{v} - \mathbf{v}_c||^2. \tag{3.9}$$

By adopting the strategy of local-global solver, the variables $\mathbf{V}^t$, $\mathbf{R}$, and $\mathbf{v}_c$ are computed in the *local step* by shape projection. Those variables are fixed in the *global step*. Therefore, the energy minimization problem defined in Eq.(3.9) becomes a least-square problem since only the position of vertices in $\{\mathbf{V}^d\}$ are

unknown variables. The local and global steps are applied in an iterative manner, and it has been proved that this solver can effectively minimize energy in a few steps. In our implementation, the Anderson-Acceleration method [118] is applied to further improve the converging speed of this iterative-based numerical solver.

On the other hand, we find that unrealistic results could be obtained when large volume-variation-based actuation is direct applied in the simulation. An example is illustrated in Fig. 3.7(b). This is caused by a very sparse density of elements in the region of chamber $\mathcal{M}_c$ in the robot's initial shape. To tackle this issue, we progressively apply actuation and remesh the chamber region $\mathcal{M}_c = \mathcal{T}(\mathcal{S}_c^d)$ when necessary. Specifically after deforming a soft robot, we apply the remeshing step if any of the chamber elements is detected as either

1. having $Vol(\mathbf{V}^d)/\,Vol(\mathbf{V}) > \alpha_{max}$ as a significant volume change, or
2. with $norm(\mathbf{\Sigma}) > d_{max}$ reflecting tremendous distortion in its shape.

Here $\mathbf{\Sigma}$ is the diagonal scaling matrix that is obtained using the SVD of the affine transformation matrix from $\mathbf{V}^d$ to $\mathbf{V}$. The threshold values $\alpha_{max} = 4.0$ and $d_{max} = 4.0$ are selected based on the experiments.

An example is presented in Fig. 3.7 to demonstrate the importance and effectiveness of this remeshing step: a pneumatic-driven twisting soft robot can have the chamber region expanded more than 100 times compared to its initial volume. By applying progressive remeshing, the simulation result of a twisting soft robot can realistically match with the result of physical experiments (illustrated in Fig. 3.7(c)). The local-global solver is illustrated in **Algorithm 2**, in which the remeshing and collision response are presented.

Algorithm 2: Collision-Aware Soft Robot Simulation

Input: Soft robot model $\mathcal{S}_c$ and $\mathcal{S}_b$, actuation parameter α, and obstacle model $\mathcal{O}_{obs}$.

Output: Deformed collision-free soft robot body shape $\mathcal{S}_b$.

```
Generate FE model M = M_c + M_b by T(S_c, S_b);
k = 0, α_k = α_max, M_c^k = M_c;
while Vol(M_c^k) < α Vol(M_c) do
    /* Collision-aware local-global solver */
    while V_coll ≠ ∅ and i < i_max do
        V^t, R, v_c ← local-step(V_coll, M^i, α_k);
        M^{i+1} ← global-step(V^t, R, v_c);
        update V_coll ← CollisionCheck(M^i, O_obs);
    end
    /* Check if remeshing needed */
    for e ∈ M_c do
        if Vol(e_{k+1})/Vol(e_k) > α_max or Σ(e) > d_max then
            Remesh and update M^k = M_b^k + T(S_c^k);
        end
    end
    k = k + 1, and α_k = kα_max;
end
return S_b = ∂M_b^k;
```

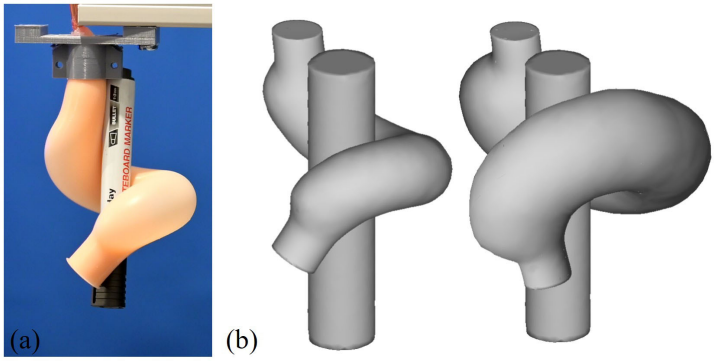


Figure 3.8: A case study that is conducted using a twisting soft robot. (a) The physical experiment test the twisting soft robot to hold a cylindrical object, and (b) our simulator can successfully model the collision behavior when the obstacle is added and when large pneumatic actuation is applied.

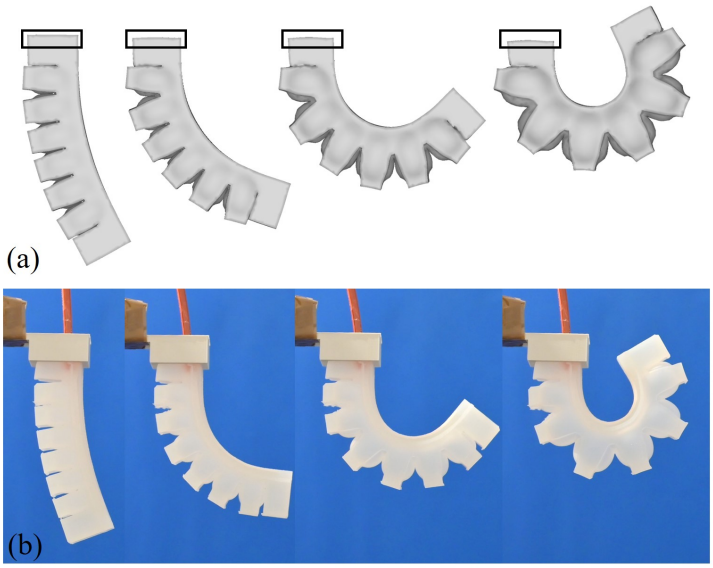


Figure 3.9: (a) The simulation result for the soft finger made by silicone rubber, which can perform large bending deformation. (b) The simulation result can capture the physical behavior.

3.4 Experiment Results

In this section, we demonstrate the behavior of our collision-aware soft robot simulator by presenting several case studies. The proposed computational framework is implemented with C++ and run with a PC with AMD 3950x CPU and 32GB memory. The parallel computing is supported by OpenMP, and the Eigen library is used to solve the linear system. The simulation results were also compared to physical experiments to verify the performance of our approach.

3.4.1 Twisting Soft Robot

The first case study is conducted on a plant-inspired soft robot [99] that can perform twisting deformation. The robot consists of a cylindrical body and a spiral-shape chamber. Eco-Flex 00-30 silicon rubber is used to fabricate the robot, and we model its hyperelastic material property by using first-order Mooney-Rivlin with c_1 = 0.0418 MPa and c_2 = 0.0106 MPa [99]. The simulation results are presented in Fig. 3.7(b) and can mimic the physical behavior

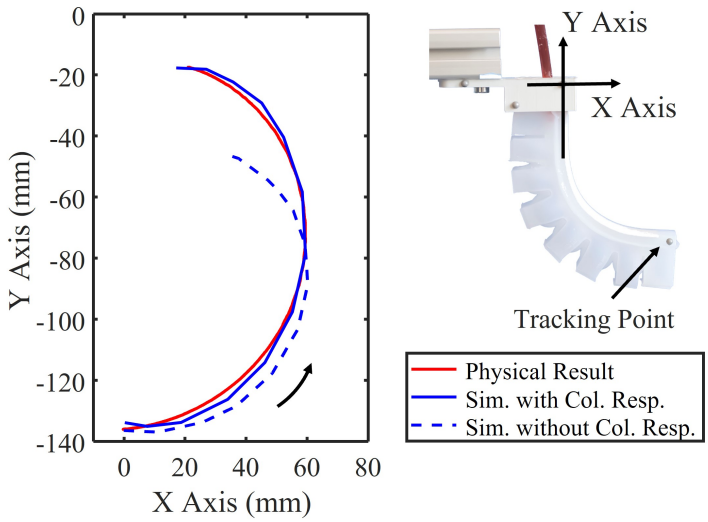


Figure 3.10: The simulator can predict the tip trajectory of the soft finger more precisely after integrating the collision response model.

when remeshing is applied to the chamber region.

This twisting soft robot is able to firmly twine a target object. As depicted in Fig. 3.8, a cylindrical obstacle is tightly grasped when the soft robot is actuated into a shape around the cylinder. This behavior is also successfully simulated using our method: larger actuation is added to the chamber region to produce a tighter grasping actuation (see the right of Fig. 3.8(b)). The mesh model used for simulating this soft robot contains 28.2k tetrahedra. With this mesh density, the simulator could achieve a computing speed of 8.09 seconds per time step. This is around 25 times faster than a nonlinear FEA-based simulation (conducted by Abaqus software) that takes more than 3 minutes on a mesh with a similar density. Commercial software such as Abaqus also suffers from the convergence problem due to the distortion in elements when large inflation is applied.

3.4.2 Soft Finger with Self-Collision

The second case study is conducted on a soft finger model, where both self-collision and robot-obstacle interactions occur at the same time. The result of free-bending that considers self-collision between chambers is presented in Fig. 3.9, and the comparison on the tracked tip position can be found in

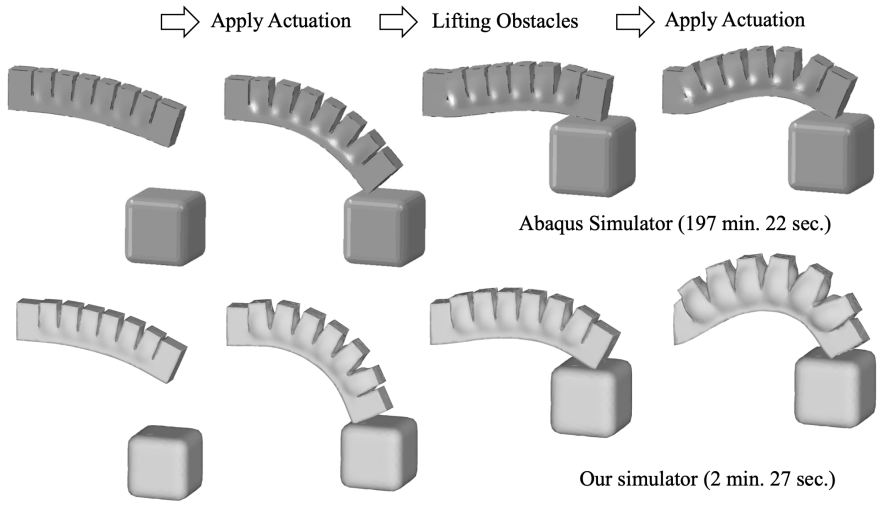


Figure 3.11: The behavior of a soft gripper that is first actuated and then interacts with the obstacle. Both robot-robot and robot-obstacle collision responses are included in the simulation. Our method demonstrates a faster computing speed while maintaining heave good convergence when large deformations happen.

Fig. 3.10. Without an effective collision response, the simulator presented in Chapter 2 cannot precisely mimic the behavior of a soft finger (i.e., large tracking error occurs in the tip trajectory). Another result of simulating the behavior of a soft finger when both robot-robot and robot-obstacle interactions happen together during the movement is presented in Fig. 3.11.

We also use two soft materials, Eco-flex 0030 and Dargon Skin 20, to fabricate two soft fingers for comparison. The Dargon Skin 20 material is around 3 times stiffer, and the Mooney–Rivlin model with $c_1 = 0.119$ MPa, $c_2 = 0.023$ MPa is used. As depicted in Fig. 3.12, two soft robots perform differently when interacting with obstacles. With a softer material, the contact region becomes larger, and the resultant shape is more around the obstacle's boundary. This phenomenon of different materials is well captured by our simulator with the incorporated hyperelastic material model.

3.4.3 Soft Membrane Driven by Interactions

To verify the behavior in interactions between multiple soft robots, the proposed method is applied to simulate the behavior of a soft membrane setup as illustrated in Fig. 3.13. The membrane can be deformed to a free-form shape

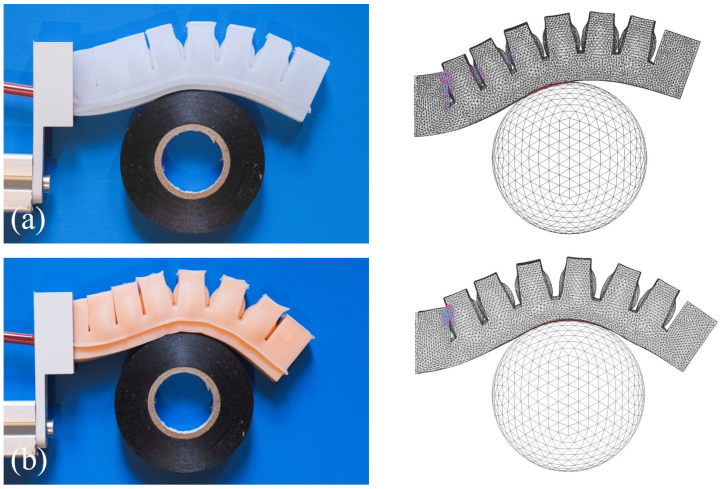


Figure 3.12: The behavior of a soft gripper while grasping an obstacle with a round shape. The soft grippers were fabricated using different types of silicon rubber, such as (a) Dragon Skin 20 and (b) Eco-flex 00-30. Our simulator results (depicted on the right) successfully predict both behaviors with corresponding material properties.

through interactions with four pneumatic-driven chambers below it [119]. The size of the setup is 250×250 mm, which is fabricated using the Dragon skin 20. Our model precisely capture the shape of the outer membrane according to the given actuation parameters applied to the inner chamber array. The predicted shape is compared to the shape captured from physical setup by motion capture system. The maximum error is 5.0 mm as shown in Fig. 3.13, which is 2% of the hardware setup's size.

3.4.4 Statistics Analysis of Computing Speed

The statistics of computing costs are listed in Table 3.1. In general, the computing speed of our simulator is around 25 times faster than the FEA-based simulation on the mesh models with similar density. In comparison to with other reduced FEA-based numerical models (e.g., SOFA [105]) that have faster computing speeds, our method is more robust when large nonlinear deformations happen, which guarantees converging with a realistic simulation result. In all steps of our simulation, solving the optimization problem Eq.(3.9) takes more than 45% of the simulation time. Collision detection is run in real time (i.e., less than 0.12 seconds for all cases), and collision response consumes 25% of the total time. The time used to generate tetrahedral mesh can be fur-

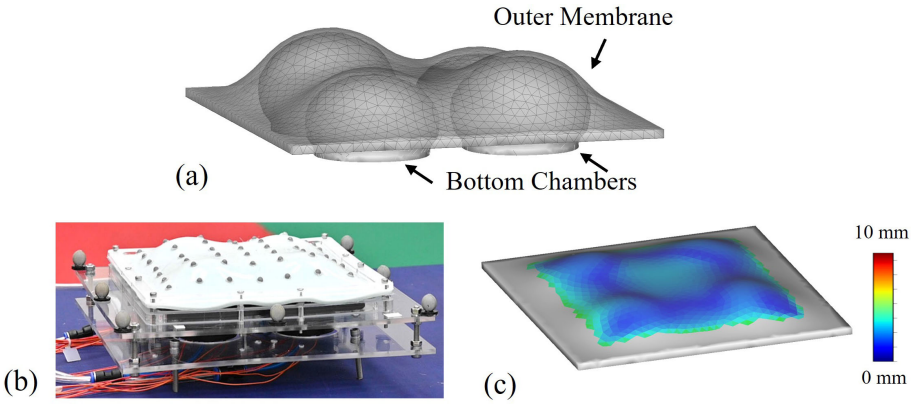


Figure 3.13: Soft membrane driven by interactions with chambers at the bottom: (a) simulation result, (b) physical result on hardware, and (c) quantities analysis of the shape estimation error.

ther reduced by utilizing a more efficient method for volume tessellation [120].

Table 3.1: Computational cost of the collision-aware simulator†

Models	Tet. #	Mesh Gener.	Col. Check	Col. Resp.	Iter. Solver
Soft Finger	79,048	3.54 s	0.07 s	4.38 s	7.46 s
Twisting Soft Robot	28,198	2.81 s	0.03 s	1.71 s	3.54 s
Soft Membrane	34,680*	0.96 s	0.12 s	1.63 s	3.83 s

†The computing time is for single step with given actuation input.
*Contains four chamber meshes; each one contains 8,670 tetrahedrons.

3.5 Conclusion and Discussion

In this chapter, the previously developed fast simulator is enhanced through the addition of functions collision checking and response. In comparison to existing FEA-based approaches, the proposed method demonstrates a much faster computational speed and very robust convergence with complex deformation in a robot's body shape (e.g., expanding, twisting, and bending). Both self-collision and robot-obstacle interactions are successfully modeled. We have tested the simulator on a variety of soft robot systems that are fabricated from different materials with hyperelastic properties; the results accurately simulate the physical deformation of soft robots.

The major limitation of the proposed collision-aware simulator is that the prediction of contact force is not included in the proposed collision-response model. Only friction-less contact is considered in this work to enable soft robot slides on the contact plan. This limitation and potential future work is discussed in Chapter 6.

The fast numerical simulator developed in Chapter 2 and Chapter 3 provides the solution to the FK computing of soft robots, which can further contribute to solving the IK problem. In the next chapter, an iterative algorithm is developed to compute the IK of soft robots: the simulator is used as a basic function to evaluate task-specific kinematics objectives and the corresponding Jacobian matrix.

4

Inverse Kinematics of Soft Robots: Algorithms and Case Studies

The inverse kinematics (IK) of soft robots is defined as finding proper control parameters in actuator space that drive soft robots to finish tasks conducted in either task space or configuration space. In this chapter, we formulate IK computing as an optimization problem in which task-specific objectives are defined and to be minimized. A gradient-based iterative solver is utilized to effectively find the optimum actuation solutions, and the geometry-based simulator developed in the previous chapter is applied to evaluate the gradient by numerical difference. Two tasks including path following and object grasping are studied on different soft robot setups. The computation results in the virtual environment and physical experiments prove the effectiveness of the proposed method[1].

[1]This chapter is based on papers that have been published as:
Guoxin Fang, Rob B.N. Scharff, and Charlie C.L. Wang, "Controlling Multi-segment Soft Robot for Grasping", *IEEE International Conference on Automation Science and Engineering (CASE)*, 2019, short paper (4 pages);
Guoxin Fang, Christopher-Denny Matte, Rob B.N. Scharff, Tsz-Ho Kwok, and Charlie C.L. Wang, "Kinematics of soft robots by geometric computing", *IEEE Transactions on Robotics*, vol.36, no.4, pp.1272-1286, August 2020.
Note: modification and combination have been made to the original published text.

4.1 Introduction

In Chapter 2 and Chapter 3, we present a fast simulation tool that can model the behavior of soft robot under given actuation parameters. In the applications of soft robots, obtaining the needed actuation by the given deformed shape is also demanded. This is an inverse kinematic problem to be solved in this chapter. To better formulate the problem, we first remark that the deformed shape of the soft robot body as the output of an implicit function of forward kinematics as $\mathcal{M}(\mathbf{c}) = f_{fk}(\mathbf{c})$. Here, $\mathbf{c}$ is the input actuation parameter.

For articulated robots, IK is generally described as calculating joint status that makes the end-effector reach its target pose (remark as $\mathbf{v}^t$). Unlike low DOFs articulated robots where analytical IK can be obtained, IK of soft robots cannot be directly calculated as $\mathbf{c} = f_{fk}^{-1}(\mathbf{v}^t)$. Moreover, as discussed in Chapter 1, for the application where the soft robot shape needs to be controlled, IK can also be defined as finding the proper parameters of actuation to control the *configuration space* with large/infinite DoFs. Therefore, IK of soft robots needs to be solved via numerical computation (ref. [121]), and is formulated as optimizing task-specific kinematic objectives.

In this chapter, a gradient-based iterative solver is first developed to compute proper actuation parameters as the solution of IK problem. After that, two case studies that include trajectory following and pick-and-place are presented to verify the effectiveness of the IK solver.

4.2 Algorithm for Gradient-Based IK Computing

We remark the objective of kinematics as $\mathcal{O}(\mathbf{c})$, which is a function of $\mathbf{c}$ and computed with the pose of end-effector or the entire configuration space (i.e., shape of robot body). Without lost of generality, we consider the second case where the target shape is given as $\mathcal{M}^t$. In this case, the objective is written as the difference between two shapes as $\mathcal{O}(\mathbf{c}) = \mathrm{diff}(\mathcal{M}^t, f_{fk}(\mathbf{c}))$. The IK solution of a soft robot can be computed by optimizing $\mathcal{O}(\mathbf{c})$ as

$$\mathbf{c}_{opt} = \underset{\mathbf{c}=\{c_1,c_2,\ldots,c_n\}}{\mathrm{argmin}} \quad \mathcal{O}(\mathbf{c}). \tag{4.1}$$

We use the gradient-based method to solve this optimization task, which needs to first figure out the gradients of objective function with respect to $\mathbf{c}$. The analytical solution of $\frac{\partial \mathcal{O}}{\partial c_i}$ cannot be obtained as the forward kinematics

function $f_{fk}(\mathbf{c})$ is only an implicit function of $\mathbf{c}$. Fortunately, we can efficiently and effectively evaluate the value of $f_{fk}(\cdot)$ by our fast simulator – i.e., we can easily get the status of investigated vertices in regions-of-interest or deformed soft robot shape according to the given actuation. As a result, numerical differences are employed to compute the gradient $\nabla\mathcal{O} = \left[\frac{\partial\mathcal{O}}{\partial c_i}\right]$ as

$$\frac{\partial\mathcal{O}}{\partial c_i} = \frac{\mathcal{O}(\ldots, c_i + \Delta c, \ldots) - \mathcal{O}(\ldots, c_i - \Delta c, \ldots)}{2\Delta c} \tag{4.2}$$

where Δc is a small constant that can be determined according to the value of objective functions by using the strategy from [122].

Directly updating the values of $\{c_i\}$ by using the gradient $\nabla\mathcal{O}$ may lead to a computation with slow convergence. To improve this, we apply a linear search method to determine the best updating scale h so that

$$h = \arg\min \mathcal{O}(\mathbf{c} + h\nabla\mathcal{O}). \tag{4.3}$$

Specifically, we first determine a value of Δh so that $\mathcal{O}(\mathbf{c} + h\nabla\mathcal{O}) < \mathcal{O}(\mathbf{c})$ by a *shrinking* step starting from $\Delta h = 1.0$. The shrinkage speed is controlled with a ratio $\tau \in (0, 1)$; we use $\tau = 0.1$ in all our experimental tests. After that, the scale h is further optimized by being incrementally enlarged with the step size of Δh (this is called an *expanding* step). These two steps of linear search help us find a 'loose' optimum along the direction of $\nabla\mathcal{O}$.

The terminal condition of optimization process for solving (4.1) is chosen to be $\mathcal{O}(\mathbf{c}, \mathcal{M}^t) \leq \lambda$ with λ being a threshold according to the accuracy allowed in different applications. However, it is also possible that a user-specified goal cannot be realized by a soft robot (e.g. when a desired position $\mathbf{v}^t$ falls outside the reachable space of a robot; detail presented in Sec. 4.3.1). Therefore, we also set a maximally allowed iteration number, i_{max}, as another terminal condition in our IK algorithm. Since the line-search strategy is used to ensure the decrease of an objective function in every iteration, our method can always provide a local optimum for objective function. The pseudo-code of the IK computation is provided in **Algorithm 3**.

To verify the accuracy and efficiency of the IK solver in soft robot applications, piratical tasks that includes trajectory following and pick-and-place are demonstrated in the following sections. Objective functions are specifically defined for each task. Notice that all the algorithms with IK computing are implemented in C++ and tested on a standard PC with a 6-core Intel i7 2.4GHz CPU and 16GB RAM.

Algorithm 3: InverseKinematicComp

Input: The rest shape $\mathcal{M}$, the target shape $\mathcal{M}^t$, and the maximally allowed iterations i_{max}.

Output: The actuation parameters $\mathbf{c}$

1 Set the initial actuation parameters as the rest configuration $\mathbf{c} = \mathbf{c}_0$;
2 Set the iteration time $i = 1$;
3 **while** $\mathcal{O}(\mathbf{c}, \mathcal{M}^t) > \lambda$ *and* $i < i_{max}$ **do**
4 Compute the gradient of objective function as $\nabla\mathcal{O}$;
5 Set the step size $\Delta h = 1.0$;
6 Compute $\mathcal{O}_{new} = \mathcal{O}(\mathbf{c} + \Delta h \nabla\mathcal{O})$;
/* Soft line-search (line 7-17) */
/* Step 1: Shrinking */
7 **while** $\mathcal{O}_{new} \geq \mathcal{O}_i$ **do**
8 $\Delta h = \tau \Delta h$;
9 Compute and update $\mathcal{O}_{new} = \mathcal{O}(\mathbf{c} + \Delta h \nabla\mathcal{O})$;
10 **end**
/* Step 2: Expanding */
11 Set $h = \Delta h$;
12 **repeat**
13 Compute and update $\mathcal{O}_{new} = J(\mathbf{c} + (h + \Delta h)\nabla\mathcal{O})$;
14 **if** $\mathcal{O}_{new} \leq \mathcal{O}_{opt}$ **then**
15 Set $\mathcal{O}_{opt} = \mathcal{O}_{new}$ and $h = h + \Delta h$;
16 **end**
17 **until** $\mathcal{O}_{new} > \mathcal{O}_{opt}$;
/* Best h has been found */
18 Set $\mathbf{c} = \mathbf{c} + h\nabla J$ and $i = i + 1$;
19 **end**
20 **return** $\mathbf{c}$;

4.3 Case Study I: Trajectory Following

We first demonstrate the behavior of the IK algorithm in the trajectory planning task. Given the desired motion trajectory $\mathcal{L}$ for an investigated point $\mathbf{p}$ on the soft robot $\mathcal{M}$, the task of trajectory planning can be solved by finding the parameters of actuation that drive $\mathbf{p}$ traveling along $\mathcal{L}$ accurately. To realize this, we first sample N points on $\mathcal{L}$ and category them as $\mathcal{P}_{\mathcal{L}} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$. After running the IK computation presented in **Algorithm 3** for each target point $\mathbf{p}_k \in \mathcal{P}_{\mathcal{L}}$, we are able to determine the corresponding parameter set $\mathcal{C}_k$ in joint space for actuation (i.e., the shortening ratio for each tendon's length). For the trajectory following case, the kinematics-related objective is defined as

$$\mathcal{O}(\mathbf{c}, \mathbf{p}^t) = \|\mathbf{p}_c - \mathbf{p}^t\|^2, \quad \mathbf{p}_c \in \mathcal{M}(\mathbf{c}) \tag{4.4}$$

which measures the distance between the current position of the end-effector $\mathbf{p}_c$ and the target position of the waypoint $\mathbf{p}^t$. For the terminal condition $\mathcal{O}(\mathbf{c}, \mathbf{p}^t) \leq \lambda$ that is used for IK computation, we choose $\lambda = 0.2$ mm and $i_{max} = 30$ for all waypoints.

4.3.1 Cable-Driven Soft Finger

The first experiment is conducted using a cable-driven soft finger with three knuckles. The soft finger is fixed on a solid base in our experimental setup (as illustrated in Fig. 4.1(a)) and for every 'knuckle', one iron cable is linked to its top and driven individually by its corresponding stepper motor through the pulley shaft. The design with multiple actuators enables the ability to control the soft finger to move in a plane.

Implementation Details: After using different initial guesses for the IK computation, we find that its speed of convergence strongly relies on the position of initial guess (see Fig. 4.1(c)). Therefore, the following two strategies are conducted to speed up the computation in our trajectory planning application.

- First, we generate a sample-based representation for the configuration space $\mathcal{P}_\omega$ (see Fig. 4.1(b)) where the sample points $\mathbf{p}_c \in \mathcal{P}_w$ are obtained by applying the FK algorithm with various combination of actuation parameters. The initial guess of IK solution (i.e. $\mathbf{q}$ in **Algorithm 3**) for the first point $\mathbf{p}_1$ on a trajectory $\mathcal{L}$ is then set as the control parameter of its closest sample point in $\mathcal{P}_w$.
- Second, a deformed shape is always kept during the computation and serves as the initial shape for realizing the next target point. Specifi-

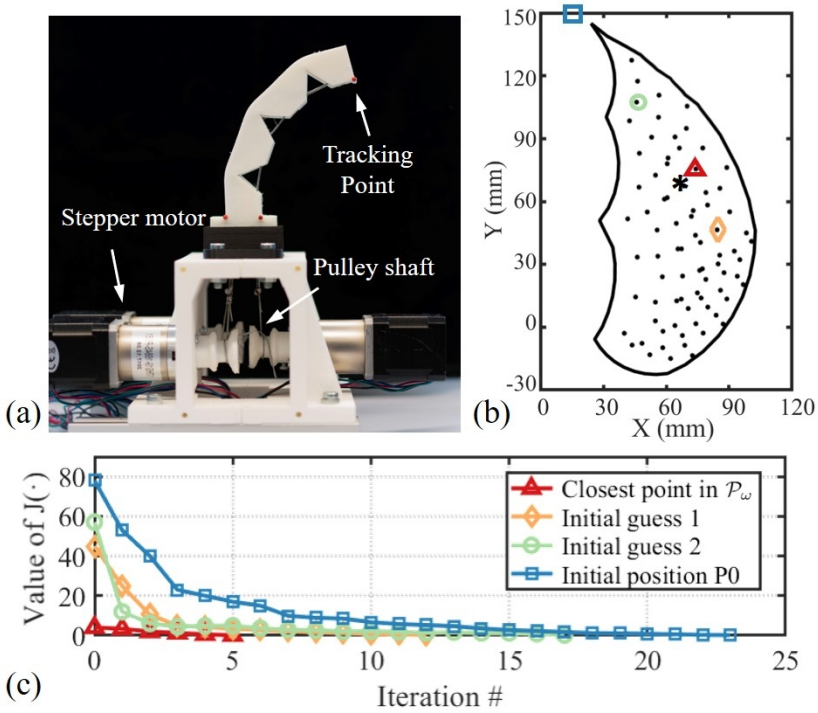


Figure 4.1: A cable-driven soft finger with three tendons is used for the validation of IK computation. (a) The experiment setup. (b) The configuration space is sampled to obtain good initial values for IK computing (125 sample points are displayed). (c) The study of convergence for IK computation by evaluating the objective function $E_{IK}(\cdot)$ (Eq.(4.4)) with the target position of the tracking point being given as the black star shown in (b). We find that the converging speed of IK computation greatly improved when the closest point (red triangle) in the sampled configuration space is used as the initial guess.

cally, after obtaining the actuation parameters $\mathbf{c}_k$ for the target point $\mathbf{p}_k$, we update the shape of soft robot, $\mathcal{M}(\mathbf{c}_k)$, by applying the forward kinematics with $\mathbf{c}_k$ as the input. This updated shape will be used as the input for IK computation targeting on the next point $\mathbf{p}_{k+1}$.

The pseudo-code of the trajectory following algorithm and the above acceleration strategies are provided in **Algorithm 4**. The computation of this method is very efficient. First, a roughly sampled configuration space (e.g. 125 sample points) can be generated in 8.3 sec. to obtain good initial values. Then, we conducted the tests on a 'L' trajectory (with $N = 55$) and a flame trajectory (with $N = 85$) as shown in Fig. 4.2, the computing times are 38 sec.

Algorithm 4: TrajectoryFollowing

Input: The rest shape $\mathcal{V}$, the sampled working space $\mathcal{P}_w$, the investigated point $\mathbf{q}$ and the target trajectory $\mathcal{L}$.

Result: The parameters of actuation $\mathcal{C}$.

```
Generate the set of sample points P_L = {p_1, p_2, ..., p_N} on L;
Find the point p_c ∈ P_w that minimizes ||p_c − p_1||;
/* Using the corresponding C_c of p_c                    */
V^d ← ForwardKinematicComp(V, C_c);
/* Computing the actuation parameters                    */
for k = 1, 2 ... N do
    /* Accelerate the IK computation by using V^d
       and C_c as initial guess                          */
    C_k ← InverseKinematicComp(V, p_k);
    V^d ← ForwardKinematicComp(V, C_k) and C_c ← C_k;
end
return C;
```

and 46 sec. respectively.

We further conduct physical experiments (see the hardware setup illustrated in Fig. 4.1(a)) to verify the parameters of actuation generated by **Algorithm 4**. Arduino Mega 2560 and the RAMPS extension board are used to generate the modulated pulse signals that control the pull and release of cables. To generate a motion that linearly interpolates the neighboring target points, dynamic speed controller provided by Marlin firmware[123] is used to synchronize the three motors. A camera system is used to track the actual position of the investigated point $\mathbf{q}$, which is located at the top-right corner of the soft finger. The resultant trajectories of physical movement are given in Fig. 4.2(a), which are compared to the target trajectories. The errors of motion are also visualized as two error curves in Fig. 4.2(b). Besides computation error, errors in physical experiments are also generated by many other factors, including the fabrication error, the control strategy and the unpredictable friction between cables and the soft finger.

By sampling the configuration space for a soft robot, one intuitive solution of trajectory following can be realized by directly searching the closest sample points in $\mathcal{P}_\omega$ and using their corresponding parameters for actuation. However, this method needs a very dense sampling rate to guarantee the required numerical accuracy that is comparable with our IK computing. Although our FK

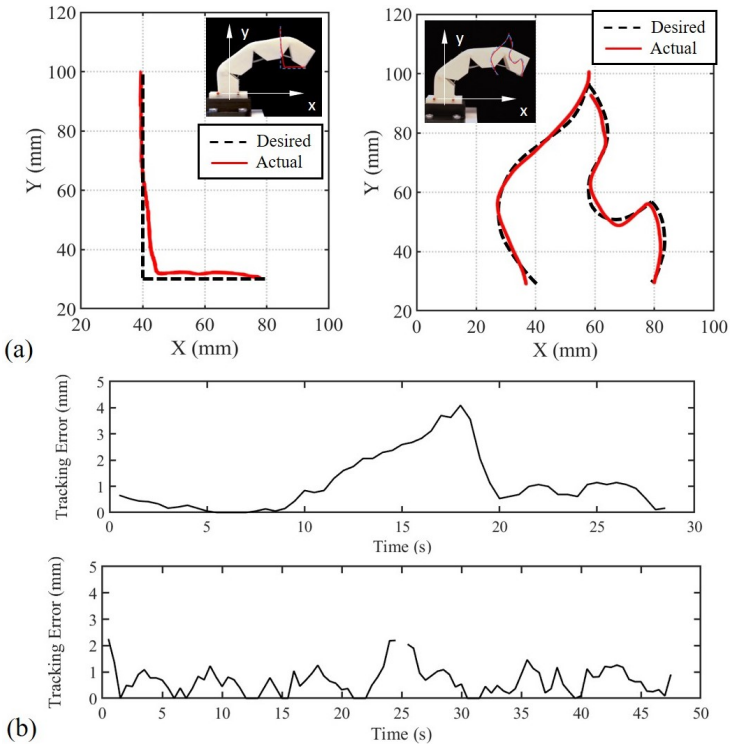


Figure 4.2: The results of experimental tests for moving a marker point along desired trajectories: (a) comparison between the tracked actual movement and two target trajectories, and (b) position errors of the investigated point that occurs while moving along the 'L' trajectory (top) and the flame-shaped curve (bottom). The dimension of the actuator is $120\text{mm} \times 25\text{mm} \times 25\text{mm}$.

computing is very fast, the cost of this searching-based planning is still much higher than the IK-based trajectory following method presented in **Algorithm 4**. For example, computing 3375 sample points beforehand (see Fig. 4.1(b)) takes more than 5 minutes. Moreover, preserving continuity on a path that is realized by the sample-searching method is difficult. Notably, the IK computing presented in **Algorithm 4** can ensure continuity by its nature as an iterative algorithm. We test it on an extreme case as shown in Fig. 4.3 where part of the desired trajectory falls out of the working space. The result of our algorithm is a smooth path that is completely inside the feasible region.

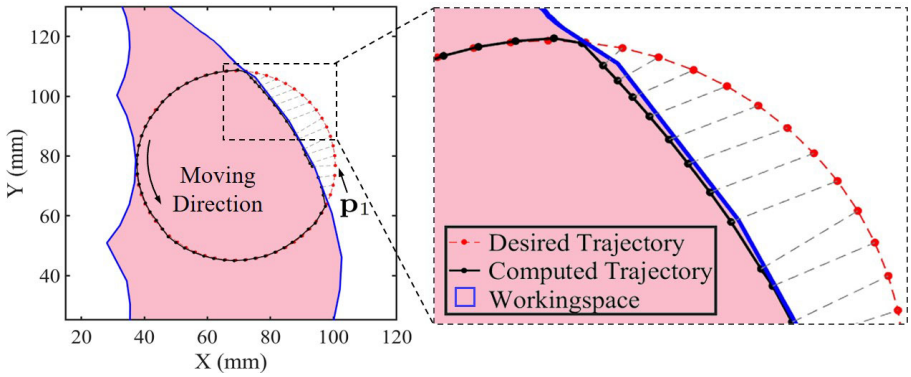


Figure 4.3: The results of IK computing and trajectory following with a desired trajectory $\mathcal{L}$ that is partially out of the soft robot's working space. Waypoints (red) on $\mathcal{L}$ and their corresponding reachable points (black) determined by our IK solution are depicted by the gray dashed lines.

4.3.2 Pneumatic-Driven Multi-Chamber Soft Robot

We also test the IK computing method to finish the trajectory following task on a soft robot driven by multiple pneumatic actuators as illustrated in Fig. 4.4. Efficiently predicting the required pressure that can generate an expected deformation on a pneumatic-driven soft robot can be very challenging when using conventional methods (such as the static force modeling [47] or the FEM analysis [50]) because of the highly non-linearity of the problem. Specifically, the required volume for each chamber for every waypoint is computed using the IK solver and is then converted into the required pressure to be provided by the actuation system.

The result of the trajectory following is illustrated in Fig. 4.5 with the tracked trajectory plotted in the top view while compared with the desired trajectory. By applying the **Algorithm 4** to compute the motion sequence, the end-effector of the soft robot can follow the desired flower-shaped trajectory with an error under 4mm. The average computing time to find the IK solution with a single waypoint is 138 seconds. The high computing cost is caused by the usage of numerical simulation in the iteration loop to evaluate the objective and its gradient. In the next chapter, a learning-based pipeline is utilized to accelerate this to real-time speed with further improved accuracy.

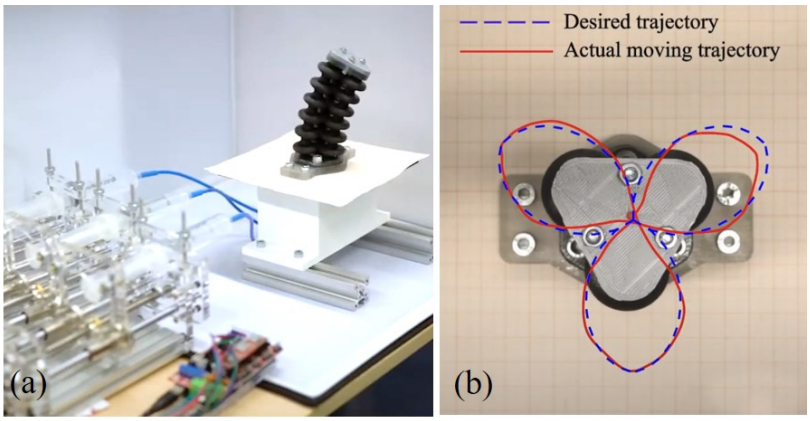


Figure 4.4: (a) The experiment setup with multi-chamber 3D printed pneumatic-driven soft actuators. (b) The top view of the setup and an illustration of the desired trajectory.

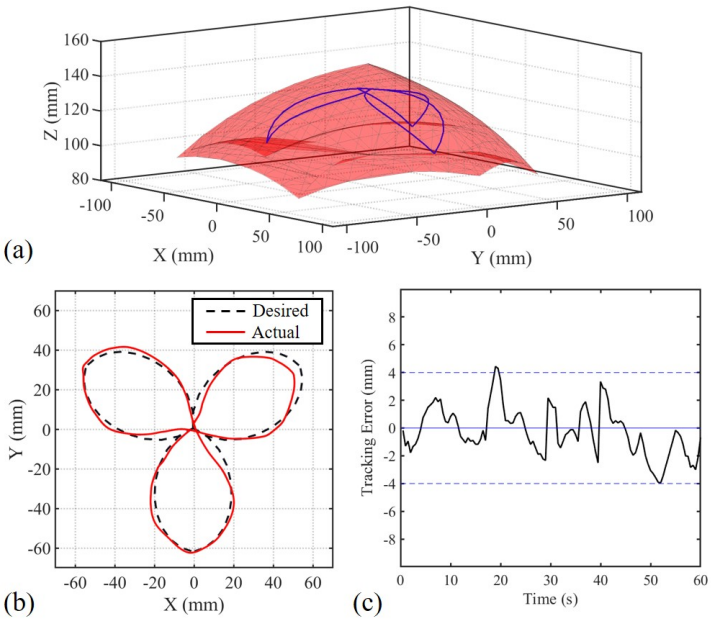


Figure 4.5: Results of trajectory planning on a pneumatic-driven soft robot: (a) Configuration space for the tip point on this robot is presented as the red 3D region, and the blue curve illustrates the target trajectory. (b) Top view of tracked trajectory realized by our IK computation based actuation. (c) The corresponding position error for the tracked tip point.

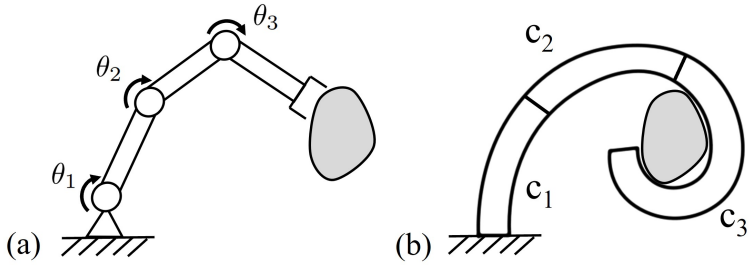


Figure 4.6: An illustration of a grasping task for multi-segment rigid-joint and soft-bodied robots. a) A 3-DoF rigid robot with an end-effector can grasp the target object by controlling the angle θ_i of each joint. b) A three-segment soft robot driven by actuation variables $\{c_i\}$ can bend to complete the grasping task in an adaptive way.

4.4 Case Study II: Pick-and-Place

In robotic research, controlling precise grasping is a fundamental problem that has been widely studied regarding industrial automation [3]. A traditional rigid-joint robot (as illustrated in Fig.4.6(a)) usually realizes the grasping task by moving the manipulator to the desired point and then using a gripper to pick and place the object [124]. By building up the kinematics model analytically, one can effectively compute the control parameter θ_i of each joint using IK, which makes the control problem relatively intuitive. However, soft robot manipulators can achieve better robotic grasping solutions and have already been commercialized by several companies (e.g. FESTO and Soft Robotics Inc.). For this case study, we applied the developed IK solver to compute a feasible actuation parameter for soft robots to accomplish the pick-and-place task. As illustrated in Fig.4.6(b), the motion of the soft robot is controlled by the actuation parameters $\{c_i\}$, where c_i is formulated as the pressure of the inner chamber of the i-th bendable soft segments.

The design of a three-segment soft robot used to conduct pick-and-place task is illustrated in Fig.4.7. This multi-segment soft robot is composed of three pneumatic-driven soft bending actuators that are rigidly connected and moved together. The cross-section drawing of a single actuator presented in the middle figure of Fig.4.7(a) shows that the design includes an inner fluid channel, a soft continuum body, rigid connectors, and a thin inextensible layer. Rigid plugs with inner fluid channels and tube connectors are used to link the soft robot with the actuation system and make sure each gripper can be actuated individually. When applying air pressure, with the help of the teeth-

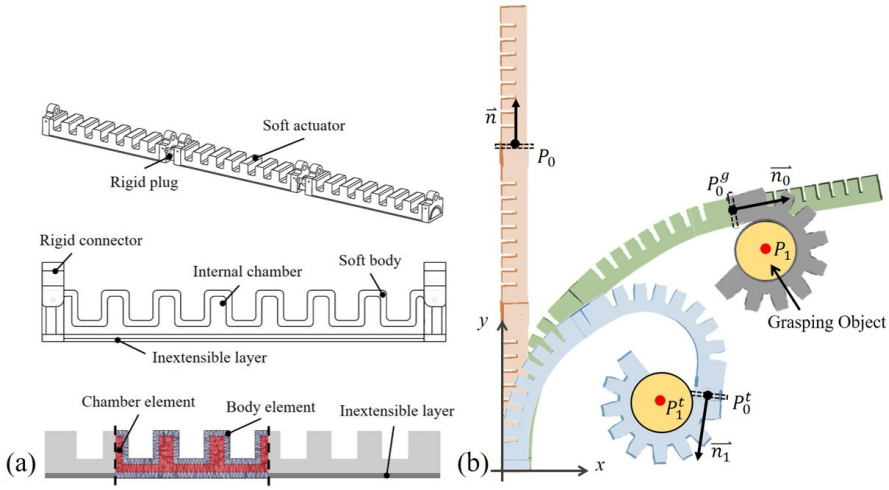


Figure 4.7: (a) Design and structure of the three-segment soft robot manipulator. (b) Illustration of the pick-and-place process. The orange model represents the initial shape, and the green model demonstrates how the end-effector is first positioned. After grasping the object (gray model), the target object is placed in a new position (blue model).

structure design and the inextensible layer, each soft segment can realize a bending deformation of up to 180 degrees. This guarantees that the working space for solving positioning and grasping tasks is large enough. The shape of this soft actuator can be effectively modeled by the geometry-based simulator presented in Chapter 2.

4.4.1 Objectives for Pick-and-Place Task

The pick-and-place task contains three steps as depicted in Fig.4.7(b). The bottom-two segments are first actuated to position the target point P_0 (located at the tip of the second chamber) from its initial position to the desired grasping point P_0^g. Then the end-effector is actuated to realize the desired bending deformation to match the end-effector's curvature with the shape of the grasping object. The object is then dragged and placed into its new position P_1^t together with the soft robot. For each step, corresponding geometry-defined objectives are defined to formulate the optimization problem for IK computing.

We first build the objective function for the positioning step where only the first two grippers are actuated to realize the desired bending deformation. As presented in Fig.4.8, there are two requirements to realizing precise grasping:

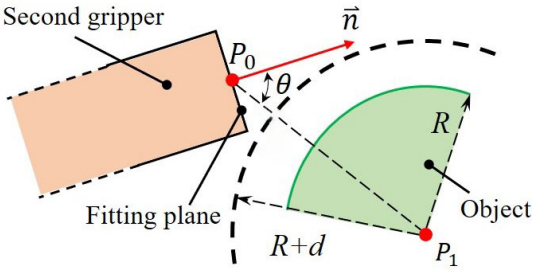
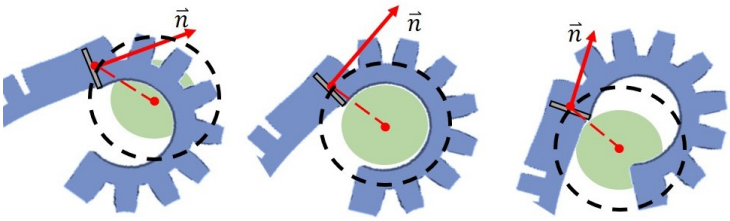


Figure 4.8: A zoom-in view of the positioning process; the first two actuators are inflated to locate the tip point P_0 in the desired position. The normal direction $\vec{n}$ is computed using the second gripper's upper plane.



(a) Self-collision (b) Tight grasping (c) Unstable grasping

Figure 4.9: An illustration of how the angle θ influences the behavior of the grasping task with soft gripper. a) Collision between the manipulator and target object occurs if θ is less than 90 degrees. b) The orthogonality between $\vec{n}$ and $\overrightarrow{P_0P_1}$ is one of the objective functions for robust grasping. c) Obtuse angle θ leads to unstable grasping.

- The target point P_0 needs to be located in a circle boundary which is defined by center position P_1 and diameter $R + d$.
- The normal direction $\vec{n}$ and $\overrightarrow{P_0P_1}$ should be orthogonal (see Fig.4.9 for a detailed explanation).

The parameters P_1 and R are both computed using the center position and diameter of the grasping object's boundary circle. Meanwhile, d is defined as half of the width of the gripper design. Those variables are all treated as input for the inverse control problem.

Since the soft actuator is compliant with the object's shape, there is tolerance for the second normal requirement. However, the position of point P_0 has to be satisfied to grasp the object. Thus, we build a constrained optimization problem to compute proper control parameters $\mathbf{c} = \{c_1, c_2\}$ as:

$$\underset{\mathbf{c}=\{c_1,c_2\}}{\text{argmin}} \quad \mathcal{O}_{pick} = (||\mathbf{p}_1 - \mathbf{p}_0|| - (R+d))^2 + \omega_1(\vec{\mathbf{n}} \cdot (\mathbf{p}_1 - \mathbf{p}_0))^2 \tag{4.5}$$

$$s.t. \quad c_{max} \geq c_1 \geq 1,\ c_{max} \geq c_2 \geq 1 \tag{4.6}$$

where the objective $\mathcal{O}_{pick}$ contains the energy for the control of both tip position and orientation of the end-effector. $\mathbf{p}_0$ and $\mathbf{p}_1$ are the position vector of point P_0 (detected from $\mathcal{M}$) and P_1 (input variables), respectively. Eq.4.6 presents the actuation constraint where we should have avoided negative pressure inside the chamber. Here c_{max} represents the maximum expanding ratio before the chamber starts to leak.

As the second movement, the end-effector begins to bend and grasp the object. By applying the forward kinematic function f_{fk}, the position of vertex set $\mathcal{V}$ located in the inextensible layer can be directly obtained from $\mathcal{M}(\mathbf{c})$. We apply the least-square fitting method to compute the diameter $\mathcal{K} = f_{fit}(\mathcal{V})$ to approximate the end-effector's curvature. The closer this curvature diameter matches with the diameter R, the tighter the grasping behavior become; therefore, we define another objective function:

$$\mathcal{O}_{grasp} = (f_{fit}(\mathcal{V}) - R)^2 \tag{4.7}$$

By minimizing $\mathcal{O}_{grasp}$ as subject to the same actuation constraint in Eq.4.6, the control variables are found for the grasping step.

During the final stage of the pick-and-place task, objective $\mathcal{O}_{place} = (||\mathbf{p}_1^t - \mathbf{p}_0|| - (R+d))^2$ is used as the orientation of the end-effector no longer need to be controlled. Again, the actuation constraint is also applied when computing the kinematics solution.

4.4.2 Experimental Results

The gradient-based IK solver presented in **Algorithm 3** is used to solve the optimization problem for all pick-and-place related objectives. Their gradients are evaluated using the numerical difference, as depicted in Eq. 4.2. The computed actuation parameter is verified on physical setup, where actuators are fabricated with an Objet 350 Connex3 3D printer. Rigid (VeroCyan, σ_r = 46.6 MPa) and soft (Agilus Black, σ_s = 8.2 Mpa) materials are used. The inextensible layer is fabricated using VeroCyan material, whereas the extensible areas are printed using a combination of VeroCyan and Agilus Black with a Shore value of 60A. The soft actuators are pressurized by a syringe

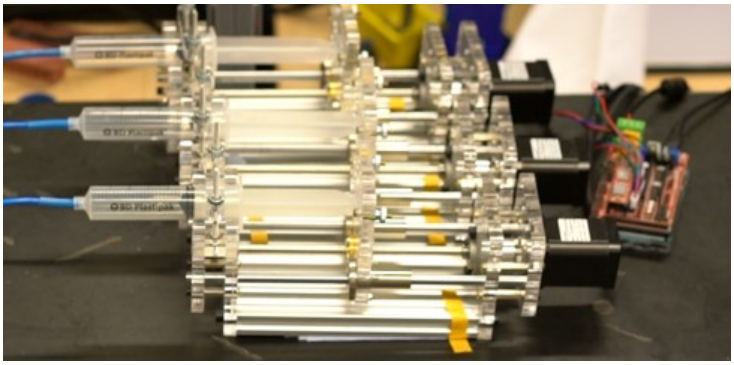


Figure 4.10: The actuation system used in physical experiment. Compressible air in syringes are actuated to the chambers with the motion from linear motors controlled by Arduino.

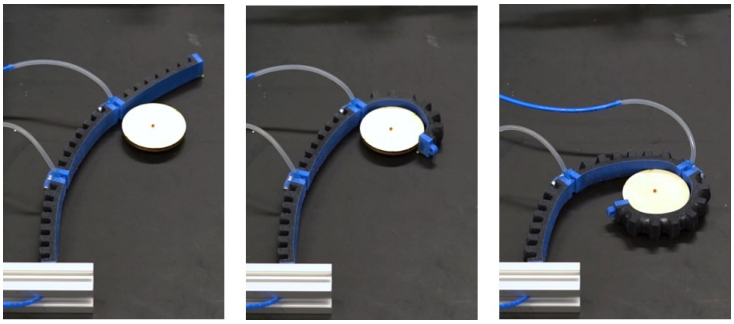


Figure 4.11: Physical experiment for the grasping task. The pictures present the behavior of a 3D printed multi-segment soft robot deformation in different steps, which includes tip position positioning, end-effector grasping, and replacing.

pump module (shown in Fig.4.10) where the moving distance of the syringe plunger is used to control the amount of air that is pumped into the chamber. We successfully complete the three-stage grasping task (positioning, grasping and placing the object) as is shown in Fig. 4.11 and in the video (available at `https://sites.google.com/view/controlsorograsping/`).

4.5 Conclusion and Discussion

In this chapter, IK of soft robot defined in both *task space* and *configuration space* are solved by numerical optimization. Task-specific objectives are defined and minimized using a gradient-based iterative solver. Soft line search is integrated to improve the converging speed of the proposed solver. The geometry-based simulator developed in the previous chapter is applied to effectively evaluate the gradient by numerical difference.

We verify the functionality of developed IK solver by conducting two case studies on different soft robot setups. We first consider the path following tasks for both cable-driven and pneumatic-driven soft robots. An algorithm is developed to accelerate the IK computing speed for a sequence of motion. This algorithm ensures a minimum variation in *actuator space* between neighboring waypoints. The second task is grasping and placing objects with a soft robot manipulator. To conduct the tight grasping action, objectives based on the pose of the end-effector and shape parameters (i.e., curvature) of the robot body are formulated. The IK solver successfully minimizes all geometry-defined objectives, and the computed actuation parameters successfully drive soft robots to finish given tasks.

As the first attempt to solve the IK problem, the computing speed of the proposed method is still not satisfying and cannot support real-time kinematics control for soft robots. This is because the evaluation of objectives and their gradients is based on a numerical simulator. In the next chapter, the speed of IK solver is accelerated using machine learning. On the other hand, the control error in *task space* is relatively high since the IK solutions computed in the virtual space are directly applied to physical setups. A sim-to-real transfer learning pipeline is introduced in the next chapter to eliminate the reality gap and help reduce the effort in generating training data.

5

Efficient Jacobian-Based Inverse Kinematics with Sim-to-Real Transfer of Soft Robots by Learning

This chapter presents an efficient learning-based method for solving the inverse kinematic (IK) problem of soft robots with highly non-linear deformation. The major challenge of efficiently computing IK for such robots is caused by the lack of analytical formulation for FK and IK. To address this challenge, we employ neural networks to learn both the mapping function of forward kinematics and also the Jacobian of this function. As a result, Jacobian-based iteration can be applied to solve the IK problem. A sim-to-real training transfer strategy is conducted to make this approach more practical. We first generate a large number of samples in a simulation environment for learning both the kinematic and the Jacobian networks of a soft robot design. Thereafter, a sim-to-real layer of differentiable neurons is employed to map the results of simulation to the physical hardware, where this sim-to-real layer can be learned from a very limited number of training samples generated on the hardware[1].

[1]This chapter is based on the paper that has been published as: **Guoxin Fang**, Yingjun Tian, Zhi-Xin Yang, Jo M.P. Geraedts, and Charlie C.L. Wang, "Efficient Jacobian-Based Inverse Kinematics with Sim-to-Real Transfer of Soft Robots by Learning", *IEEE/ASME Transactions on Mechatronics (T-MECH), accepted, 2022.* Note: a few modification and combination has been made to the original published text.

5.1 Introduction

With the use of flexible material, soft robots have the ability to make a large deformation and interact safely with the environment [9], which leads to a broad range of applications such as exoskeleton / wearable devices [125], soft manipulators [126, 80] and surgery assistance [127]. However, as a hyper-redundant system with high nonlinearity in both material propriety and geometric deformation, it is difficult to formulate an effective kinematic model for solving the control task. The analytical *forward kinematics* (FK) solution only exists for specific designs with a relatively simple shape (e.g., [57, 58]). For a general soft robot with complicated structures / shapes, efficiently computing its IK solution remains a challenging problem. For soft robots with redundancy, fast and reliable IK solution is a very important means for improving the control precision and response frequency in practical tasks [128].

5.1.1 Related Work

To efficiently model the behavior of soft robotic systems (i.e., computing FK), both analytical formulation and numerical simulation were conducted in previous research. Those analytical solutions, based on differential geometry [57, 58, 129] and mechanics analysis [61], are difficult to be generalized for soft robots with a complex shape, where numerical simulation by the *finite element method* (FEM) is usually employed [130, 131]. Computational efficiency is a bottleneck of applying FEM in the IK computation, as the simulation needs to be repeatedly conducted to estimate the Jacobian [132]. To overcome this, a reduced model by voxel representation [67] or computing quasi-static equilibrium [53] are presented to accelerate. However, these methods can easily become non-realistic after applying large rotational deformation. The geometry-oriented simulation pipeline [13] can precisely compute the deformation of a variety of soft robots even in large rotation, which is later extended into a general IK solver [100] by using the Jacobian-based iteration. A model reduction method is applied to further accelerate the numerical-based simulation [69]. However, it is still difficult to directly include the simulator in the loop of iteration and achieve fast IK computing.

The data-driven methods used in soft robotics are often treated as regression problems of machine learning where kinematic models can be effectively learned from datasets [128]. To enable the inverse kinematic tasks on soft robots, an intuitive solution is to directly learn the mapping of IK which takes the motion as the input of a network and generates the corresponding param-

eters of actuation as output (ref. [41, 133, 134, 42, 135, 136, 43]). However, this intuitive method does not perform well in a redundant system as the one-to-many mapping from task space to actuator space is generally difficult to learn. Although this issue can be partly solved by setting constraints in the actuator space [43] or specifying the preference of configurations in the IK equation [136], we solve the problem by using a different method to combine learning with Jacobian-based IK. Our method is efficient when planning a smooth motion (i.e., by minimizing variation in the actuator space) for soft robot systems with redundancy. To reach a similar goal, Thuruthel *et al.* [137, 138] attempt to learn the differential IK model with local mapping. Another method is presented in [73] to estimate the soft robot's Jacobian by Kalman filter approximation. Recently, Bern *et al.* [139] presented a method to effectively evaluate the Jacobian by using the gradients of the FK network, which however limits the type of network used for FK learning and requires more time to compute the Jacobian for determining IK solutions. In our work, both the mapping functions of FK and the Jacobian are learned by neural networks as explicit functions.

On the other hand, learning a kinematic model for soft robots usually needs a large number of samples, which can be very time-consuming when generating the data in a physical environment either by the motion capture system [80] or embedded sensors [140, 40]. Moreover, to explore the boundary of the work space, a large extension in material under large actuation needs to be applied [141]. Soft materials on a robot can become fragile and might generate plastic deformation after repeating such deformation may times [80]. Consequently, the learned model becomes inaccurate. Furthermore, errors generated during the fabrication of a specimen can make the network learned on this specimen difficult to be used on other specimens with the same design. To reduce the cost of generating training data, Daniel *et al.* reported a data-efficient method by exploiting structural properties of the kinematic mapping [142]. Another solution is to generate accurate dataset in the simulation environment and then convert the model learned from simulation into physical reality by transfer learning (ref. [143, 144, 145, 146]). The hybrid model contains the analytical formulation and the network-based correction is conducted in [147, 136] where more precise control of the soft robot is achieved. Similarly, FEM is used in [148] to generate a simulation dataset for training a hybrid kinematic model by transfer learning. The efficient numerical simulator conducted in the previous chapter [100] is adopted in this work to generate the training dataset. When working together with the sim-to-real network, IK with high accuracy can be achieved. The comparison of IK with sim-to-real learning by using

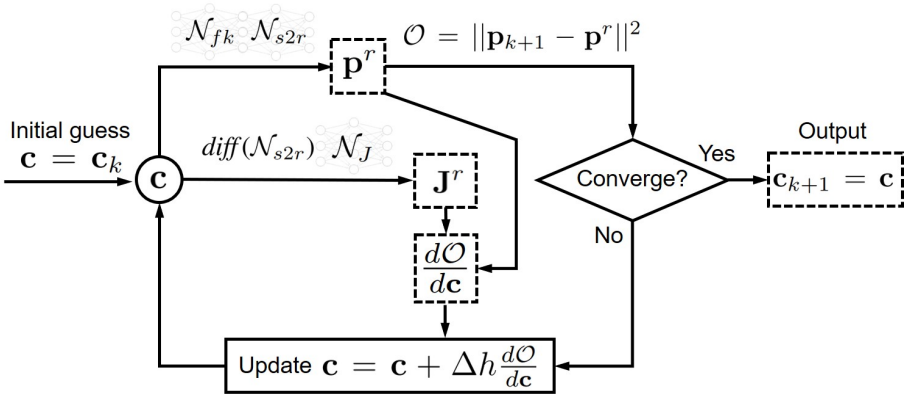


Figure 5.1: Pipeline of Jacobian-based method to determine the actuation parameters $\mathbf{c}_{k+1}$ for the way-point $\mathbf{p}_{k+1} \in \mathcal{L}$ that solves the IK problem of soft robots by minimizing $\mathcal{O}(\cdot)$ in (5.1). Both the position $\mathbf{p}^r$ in task space and the Jacobian $\mathbf{J}^r$ are effectively estimated by the networks $\mathcal{N}_{fk}$, $\mathcal{N}_J$ and $\mathcal{N}_{s2r}$ obtained from offline training. When $\mathcal{O} < \epsilon^2$, we regard the convergence as having been achieved (e.g., 0.1% of the workspace width is employed as ϵ).

the reduced analytical model vs. our simulation based model can be found in Sec. 5.4.3.

5.1.2 Our Method

Three networks - 1) forward kinematics $\mathcal{N}_{fk}$, 2) Jacobian $\mathcal{N}_J$, and 3) sim-to-real mapping $\mathcal{N}_{s2r}$ are trained to support the effective computing of IK for soft robots in both virtual and physical spaces at a fast speed. With an objective function defined in quadratic form and the network-based efficient estimation of FK and Jacobian, Jacobian-based iteration is used to compute the IK solution. The pipeline of our method is presented in Fig. 5.1 with detail discussed in Sec. 5.2.

The technical contributions of our work are:

- A direct pipeline for learning both the FK and Jacobian from accurate numerical simulation results, to support effective IK computing for soft robots. This method can compute IK solutions in a fast speed and has the capability to plan a smooth motion for soft robotic systems with redundancy.
- A two-step learning strategy by using the sim-to-real transfer learning to eliminate the gap between the prediction based on simulation and

the physical behavior, which can greatly reduce the required amount of empirical data when compared to directly learning a predictor from the physical experiment.

The behavior of our method has been verified on two hardware setups of soft robots giving 2D and 3D motions. The effectiveness of our method is quantitatively evaluated and compared with other approaches in the IK tasks of soft robots. Experimental tests are also conducted to demonstrate the performance of our method on soft robots with the same design but fabricated with different materials.

5.2 Jacobian-Based Kinematics and Learning

In this paper, we focus on solving the IK problem for soft robots – specifically, to determine the parameters of actuation that can drive a soft robot to reach a target position / shape. As the analytical IK solution cannot be obtained, we adopt a Jacobian-based numerical method where a target-oriented objective function $\mathcal{O}(\cdot)$ is minimized to determine the parameters of actuation. In this section, we first introduce the Jacobian-based IK computation for the path following task. After that, we demonstrate how it can be solved practically by applying the training in a virtual environment and then the sim-to-real transformation.

5.2.1 Jacobian-Based IK Solution

The path following problem of a soft robot is described as driving a marker on its end-effector to move along a path $\mathcal{L}$ presented by a set of target waypoints $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_i, \mathbf{p}_{i+1}, \cdots\}$ in the task space. For each waypoint $\mathbf{p}_i$ to be reached by the marker, numerical computation of IK attempts to minimize the distance between $\mathbf{p}_i$ and the marker's position. This is formulated as an optimization problem

$$\mathbf{c}_i = \underset{\mathbf{c}}{\operatorname{argmin}}\, \mathcal{O}(\mathbf{p}_i, \mathbf{c}) = \underset{\mathbf{c}}{\operatorname{argmin}}\, \|\mathbf{p}_i - \mathbf{p}(\mathbf{c})\|^2 \tag{5.1}$$

where $\mathbf{p}(\cdot) \in \mathbb{R}^n$ denotes the forward kinematic function to compute the position of the marker. The input of $\mathbf{p}(\cdot)$ is a vector of actuation parameters, $\mathbf{c} = (c_1, c_2, \cdots, c_m) \in \mathbb{R}^m$. Here n and m are dimensions of the task space and the actuator space respectively.

To find the solution of (5.1), the gradient of $\mathcal{O}(\cdot)$ as

$$\frac{d\mathcal{O}}{d\mathbf{c}} = -2(\mathbf{p}_i - \mathbf{p}(\mathbf{c}))\mathbf{J}(\mathbf{c}) \tag{5.2}$$

will be employed to update the value of $\mathbf{c}$ with $\mathbf{J}(\mathbf{c}) = d\mathbf{p}/d\mathbf{c} \in \mathbb{R}^{n\times m}$ being the Jacobian matrix that describes the moving trend of a soft robot's body at certain actuation parameters. The value of $\mathbf{c}$ is updated by $\mathbf{c} = \mathbf{c} + \Delta h \frac{d\mathcal{O}}{d\mathbf{c}}$. Δh is a step size to minimize the value of $\mathcal{O}(\cdot)$ along the gradient direction, which can be determined by soft linear-search [100]. Fig. 5.1 gives the illustration of this algorithm.

When a physics-based simulation is employed to evaluate the forward kinematic function $\mathbf{p}(\cdot)$, the Jacobian matrix $\mathbf{J}$ can be obtained by numerical

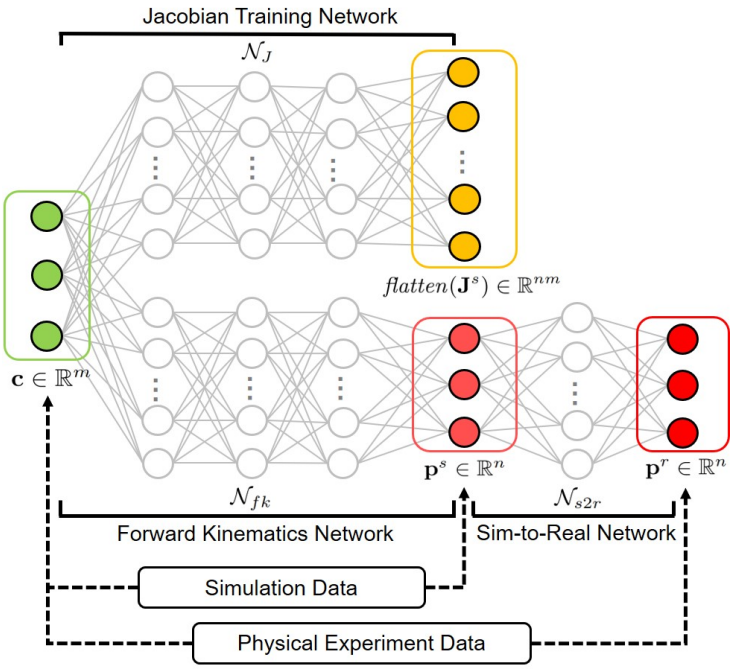


Figure 5.2: The network structure used in our approach to train the kinematic model and the sim-to-real transfer.

difference [100, 148]. The k-th column of $\mathbf{J}$ is computed as

$$\mathbf{J}_k = \frac{\partial \mathbf{p}(\mathbf{c})}{\partial c_k} \approx \frac{\mathbf{p}(..., c_k + \Delta c, ...) - \mathbf{p}(..., c_k - \Delta c, ...)}{2\Delta c}, \tag{5.3}$$

where Δc is a small constant determined according to experiments and assigned as $1/10N$ of the actuation range. Here N is the number of samples for each actuation parameter presented in Sec. 5.3.2. Notice that it can be time-consuming to evaluate the values of $\mathbf{p}(\cdot)$ and $\mathbf{J}(\cdot)$ by physics-based simulation in IK computing. We therefore introduce a learning-based method to learn both the FK and the Jacobian model in the offline stage, which can support a fast IK computing during online usage. In the meantime, the difference between the simulation and the physical behavior is fixed by the sim-to-real transfer learning.

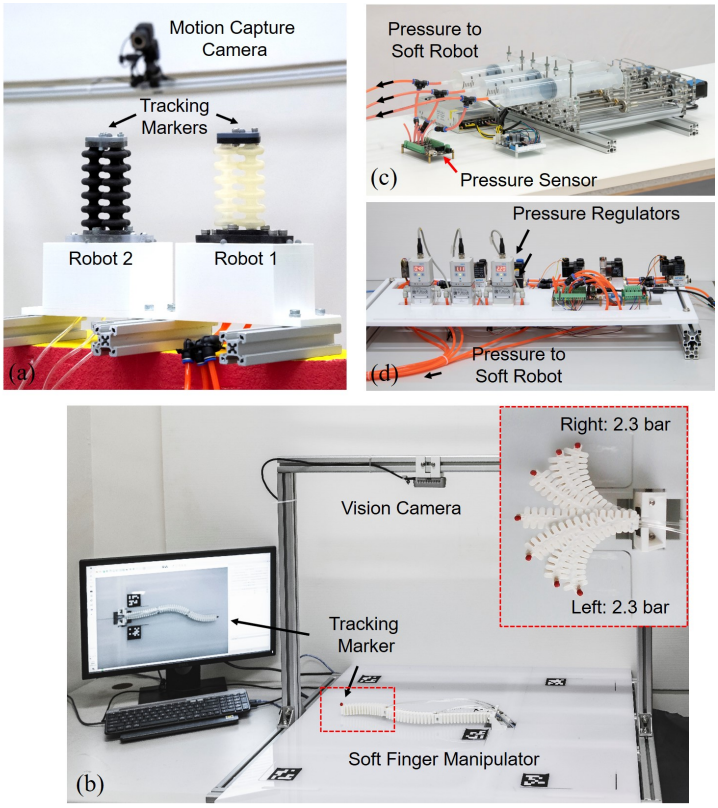


Figure 5.3: Two hardware setups employed in our experiments to collect data and verify the performance of our method – (a) a soft actuator with multiple chambers that are actuated by an array of syringes (see (c)) and (b) three connected soft fingers that can be actuated individually by proportional pressure regulators (see (d)).

5.2.2 Learning-Based Model for IK Computing

We learn both the forward kinematic model and its Jacobian from simulations – denoted by $\mathbf{p}^s(\cdot)$ and $\mathbf{J}^s(\cdot)$, which are transferred to physical hardware by learning a sim-to-real mapping function $\mathbf{r}(\cdot)$. Denoting the location of a traced marker on physical hardware as $\mathbf{p}^r$, the function of sim-to-real mapping is required to have $\mathbf{r}(\mathbf{p}^s) \approx \mathbf{p}^r$. Neural networks are employed to learn these functions (see the architecture of neural networks shown in Fig. 5.2).

In the simulation environment, $\mathbf{p}^s(\cdot)$ and $\mathbf{J}^s(\cdot)$ are trained on two networks $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ by spanning the work space of actuators with a large number of samples. Note that the output layer for $\mathcal{N}_J$ is a column vector as the flattened

Jacobian matrix $\mathbf{J}^s$. After obtaining the network $\mathcal{N}_{fk}$, the sim-to-real mapping function $\mathbf{r}(\cdot)$ is trained on a differentiable network $\mathcal{N}_{s2r}$ by using a few samples obtained from a physical experiment conducted on the hardware setup.

With the help of these trained networks, we can estimate the Jacobian on the hardware setup as

$$\mathbf{J}^r(\mathbf{c}) = \frac{d\mathbf{p}^r}{d\mathbf{p}^s}\frac{d\mathbf{p}^s}{d\mathbf{c}} \approx \mathrm{diff}(\mathcal{N}_{s2r})\mathbf{J}^s(\mathbf{c}) \tag{5.4}$$

Considering the difficulty of data acquisition on hardware specimens, the *feed-forward neuronal network* (FNN) with a single layer of fully connected neurons is adopted in our implementation for $\mathcal{N}_{s2r}$. The differentiation $\mathrm{diff}(\mathcal{N}_{s2r})$ as a $n \times n$ matrix can be computed analytically by differentiating the network's activation functions. As most of the complexity in kinematics can be effectively captured by $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, a lightweight network $\mathcal{N}_{s2r}$ trained by a small dataset obtained from the physical experiment can already show very good performance on eliminating the inconsistency in material properties and fabrication.

Through this learning-based model, the gradient of the IK objective function in the physical environment can then be computed by

$$\frac{d\mathcal{O}}{d\mathbf{c}} = -2(\mathbf{p}_i - \mathbf{p}^r(\mathbf{c}))\mathbf{J}^r(\mathbf{c}) \tag{5.5}$$

$$\approx -2(\mathbf{p}_i - \mathbf{r}(\mathbf{p}^s(\mathbf{c})))\mathrm{diff}(\mathcal{N}_{s2r})\mathbf{J}^s(\mathbf{c}) \tag{5.6}$$

Note that the real positions of markers, $\mathbf{p}^r(\mathbf{c})$ in (5.5), can also be obtained from a hardware setup (e.g., by a motion-capture system). However, using positions predicted by $\mathcal{N}_{fk}$ and $\mathcal{N}_{s2r}$ networks can avoid physically actuating the hardware inside the loop of numerical iteration. After training the networks $\mathcal{N}_{fk}$, $\mathcal{N}_J$ and $\mathcal{N}_{s2r}$, an iteration-based algorithm (as shown in Fig. 5.1) is used to effectively solve (5.1). The actuation parameters $\mathbf{c_{i-1}}$ for realizing $\mathbf{p}_{i-1}$ are employed as the initial guesses when computing $\mathbf{c_i}$ for $\mathbf{p}_i$. As a result, the iteration converges rapidly and the continuity of motion in configuration space can be preserved.

5.3 Data Generation and Training

We first present two hardware setups that are used in our research to verify the performance of the learning-based method presented above. After introducing the steps for generating datasets, the training details are provided.

5.3.1 Soft Robotic Hardware

Two hardware setups are constructed to investigate the performance of our IK solver. Both setups are equipped with cameras to capture the real positions of markers for the purpose of training and verification.

Actuator with 3D motion

The first setup is a 3D printed soft actuator with three chambers that can be actuated individually [80]. Its soft body can extend and bend in a 3D task space. To verify the behavior of our sim-to-real method, two specimens are fabricated by the same Object 350 Connex 3D printer but using slightly different materials – the Agilus Black and Agilus transparent materials. Both have the softness 70A according to their factory specification. These two models are shown as Robot 1 and Robot 2 in Fig. 5.3(a). The soft robot is actuated by an array of syringes that has close-loop control with the help of pressure sensors as shown in Fig. 5.3(c). For this setup, we have the same dimension for the work space ($m = 3$) and the actuator space ($n = 3$).

Planar finger manipulator

The second setup is a soft manipulator that can move in the xy-plane (see Fig. 5.3(b)). The manipulator contains three soft finger sections that are rigidly connected. We use Festo Pressure Regular VPPE-3-1/8-6-010 to provide the pressure for each section (see Fig. 5.3(d)). Every finger section contains dual chambers that can bend symmetrically for both sides up to $120°$. To maximize the deformation of each finger section, we only actuate one side for a segment each time with the pressed air in the range of $[0, 3]$ bar. When considering both sides of a segment, this results in a range of $[-3, 3]$ as actuation – i.e., '+' for actuating the chamber at one side and '-' for the other side. This is a redundant system with the dimension $n = 2$ for the work space and $m = 3$ in the actuator space.

5.3.2 Data Generation on Simulator

In our work, forward kinematics of soft robots in a virtual environment is computed by a geometry-oriented simulator [13, 100]. When employ this simulator to generate datasets for training the forward kinematic network $\mathcal{N}_{fk}$ and the Jacobian network $\mathcal{N}_J$, the computation time for generating single sample point is 4.3 sec. (the three-chamber robot) and 1.2 sec. (the finger manipulator) respectively. It's worth mentioning that as a general training pipeline, the dataset used to train $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ can be generated by different kinematic models – e.g., those analytically computed by piecewise constant curvature [126] or numerically by FEM software such as Abaqus. Here we choose the geometry-based simulator as it can further reduce the cost of data generation than FEM. Moreover, it needs to capture less physical data than the analytical model for training the sim-to-real network (discussed Sec. 5.4.3).

We now present the sampling method in actuator space for generating training data points. We uniformly divide the pressure range of each actuator into N segments to make sure that the distance between sample points is less than 1% of the workspace width. Sampling results of the two hardware setups are shown in Fig. 5.4, which also presents the workspaces $\mathcal{P}^w$ of these two soft robots. In our experiment, $N = 16$ and $N = 29$ are used for these two setups respectively. This results in $16^3 = 4096$ samples for the three-chamber actuator (Fig. 5.4(a)) and $29^3 = 24389$ samples for the finger manipulator (Fig. 5.4(b)). Notice that the difference in choosing the N value is due to the redundancy of the finger setup, which also has a larger range in actuation. Based on our tests, datasets selected in these sizes can already well train $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ to capture the kinematic behavior of soft robots (see the results in the following section).

5.3.3 Data Generation on Hardware

For the purpose of training sim-to-real network $\mathcal{N}_{s2r}$, datasets are generated on two hardware setups. We uniformly span the actuator space to generate physical data which are classified into the training (70%) and the test (30%) dataset. Since the efficiency of the training pipeline depends on the number of samples generated on hardware setups, we test and determine the appropriate sample number used to train $\mathcal{N}_{s2r}$ – details are presented in Sec. 5.3.4.

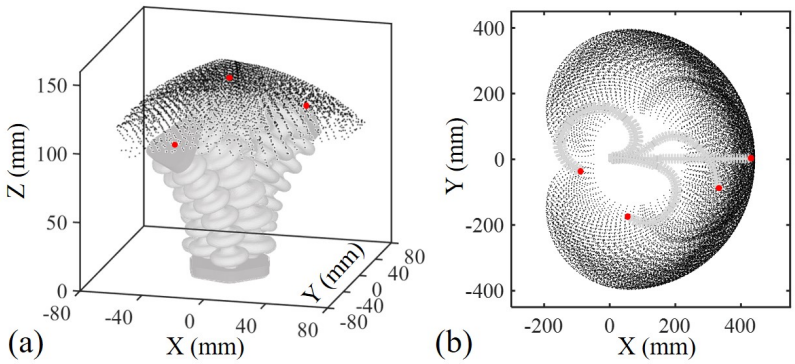


Figure 5.4: The results of the simulation are employed as training samples (present in black dots) to learn the forward kinematic network $\mathcal{N}_{fk}$ and the Jacobian network $\mathcal{N}_J$, where these samples also span the workspace $\mathcal{P}^w$ of a robot. Red dots represent some example points as targets for motion in the workspace.

Actuator with 3D motion

To trace the 3D motion of this soft actuator, we place a marker at the center of its top plane and several markers on its static base. The motion capture system that contains 8 Vicon Bonita 10 cameras and 10 Vicon Vantage 5 cameras is used to capture the movements at the rate of 30 Hz. Because of the viscoelasticity of soft materials used to fabricate this robot, it takes a relatively long time for the position of a marker to become stable (i.e., less than 0.05mm change between neighboring image frames). This makes the process of data collection more time-consuming than a robotic system with rigid bodies. As a result, the average time for collecting one sample in the physical environment is 4.0 sec.

Planar finger manipulator

As only planar coordinates are needed when tracking the positions of a marker, we use a RealSense D435 camera mounted at the top of the setup. We place a red marker on the tip of the manipulator and adopt the OpenCV library as software to track the marker's position in the plane. QR code is employed to build the mapping between the coordinates in image space and the coordinates in the real world. The speed of data acquisition for this system is 10 Hz. For this hardware setup, the average time for collecting one sample point is 3.5 sec.

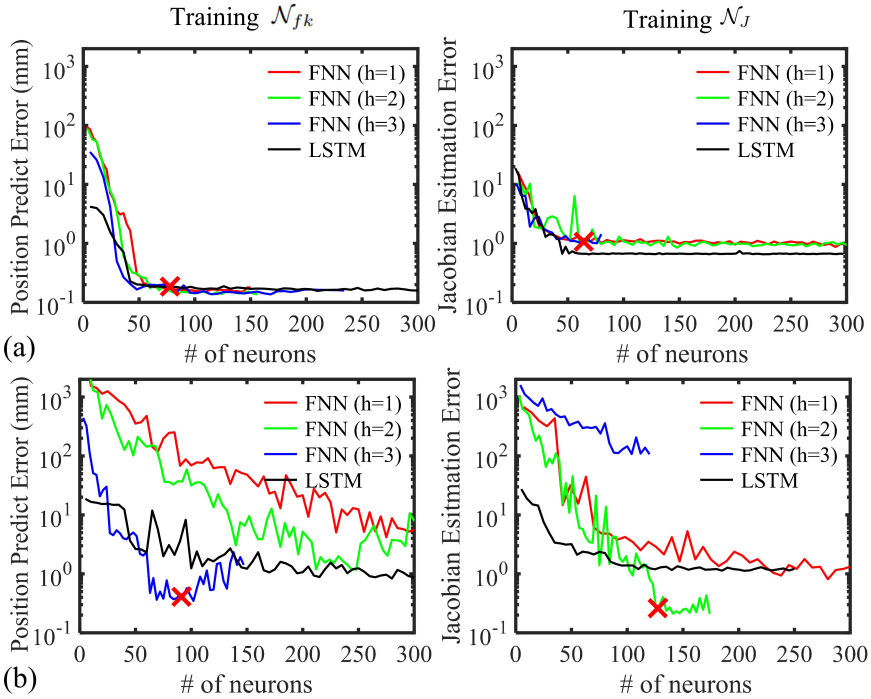


Figure 5.5: Comparison of learning results by using a different number of layers as h and a different number of neurons b per layer (i.e., the total number of neurons in a network is hb). Tests are conducted on (a) the robotic setup without redundancy (the three-chamber actuator with $m = n = 3$) vs. (b) the setup with redundancy (the planar finger manipulator having $m = 3$ and $n = 2$). Red crosses indicate the network parameters used in the physical experiment.

5.3.4 Details of Training

In this work, the type and structure of training models used in experiment are carefully selected based on their performance. For training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, FNN and *Long Short-term Memory* (LSTM) are tested as they can both adequately capture the nonlinear behavior in a training dataset, including the many-to-one forward kinematic mapping for redundant systems. For the sim-to-real transfer network $\mathcal{N}_{s2r}$, it needs to be differentiable with analytic gradients. Meanwhile, it should be lightweight as only a limited number of samples can be obtained from physical experiments. For these reasons, single-layer FNN is selected for $\mathcal{N}_{s2r}$. All networks are trained by using the Deep Learning Toolbox of MATLAB running on an NVIDIA GeForce RTX 2070 graphics card.

Training for FK and Jacobian

We first study the effectiveness of training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ by using a different number of layers and different numbers of neurons. Each data set is divided into training and test subsets in the ratio of $70\% : 30\%$. For all networks, the activation function is set as Tan-Sigmoid

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{5.7}$$

as it can well fit the non-linearity in kinematic mapping. Moreover, it's differentiable and can provide a faster training speed. We set the batch size as 200, the maximum number of epochs as 13500 and the learning rate as 0.04. The Levenberg-Marquardt backpropagation is employed for training.

The estimation errors for both soft robot setups are evaluated on the test datasets as shown in Fig. 5.5, where we can find that FNN and LSTM can both converge to a good result after carefully tuning the network parameters. It is hard to find significant improvement in the accuracy by using network with feedback connections (i.e., LSTM). This is because the training data only contain quasi-static information of the system, where the time-related network structure cannot show its advantage. On the other hand, it is found that the structure of the network for learning the Jacobian $\mathcal{N}_J$ on a redundant system (i.e., the planar finger manipulator) needs to be selected more carefully. FNN is selected as the final network structure, and the best performance for training $\mathcal{N}_J$ is observed on this hardware setup when FNN with $h = 2$ hidden layers and $b = 64$ neurons per layer is employed to learn $\mathcal{N}_J$. Differently, FNN with $h = 3$ layers and $b = 30$ neurons per layer gives the best results in training $\mathcal{N}_{fk}$. The error of position prediction by using $\mathcal{N}_{fk}$ is less than 0.5 mm (i.e., 0.58% of the work space's width). For the three-chamber actuator, the numbers of layers and neurons have less influence on the training result. For this setup, we select $h = 2$ and $b = 35$ for both networks, which results in a FK prediction with error less than 0.17 mm on the test dataset (i.e., 0.34% of the workspace's width). With such accurate predictions generated by $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, we can obtain IK solutions efficiently and accurately (see the behavior study given in Section 5.4).

Training for sim-to-real transfer

When training for $\mathcal{N}_{s2r}$ an important parameter here is the number of neurons, which is selected as η times the number of samples to avoid over-fitting on the

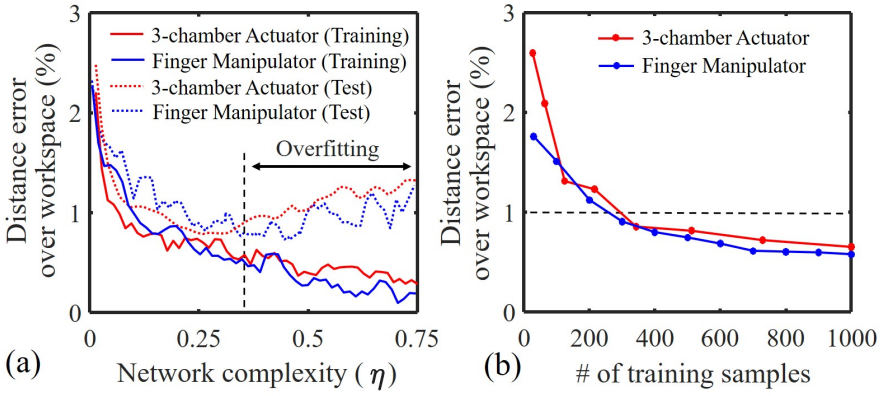


Figure 5.6: Experimental study for the performance influence in the sim-to-real network $\mathcal{N}_{s2r}$ by using (a) different numbers of neurons and (b) datasets in different sizes. With the properly selected complexity of network structure, the over-fitting problem can be avoided. The distance predict error can be controlled within 1% of the workspace width for both setups when a limited number of training samples are used (see (b)).

training dataset. Fig. 5.6(a) presents the behavior on both the training dataset (denoted by the solid curves) and the test dataset (denoted by the dash curves) when using different values of η. Based on the analysis, $\eta = 1/4$ is selected for our experiment to avoid over-fitting.

As the time used to collect physical data points should be controlled, we also study the behavior of $\mathcal{N}_{s2r}$ with different numbers of training samples. For this purpose, the prediction errors as the ratios of the distance errors over the workspace widths are given in Fig. 5.6(b) to study the effectiveness of using different numbers of samples. In these tests, the number of neurons is always assigned as $\eta = 1/4$ of the training samples. For both setups, we find that the network $\mathcal{N}_{s2r}$ can be well trained when using a limited number of training samples. In our implementation, 1% is selected as the threshold of accurate prediction and this threshold is used to determine the number of samples for training $\mathcal{N}_{s2r}$. As a result, 343 samples are used for the three-chamber actuator and 620 samples are conducted for the finger manipulator. The datasets for training $\mathcal{N}_{s2r}$ on two hardware setups can both be collected within 30 min.

5.4 Experimental Results

In this section, we present all the experiment results of IK computing for soft robots by using our learning based Jacobian iteration. The results are generated in both the virtual and the physical environments. Computation of the learned neural networks in prediction is implemented in C++ and integrated into our control platform to gain the best computational efficiency. All the IK computations were efficiently run on a laptop PC with Intel i7-9750H 2.60GHz CPU and 16GB memory. Note that the prediction made by networks and the IK algorithm is entirely run on a CPU.

5.4.1 Path Following by Actuator with 3D Motion

We first test the behavior of the learning-based IK computing method in the task of path following on the soft actuator with three chambers. Given 3D trajectories of the 'flower' shape (Fig. 5.7) and the 'box' shape (Fig. 5.8), 120 and 240 waypoints are sampled on the paths in uniform distances respectively. The proposed IK solver is then used to compute the actuation parameters for each waypoint. This leads to the actuation sequence that can drives soft robot to move along the trajectories. When running in the simulation environment, the trained networks can generate actuation that results in very accurate trajectories with the average tracking error as $0.13\mathrm{mm}$. In the physical environment, we learned the sim-to-real networks separately on two soft robots as shown in Fig. 5.3(a). If we directly apply the actuation parameters obtained from IK computing in the simulation environment, the error of path following is high (i.e., up to $5\ \mathrm{mm}$). At the same time, the variation caused by fabrication and material can be clearly observed from the difference between Robot 1 and 2, as shown in Fig. 5.7. By incorporating the sim-to-real transfer in our method, we can successfully reduce the error in the physical environment to less than $1.2\mathrm{mm}$ for both robots (see Fig. 5.7(b)), which is 1.71% of the workspace width. For the tests given in Fig. 5.8, the maximal errors are reduced from $4.5\mathrm{mm}$ to $1.0\mathrm{mm}$.

Besides the accuracy, another advantage of our learning-based approach is its low computational cost. Thanks to the efficient forward propagation process of FNN networks and fast converge speed of the Jacobian-based algorithm, the time used to compute single waypoint IK for the three-chamber setup is less than 30ms even when a large number of neurons $hb = 128$ is used. The quantitative analysis of the converge speed and IK computing time of our algorithm is showed in Fig. 5.9, and also compared with other existing

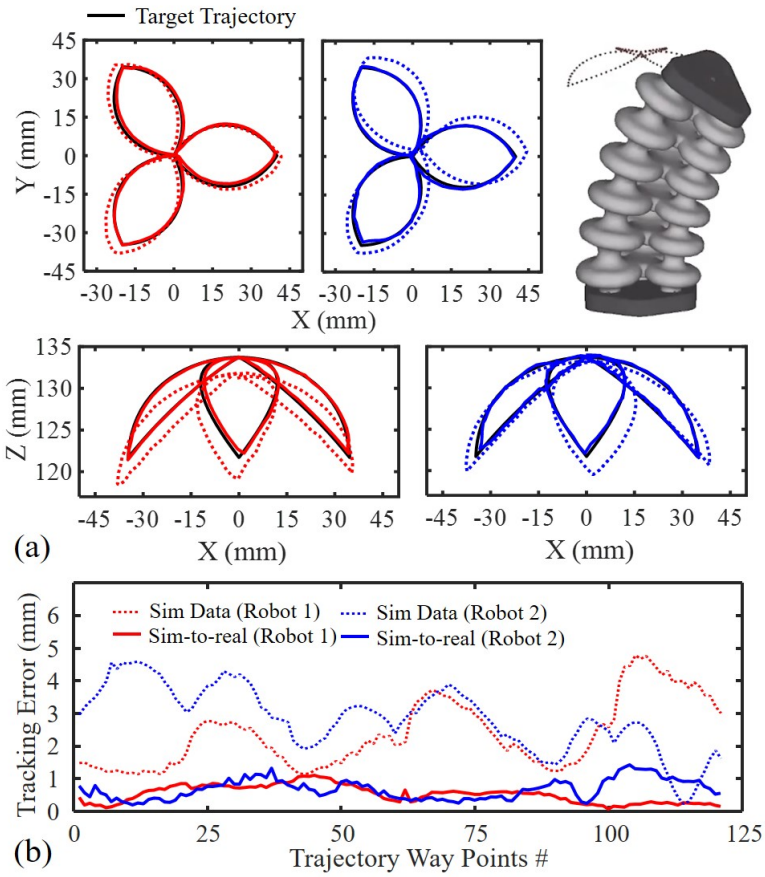


Figure 5.7: Results of path following task on two soft robots with the same design but fabricated with different materials (shown as Robot 1 and Robot 2 in Fig. 5.3(a)). (a) Trajectory of the soft robots by applying IK solutions with (solid line) and without (dash line) the sim-to-real network. (b) Visualized tracking errors on trajectory way-points.

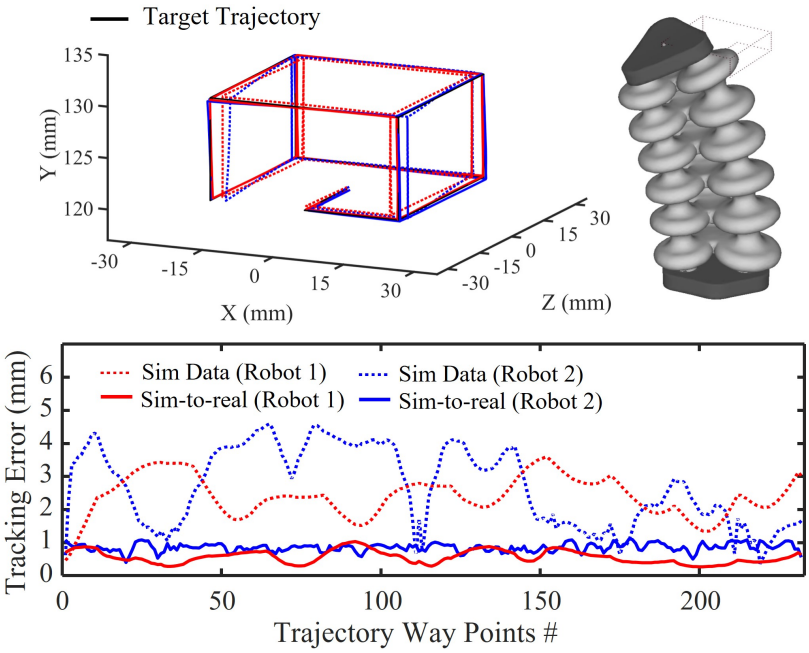


Figure 5.8: Results of path following for the 'box' shape trajectory demonstrate the vertical motion of three-chamber soft robot. It can also find that the maximal tracking errors are reduced from 4.5mm to 1.0mm for both robots after applying the sim-to-real network.

solutions (see Table 5.1). It can be found that learning-based method (for both analytical or numerical methods) can generally provide a more accurate IK result than model-based solution as it can well capture the uncertainties that are hard to calibrate (e.g., material shifting, fabrication error, etc.). When compared between learning-based methods, direct IK learning is the fastest (see Fig. 5.9(b)). Our Jacobian-based method provides the best accuracy and can also ensure a real-time computing speed (i.e., at the rate of $35+$ waypoints per second). What's more important and also as aforementioned in Sec. 5.1.1, our solution can well handle the redundant soft robots to ensure minimum variation when travelling along the trajectory. This is difficult to be handled by direct IK training.

Table 5.1: Computing Speed and IK Accuracy by Different Methods

	Model-based Solution		Learning-based Method†	
	Analytical Model [129]	Numerical Model [?]	Direct IK Learning [41]	This work
IK Time	12.6 ms	138 s	3.2 ms	28 ms
Ave. Error	8.2 mm	4.7 mm	0.97 mm	0.78 mm

†The same dataset was applied to learn the direct IK mapping [41] and sim-to-real network $\mathcal{N}_{s2r}$ in our pipeline.

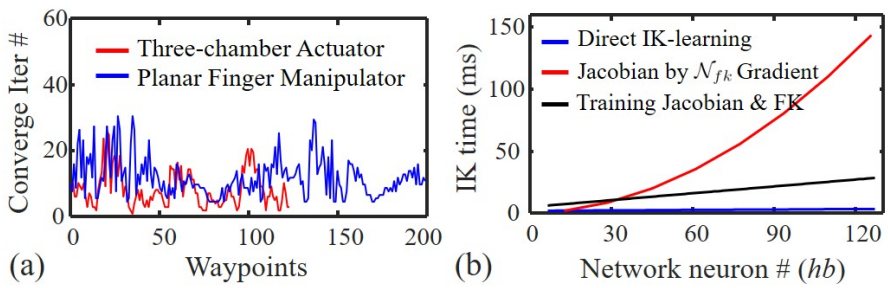


Figure 5.9: Quantitative analysis for (a) the speed of convergence and (b) the time efficiency of our Jacobian-based training method. Three learning strategies for computing IK are compared in (b).

5.4.2 Experiment with Soft Finger Manipulator

The soft finger manipulator shows in Fig. 5.3(b) is a redundant system, which has a higher DOFs in its actuation space ($m = 3$) than the task space ($n = 2$). Therefore, an input waypoint can have multiple IK solutions. Both the path following and the interactive positioning tasks are conducted to validate the performance of our learning-based IK solver on the redundant systems.

Path following

We first present the results of following an '8'-shape trajectory that contains 200 way points as shown in Fig. 5.10(a). The actuation parameters obtained from the Jacobian-based method are compared with those resulting from direct IK-learning. Our Jacobian-based IK by learning demonstrates excellent

performance in the accuracy of tracking precision. The average and maximum tracking errors for all waypoints are 0.08% and 0.18% of the workspace width, respectively. As shown in Fig. 5.10(c), our method is able to ensure a smooth motion that minimizes the variation in actuator space. As a comparison, large variation (i.e., jumps) in the actuator space can be found in the results of direct IK-learning (circled by dash lines in Fig. 5.10(c)). This problem of direct IK learning is mainly caused by its lack of capability to support the one-to-many IK mapping. For this trajectory, the IK solutions can be efficiently computed by our method at the average speed of $39\mathrm{ms}$ per waypoint.

It is worth to mention that the configuration of motion computed by our method is highly dependent on the selection of the IK solution at the starting waypoint (i.e., the initial value). As demonstrated in Fig. 5.11, the configuration of the finger manipulator determined by our IK solver at a waypoint is always close to the IK solution of the previous waypoint where this dependency can trace back to the beginning point. This is because our Jacobian-based iteration tends to minimize the distance-based objective function defined in (5.1) while minimizing the change in actuation parameters. This preferred property is also kept when computing IK solutions for a motion passing through the singularity region (see the example in Fig. 5.12 for following a 'L' shape path). As a Jacobian-based iterative solver, our method can always generate a nearly optimal solution for singularity points by applying appropriate terminal conditions (e.g., minimal variation in the value of the objective function).

Interactive positioning

The experiment of interactive positioning is also conducted on the soft finger setup. As shown in Fig. 5.13(a), users can select the desired point location for the manipulator's tip through our interface and our planner will compute the IK solutions as the corresponding actuation parameters. The computation can be efficiently completed at an average speed of 47ms together with the sim-to-real network $\mathcal{N}_{s2r}$. As a result, users can interactively position the manipulator's tip – see also the supplementary video. When different positions are selected in the work space, the soft manipulator can move among configurations with large variations. The errors of positioning are evaluated and presented in Fig. 5.13(b) as a bar chart. It is found that all six target positions can be realized in the physical environment with tracking errors less than 0.9% of the work space's width. Note that, each of these six target positions is tested 10 times in random order to study the repeatability of our system. The results are displayed as the range of derivation on the bar chart.

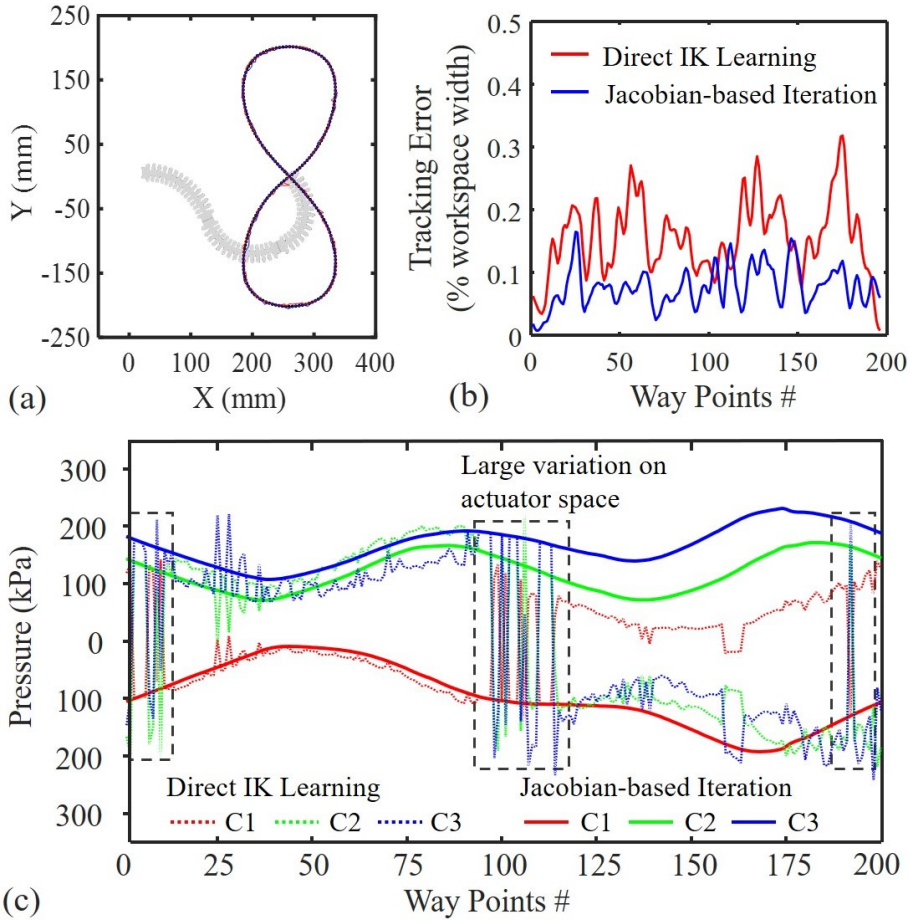


Figure 5.10: (a) Path following result on the soft finger manipulator with the '8'-shape trajectory. (b) Comparison of tracking errors in the task space from the direct IK learning vs. the Jacobian-based iteration by learning (our method). (c) Visualization of IK solutions in the actuator space, where C1, C2 & C3 present the actuation parameters (i.e., pressures) in three different chambers – large variation can be found from the results obtained by direct IK learning.

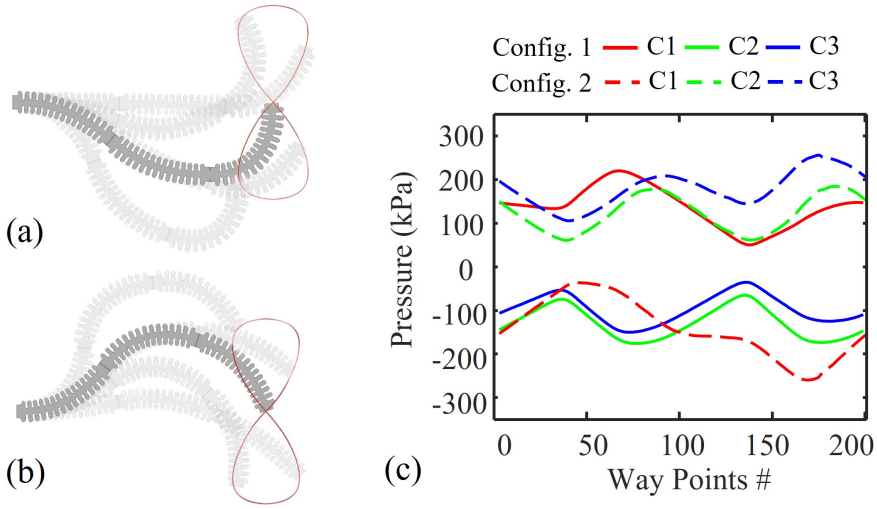


Figure 5.11: Two different configurations of motion are shown in (a) and (b), both of which are feasible IK solutions for following the '8'-shape path by the soft finger manipulator. The smoothness in motion is guaranteed by the Jacobian-based iteration for both results as shown in (c), where the actuation in every chamber has minimal variation in control parameters between neighboring waypoints.

It is observed that our method can generate results in different configurations for two close waypoints (e.g., the point b and c shown in Fig. 5.13) when using initial values that are always far from the resultant configurations (zero is adopted for all actuation parameters in this case). Together with the results presented in trajectory following experiments (e.g., Fig. 5.11), our method shows the capability of determining one 'nearest' IK solution among all feasible IK solutions.

5.4.3 Statistical Analysis for Sim-to-Real Transfer

Experiments have been conducted on the setup of three-chamber soft robot to explicitly compare the behavior of sim-to-real transfer by using different models in the virtual environment. Compared to the numerical simulator used in this work, the reality gap becomes larger when the dataset for training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ is obtained from an analytical model (ref. [129, 80]). When using the analytical model, more samples are needed for training the sim-to-real network $\mathcal{N}_{s2r}$ to achieve the similar accuracy. As observed in Fig. 5.14(a), the model trained by the dataset obtained from the numerical simulation (i.e., our

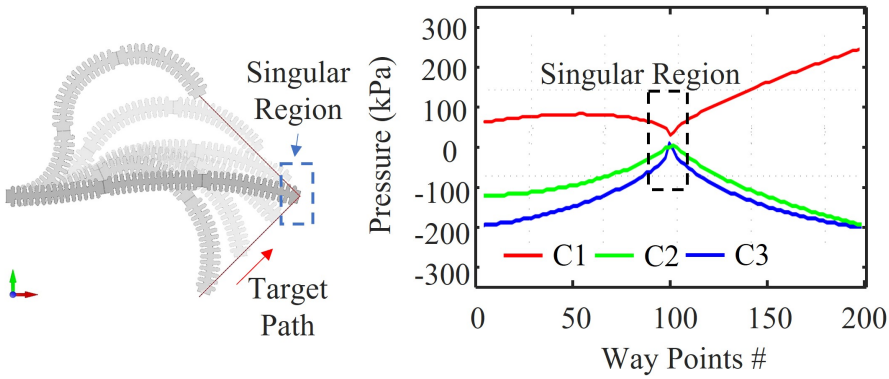


Figure 5.12: Results for the path following task passing through the singularity region. Our method can successfully compute feasible smooth motions when meeting singularity.

approach) shows smaller prediction errors when using the same number of samples to learn the sim-to-real network $\mathcal{N}_{s2r}$. Note that this less accurate result is still observed even after using more samples generated from the analytical model (e.g., 64000 in our experiment) when the number of real samples is fixed.

This experiment also proves that the sim-to-real network can effectively eliminate the gap between simulation and reality – although requiring different numbers of samples for the model learned from numerical simulation (i.e., our method) and the analytical model. Fig. 5.14(b) shows that the accuracy within 1% of the workspace width can be obtained at most points when TS# = 300 and TS# = 1000 are used by our method and the analytical model respectively. As a general case, it is more costly to generate a large number of physical training samples than simulation-based samples. Generating a dataset of empirical samples in a large number is very time-consuming and may result in material failure. Our method proposed in this paper converts this challenge into an approach that is easier to realize, in other words, learning a more accurate predictor from more accurate samples generated by numerical simulation. As a result, the gap between prediction and reality can be reduced and fixed by a light sim-to-real network.

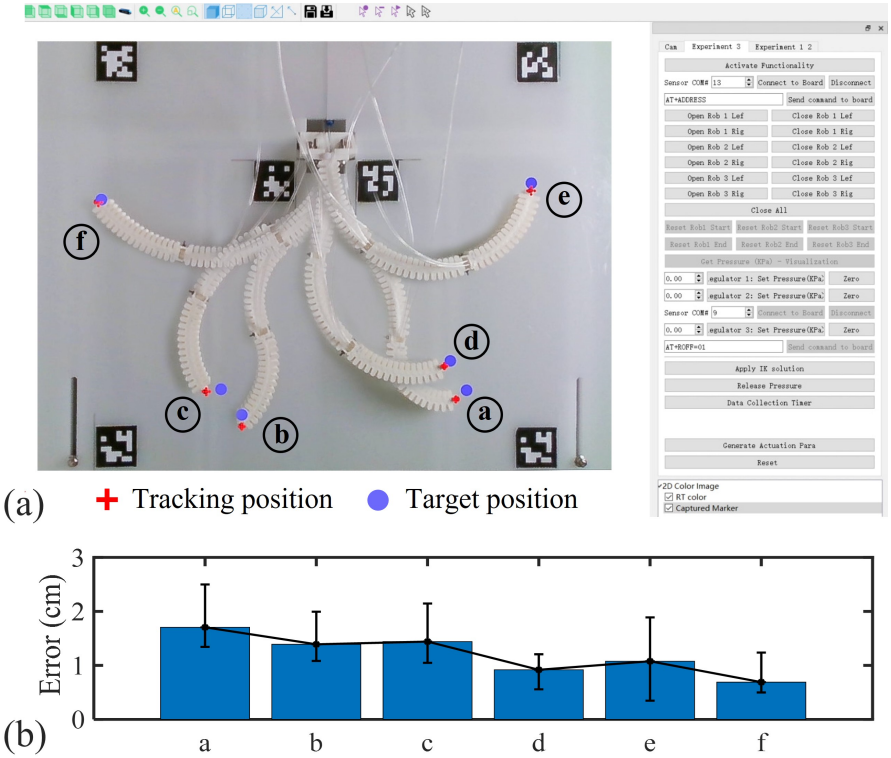


Figure 5.13: Interactive positioning results for the soft manipulator with three finger actuators. (a) With a user-specified position given through the software interface, our Jacobian-based method is applied to determine the IK solution. (b) The bar chat presents the tracking errors for different target positions, where the repeatability is also studied and displayed as the range of deviation in tracking errors.

5.5 Discussion

This section discusses limitations of our learning-based IK solver. A rigorous analysis is conducted to exhibit the influence of different velocities and external loads on soft robots. The possible extension of pose estimation by our method is presented.

In the proposed Jacobian-based learning, we focus on the computation of quasi-static kinematics. The training datasets in both virtual and physical setups are collected under the quasi-static status, where the hysteresis problem in soft materials is neglected. As one major limitation of this work, only considering the quasi-static model introduces large errors in the task of path-

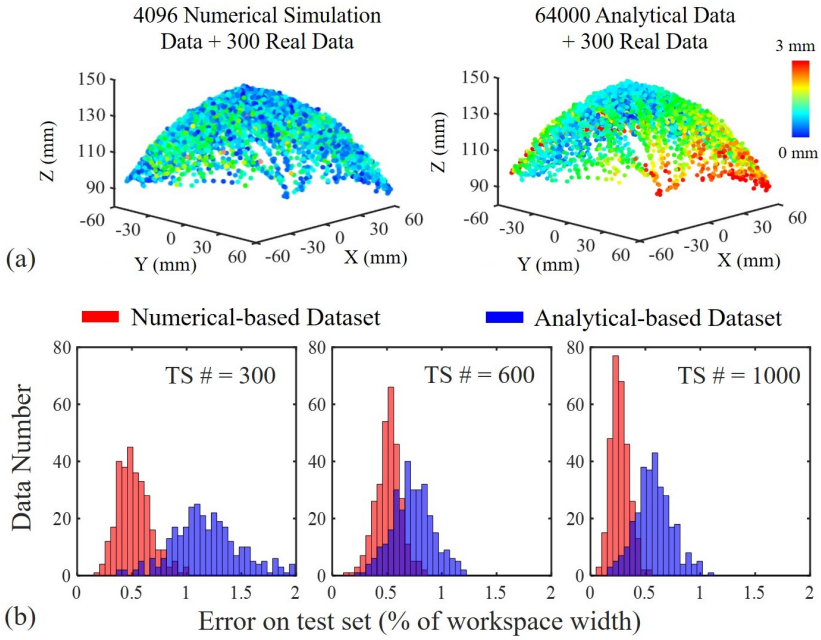


Figure 5.14: Comparison for studying the performance of sim-to-real learning on 1) numerical simulation-based (our approach) vs. 2) analytical computation-based (ref. [129, 80]) training datasets. . (a) Visualization of prediction errors in the task space. (b) Histograms show the distributions of prediction errors when using different numbers (mark as TS #) of samples to train $\mathcal{N}_{s2r}$.

following when the dynamic behavior of a soft robot is performed (e.g., the velocity of motion is high). Experiments are conducted to study the influence of speed on motion regarding the trajectory's accuracy; the results are illustrated in Fig. 5.15. When the speed of motion is set as less than 1.5 mm/s, the tracking errors can be controlled less than 1.3mm. However, when the speed is increased to 4 mm/s – i.e., the maximum speed that actuation system presented in Fig. 5.3(c) can support – a relatively large error (i.e., the maximal error as 2.7 mm) can be found on the trajectory. One possible solution to incorporate the hysteresis property of soft materials and dynamic behavior of soft robots in IK computing is to apply the time-variant network structure (e.g., recurrent neural network [149]) based on datasets generated in different velocities. This requires much larger training datasets, the generation of which is very time-consuming [73].

On the other aspect, the trained sim-to-real network lost its capability to

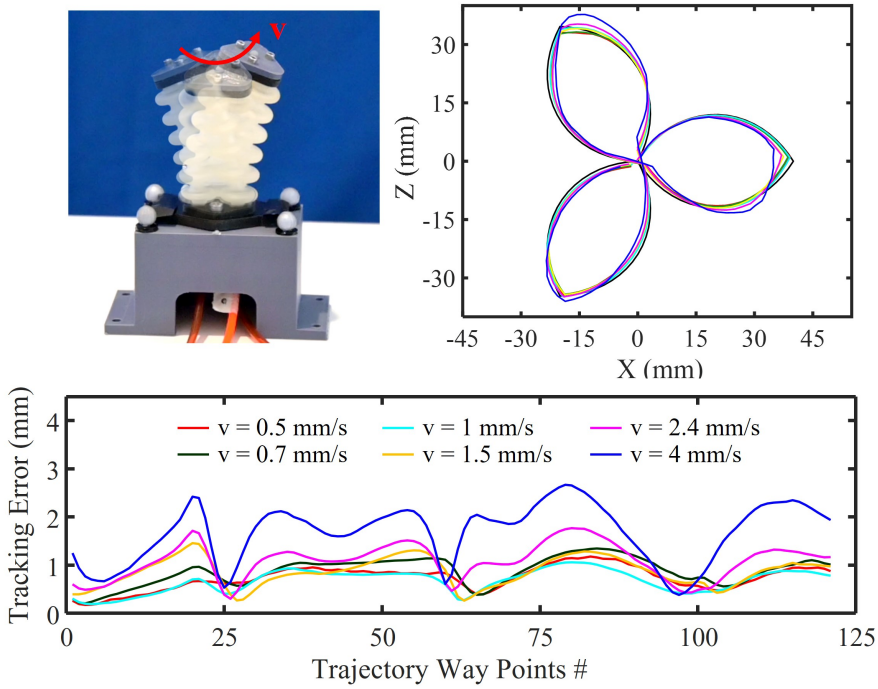


Figure 5.15: Comparison of the behavior in path following at different speeds of motion on the three-chamber soft robot.

accurately compute the IK solution when external loads are added to the end-effector of a soft robot. As shown in Fig. 5.16, significantly enlarged tracking errors can be observed when an external load is directly applied (depicted by red dashed lines). As an additional test, we train a new sim-to-real network by using the soft robot with load. We found that the tracking errors become small again by using this newly trained network. This demonstrates the functionality of the sim-to-real transfer in handling external loads.

Another drawback of this work is that we neglect the pose information in the pipeline. As an important extension of the learning framework, poses of a soft robot (i.e., including orientation) are to be considered [150, 151]. One possible solution is to directly add the rotation into the output layers of $\mathcal{N}_{fk}$ and $\mathcal{N}_J$. Meanwhile, training positions and orientations together needs to consider the balance between their different units. Higher DoFs in actuator space are needed to enhance the feasibility of IK solutions.

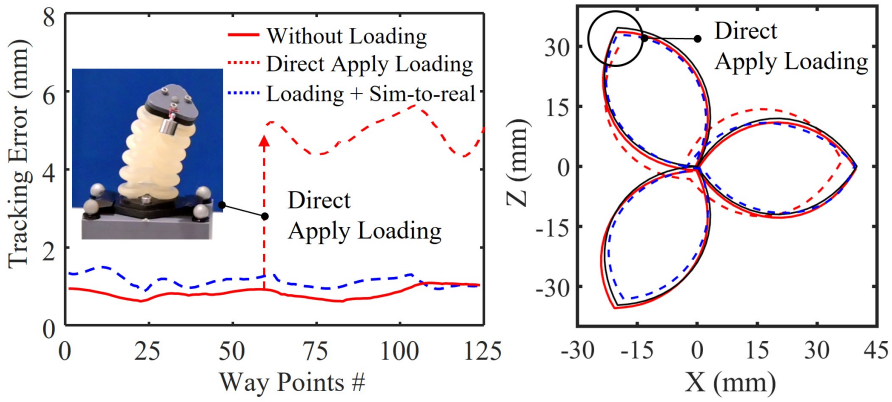


Figure 5.16: Performance of the proposed IK solver when an external load is applied to the end-effector. The black line illustrates the target trajectory.

5.6 Conclusion

In this chapter, we present a method to train the forward kinematic model and its Jacobian together as two neural networks to realize the real-time computation of IK on soft robots, which is formulated as an optimization problem. As our method can generate smooth motion in a redundant system, it outperforms the existing approaches of direct IK learning. Considering the difficulty in generating large datasets on hardware setups, we adopt a highly effective simulator to generate the training datasets and later apply a sim-to-real network to transfer the kinematic model onto hardware. A lightweight network is employed for sim-to-real mapping so that it can be trained by using a small number of samples. This sim-to-real strategy allows our approach to work on different soft robots that have variations caused by materials and fabrication processes.

We test the behavior of our learning-based method in the tasks of path following and interactive positioning on two different soft robotic setups. Our method can solve the IK problem for soft robots effectively and make a good control for the kinematic tasks. As a future work, we plan to explore the possibility of using time-related data for sim-to-real transfer learning that may further enhance the accuracy of IK computing. Moreover, it is also interesting to develop a more transferable learning pipeline that allows the trained model of kinematics can be easily applied to similar designs (e.g., when only the sizes of soft robots are changed).

6

Summary and Future Work

In this chapter, we first summarize the contributions of this Ph.D. research and provide answers to the four research questions presented in Chapter 1. The implications of this Ph.D. project are highlighted from the perspectives of design engineering and sustainability. Finally, discussions on the limitations of this research and possible future work are presented.

6.1 Contributions

This Ph.D. project provides solutions to effectively computing kinematics of soft robots. A fast numerical simulator based on geometry computing is developed and can predict shape changes in a robot body when actuation is applied. The behaviour of soft robots with (self-)collision can be computed with a spring-element assisted collision response model. On the other hand, *inverse kinematics* (IK) of soft robot is solved by Jacobian-based iteration. Control parameters are found to make soft robots accomplish tasks such as trajectory following and pick-and-place. A learning pipeline with sim-to-real transfer is developed to accelerate the IK computing to a real-time speed with improved precision. The answers to the research questions raised in Chapter 1 as the main contributions of this PhD research are presented in the following section.

6.1.1 Answers to the Research Questions

RQ1: *How can a fast and accurate modeling tool be developed to numerically compute the behavior of soft robots under actuation (solve forward kinematics)?*

In Chapter 2, a geometry-based simulator is developed to answer this research question. Based on the observation that actuation applied to soft robot systems are directly related to (or can be transformed into) geometry changes, we hypothesize and verified from the experiment that solving the soft robot modeling problem from a geometry perspective creates a better convergence than the methods based on mechanics. In our simulator, geometry-related elastic energy is developed and optimized using an effective local-global iterative solver. The properties of soft materials are incorporated along with the simulation, where the material parameters are calibrated from physical tests.

The effectiveness of this simulator is verified on soft robot designs, including cable-driven soft finger, pneumatic-driven soft manipulator, and DEA-driven soft swarm robot. Based on our tests, the simulator can effectively predict the performance of soft robots in physical experiments for all setups. An average of 20-fold speedup can be achieved in comparison to FEM simulation by Abaqus software. The geometry-based simulator also presents a good convergence when dealing with large rotational deformations, which was not previously possible with existing methods.

RQ2: *How can we effectively model the behavior of soft robots when self-collision or environment interactions happen during actuation?*

In Chapter 3, a BVH-based collision checking model is developed to enable real-time collision detection of soft robots. AABB tree, as one type of BVH, is used to avoid unnecessary checks of collision pairs during the computation. Based on our test, (self)-collision detection can be completed at 24fps for a mesh that contains 40k tetrahedra. This computational efficiency supports real-time collision checking for all soft robot setups we have tested in this Ph.D. project. After a collision is detected, spring elements are virtually added to the collided vertices to iteratively push them outside the collision region. The spring element is integrated inside the simulator developed in Chapter 2, and a local-global solver is applied to effectively solve the system to minimize the collision energy. As a result, we can always guarantee a collision-free deformed shape being computed as the final solution of the forward kinematics of soft robot, while the simulation speed is not significantly affected.

RQ3: *How can we compute the proper control parameter of a soft robot to achieve the task-specific kinematics control?*

Chapter 4 presents the algorithm for solving IK for soft robots, which answers this research question. The control parameters are detected by minimizing task-specific kinematic objectives through Jacobian-based iteration. This IK solver demonstrates good convergence when cooperating with the line-search method, which can always guarantee a minimum variation in actuator space when planning the motion for a soft robot system with redundancy.

Two case studies are performed in Chapter 4 to verify the effectiveness of the proposed IK solutions. The first task is path-following: a sequence of motion is planned to drive the end-effector of soft robots following waypoints of a given trajectory. The second task is pick-and-place, which is performed by a soft robot manipulator. Both the position and orientation of the end-effector are controlled. In addition, the curvature of the soft finger is controlled to match the grasping object and perform tight grasping. For both tasks, the geometry-based simulator developed in Chapter 2 and Chapter 3 is applied to effectively evaluate the gradient of the objective function by numerical difference.

RQ4: *How can we build a machine learning pipeline to precisely solve the inverse kinematics (IK) of soft robots in real-time speed with an efficient data generation process?*

In Chapter 5, a learning-based IK solver is introduced that answers this research question. Fully connected neural networks are used to approximate the FK function and directly learn the mapping between actuation parameters and Jacobian matrices. By replacing the numerical simulator with forward propagation of networks that computes analytically, the IK solutions are found in

real-time. Meanwhile, a sim-to-real transfer learning process is integrated into this learning-based pipeline to create a more efficient data generation process. The core idea is to train the kinematics mapping in virtual space and then transfer it to the physical setup by using a lightweight sim-to-real network. This not only improves the precision of IK solutions by eliminating the reality gap but also avoids the costly data generation process in soft robot physical setups.

By applying this learning-based IK solver, the time used to compute actuation parameters for a single waypoint in trajectory following tasks is less than 30 ms. The tracking error in the end-effector of soft robots is reduced by 80% compare with the result using the simulation-in-the-loop IK solver as presents in Chapter 4. The tracking error of the trajectory can be controlled within 1% of the workspace of all soft robot setups we have tested. In comparison to existing learning-based methods, this solver performs with similar training accuracy, while only one-third of the physical data needs to be used with the help of sim-to-real transfer.

6.1.2 Implications of the Research

This Ph.D. thesis provides a general solution to the kinematic computing of soft robots. The code for the fast geometry-based simulator and Jacobian-based IK solver is open-source to the public and can be found on Github: `https://github.com/GuoxinFang/SoftRobotKinematics`. In this section, we detail the implications of this Ph.D. project for soft robots from the perspective of design engineering and sustainability.

Design engineering for soft robots

Designing new types of soft robots requires collaboration between designers and scientists since knowledge from material science, biology, computer science, mechanical engineering, and industrial design needs to be brought together. The design process for soft robots is generally time-consuming and heavily relies on experience. Our fast simulator can help accelerate design iterations since designers can test the behavior of soft robots in the virtual environment with less cost. On the other hand, we perceive the thriving of design optimization in the soft robotics research field [32]. The enhancement of grasping stability or maximized motion speed can be achieved by finding a better combination of the shape parameters (or topology) for the existing designs. To automate the design optimization process, an efficient and effective simulation tool is always needed to evaluate the design-related objectives and

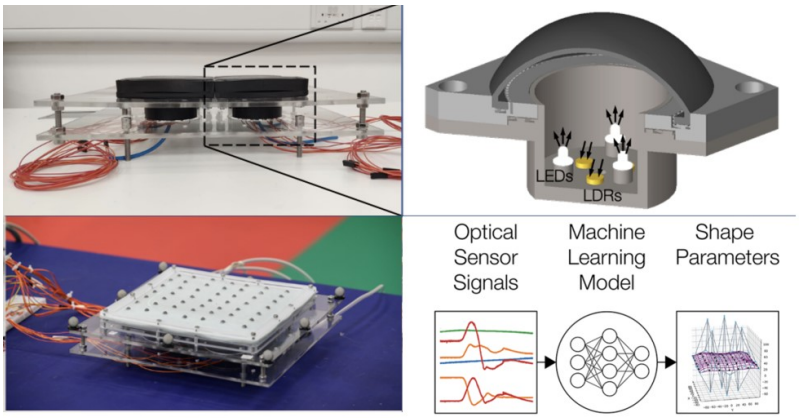


Figure 6.1: Development of a soft robot membrane with an integrated optical sensor. The simulator developed in this thesis is used to evaluate the chamber design and sensor placement before the prototype is fabricated [119].

compute their gradients [152]. We believe that the simulator developed in this work can contribute to this field and facilitate potential future work.

One trial of using the developed fast simulator to guide the soft robot prototype design in the Industrial Design Engineering Faculty of TU Delft is the project of developing a soft membrane with an integrated sensor array. The soft robot design is presented in Fig. 6.1 where optical sensors are assembled inside the chamber. The mapping between sensor signals and shape parameters is learned by networks, which can be used for model-free control [153]. In this project[1], our simulator has been used to help designers figure out proper design parameters for soft membranes (e.g., the radius of the membrane and depth of the cylinder) and find the optimized place for sensor placement. Furthermore, the deformation of the membrane can be effectively predicted by our simulator (the result is presented in Chapter 3.4).

Sustainability with soft robot application

As discussed previously, the simulator developed in this project can help verify soft robot designs in the virtual space before making prototypes. This can

[1]This work is published as: "Sensing and reconstruction of 3D deformation on pneumatic soft robots". Rob B.N. Scharff, **Guoxin Fang**, Yingjun Tian, Jun Wu, Jo M.P. Geraedts, and Charlie C.L. Wang, *IEEE/ASME Transactions on Mechatronics (T-MECH)*, vol.26, no.4, pp.1877-1885, August 2021.

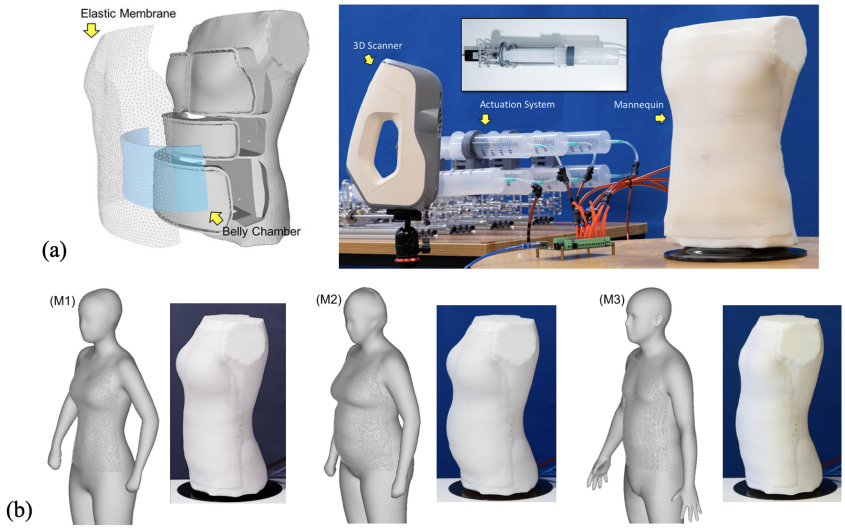


Figure 6.2: (a) Development of a soft robot mannequin that contains inner chambers and outer soft membrane. (b) The soft mannequin can be programmed to fit the given shape of a scanned human body.

largely reduce the wastage of materials used in the fabrication process, such as metals and Polylactic Acid used to fabricate molds and silicon rubber used to fabricate soft robots. Meanwhile, the design optimization process enhances the durability of soft robots in practical applications [32]. For example, the proper selection of materials and structure parameters can allow the local stress to be uniformly distributed under actuation [154]. This allows the use of higher pressures in pneumatic-driven soft robots before they start to leak.

In practice, soft robots can contribute to all kinds of automation processes. Due to their softness, they could be co-workers with humans. Here, we present two examples from agriculture and the garment industry. Firstly, soft grippers are now used to harvest fruits such as apples and strawberries. In this sense, human resources can be reduced, and the waste in the food supply chain can be largely avoided [6]. The kinematics solution developed in this Ph.D. project enables soft manipulators to effectively finish pick-and-place tasks (as studied in Chapter 3). Meanwhile, this kinematic model can be integrated into the closed-loop control of soft grippers. By working together with vision/sensor systems, soft robot proprioception can be realized and utilized to produce smarter robot systems for agricultural applications.

Soft robots can also contribute to the sustainability of the garment industry. A deformable soft robot mannequin has been developed for this purpose[2]. As presented in Fig. 6.2, the mannequin contains inner pneumatic-driven chambers and outer elastic membranes. Compared with conventional rigid-body mannequins with a fixed shape, the soft mannequin setup has the ability to deform and mimic the shapes of human bodies. As illustrated in Fig 6.2(b), the mannequin can be used to effectively mimic three target body shapes within the range of UK sizes 10-14. The soft mannequin can be reused in the fitting and ironing process when making bespoke suits. As a result, the waste of single-used mannequins can be avoided. Additionally, this soft mannequin can become a deformable mold to assist in the manufacturing of sustainable bio-fabricated textiles [155, 156].

6.2 Unsolved Problem and Future Work

Developing a kinematics model for soft robots to enable task-specific control is the main focus of this Ph.D. thesis, which has made technical contributions to soft robot research. The proposed method is limited by the force prediction and dynamics of soft robots, which are not being covered in this study. In this section, we present the unsolved problems and offer possible directions for solutions in the form of future areas of research.

6.2.1 Force Prediction and Dynamics Control

In Research Cycle 2 and Research Cycle 3, we provide a solution to model the behavior of soft robots from a purely geometric perspective. This method can balance computational cost and simulation accuracy. However, it neglects the prediction of contact forces since the system only computes the geometry information (i.e., the shape of each element). Although the collision behaviour is considered in our simulator by a fast collision checking and response model, the contact force cannot be read directly from simulator such as traditional FEA software (e.g., Abaqus) when soft robots interact with obstacles. The lack of force prediction makes the developed simulator limited in its ability to assist with tasks such as force sensing [157, 158].

Analytical models have been developed to predict the contact force of soft

[2]The design is published as: "Soft robotic mannequin: design and algorithm for deformation control", Yingjun Tian, **Guoxin Fang**, Andrew Weightman, and Charlie C.L. Wang, *IEEE/ASME Transactions on Mechatronics (T-MECH)* , accepted, 2022.

actuators [159, 160], which are more efficient but less generally applicable when compared with numerical methods [62, 161]. Studies based on machine learning have also been included to predict the grasping force through sensor or vision data [158, 162]. Predicting contact force is an important future work and could be an extension of this Ph.D. project. One solution could involve first computing the strain on each element based on the computed deformation and then modeling the external contact force by the strain-stress relationship and force equivalence functions [38, 163].

On the other hand, this thesis only demonstrates the performance of the kinematics solution for (quasi-)static tasks for soft robots. As discussed in Sec. 5.5, with physical experiments, the proposed IK solver can easily lose its precision when the dynamic behavior of a soft robot is performed (e.g., the tangent velocity of the soft robot end-effector is higher than 2 mm/s). Including the dynamic model inside a soft robot controller can enable dynamic movements and compliant interactions [164]. Existing works on model-based real time dynamic control for soft robots are mainly based on reduced analytical models [165, 164, 166] since the high computational costs of the exact finite-element models limits their usage in dynamic computing [167].

Learning-based methods fit the dynamic performance with recurrent neural networks and have proved their abilities in the closed-loop dynamic control of soft robots [97, 140, 168]. These are practical solutions, and the efforts spent on physical modeling can be greatly reduced. The complexity of a dynamic network is higher, and data needs to be generated through time sequences; nevertheless, data generation for dynamic training can be more difficult and costly than training a kinematics network. To reduce the burden of the data generation process, the sim-to-real transfer learning introduced in this Ph.D. thesis can also be applied as an interesting future work with extensions from two perspectives. First, the simulator will be enhanced to generate dynamic data. Actuation needs to be applied progressively and the velocity of each vertex can be predicted using the principle of virtual displacements [169]. Second, the better selection of a network structure can facilitate the purpose of sim-to-real transfer of dynamics. For this goal, multi-layer LSTM networks combined with a memory-less feedforward network can be used [170].

6.2.2 Learning for Shape Control in Soft Robotics

As defined in Chapter 1, tasks of soft robots can be conducted for the poses of end-effectors in *task space* or on the whole body shape in *configuration space*. In this thesis, the numerical solver for IK presented in Chapter 4 con-

siders both cases by defining corresponding geometry-related objectives. This solver is strengthened by a learning-based pipeline developed in Chapter 5 to enable real-time IK computing speeds with enhanced accuracy. However, the learning-based solver can only support the kinematics control in *task space* (i.e., only the pose of the end-effector is controlled). The network structure presented in Chapter 5.3 meets the difficulties to extend to whole-body shape prediction since *configuration space* contains extremely high degrees-of-freedom. A fully connected neural network or simple LSTM network cannot effectively learn the connection between *actuator space* and *configuration space* from a small data set. To apply learning-based methods to model and control the shape of soft robots, an effective representation of shape deformation with lower dimensions needs to be developed.

Shape learning has recently become an emerging research topic in the geometric modeling field. Gao *et al* [171] introduced variations of auto-encoders to approximate a sequence of human body deformation from a limited number of shape data as the key-frame. The network trained a low-dimension latent space to represent the mesh, and the deformation was converted into a rotation-invariant mesh difference feature to lower the difficulties in training. A similar encoder-decoder network was developed by Soter *et al* [172] in which 2D images are taken as the input to describe soft robot shape under actuation. This work also presents the usage of recurrent neural network to connect the signal of the bend sensors with the pre-trained latent space. As a result, the sensor signal can be used to predict the whole body shape of a soft manipulator. We believe shape learning can become a useful tool for conducting high-level control in *configuration space*, and our simulator can efficiently generate shape data for the training. Tasks like real-time collision checking, locomotion, and choosing optimum sensor placements for soft robots can all benefit from this learning pipeline.

In Chapter 5, we present the Jacobian learning method to evaluate the gradient of the IK objectives (defined in *task space*) at real-time speeds. We have seen differentiable simulators [103, 173] being conducted that can evaluate Jacobians of objectives defined in *configuration space* by the gradient of simulators in an analytical form. As a result, the process for control and design optimization can be highly accelerated. In this direction, making the geometry-based simulator developed in this Ph.D. project differentiable also represents interesting future work to support direct *configuration space* control of soft robots.

Bibliography

[1] W. Shen, G. Yang, T. Zheng, Y. Wang, K. Yang, and Z. Fang, *An accuracy enhancement method for a cable-driven continuum robot with a flexible backbone,* IEEE Access **8**, 37474 (2020).

[2] A. D. Marchese and D. Rus, *Design, kinematics, and control of a soft spatial fluidic elastomer manipulator,* The International Journal of Robotics Research **35**, 840 (2016).

[3] S. B. Niku, *Introduction to robotics: analysis, control, applications,* John Wiley & Sons (2020).

[4] Y. Ito, J. Hashizume, J. Ikeda, N. Utsumi, and T. Naka, *Teaching-playback robot,* (1983), uS Patent 4,420,812.

[5] R. . A. News, *The role of robotics and automation in industry 4.0,* (2021).

[6] G. Chowdhary, M. Gazzola, G. Krishnan, C. Soman, and S. Lovell, *Soft robotics as an enabling technology for agroforestry practice and research,* Sustainability **11**, 6751 (2019).

[7] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, *Continuum robots for medical applications: A survey,* IEEE Transactions on Robotics **31**, 1261 (2015).

[8] R. Bishop, *A robotic snake arm is here to charge your tesla,* (2015).

[9] D. Rus and M. T. Tolley, *Design, fabrication and control of soft robots,* Nature **521**, 467 (2015).

[10] G. Li, X. Chen, F. Zhou, Y. Liang, Y. Xiao, X. Cao, Z. Zhang, M. Zhang, B. Wu, S. Yin, *et al.*, *Self-powered soft robot in the mariana trench,* Nature **591**, 66 (2021).

[11] C. Laschi, *Robot octopus points the way to soft robotics with eight wiggly arms,* IEEE Spectrum , 1 (2016).

[12] N. L. R. Center, *Beyond the metal: Investigating soft robots at nasa langley,* (2019).

[13] G. Fang, C.-D. Matte, T.-H. Kwok, and C. C. Wang, *Geometry-based direct simulation for multi-material soft robots,* in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 4194–4199.

[14] J. Shintake, S. Rosset, B. Schubert, D. Floreano, and H. Shea, *Versatile soft grippers with intrinsic electroadhesion based on multifunctional polymer actuators,* Advanced materials **28**, 231 (2016).

[15] H. Lu, M. Zhang, Y. Yang, Q. Huang, T. Fukuda, Z. Wang, and Y. Shen, *A bioinspired multilegged soft millirobot that functions in both dry and wet conditions,* Nature communications **9**, 1 (2018).

[16] F. Connolly, P. Polygerinos, C. J. Walsh, and K. Bertoldi, *Mechanical programming of soft actuators by varying fiber angle,* Soft Robotics **2**, 26 (2015).

[17] F. Connolly, C. J. Walsh, and K. Bertoldi, *Automatic design of fiber-reinforced soft actuators for trajectory matching,* Proceedings of the National Academy of Sciences **114**, 51 (2017).

[18] F. Chen, Y. Miao, G. Gu, and X. Zhu, *Soft twisting pneumatic actuators enabled by freeform surface design,* IEEE Robotics and Automation Letters **6**, 5253 (2021).

[19] Y. Chen, H. Chung, B. Chen, H. Y. Ping, and Y. Sun, *Pneumatic actuation-based bidirectional modules with variable stiffness and closed-loop position control,* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021) pp. 6797–6803.

[20] Y. Tang, Y. Chi, J. Sun, T.-H. Huang, O. H. Maghsoudi, A. Spence, J. Zhao, H. Su, and J. Yin, *Leveraging elastic instabilities for amplified performance: Spine-inspired high-speed and high-force soft robots,* Science advances **6**, eaaz6912 (2020).

[21] Aceo3D, *Customized robot gripper designs using 3d printing with silicone,* (2019).

[22] FESTO, *Bionic handling assistant,* (2012).

[23] N. R. Sinatra, C. B. Teeple, D. M. Vogt, K. K. Parker, D. F. Gruber, and R. J. Wood, *Ultragentle manipulation of delicate structures using a soft robotic gripper,* Science Robotics **4** (2019).

[24] R. B. Scharff, E. L. Doubrovski, W. A. Poelman, P. P. Jonker, C. C. Wang, and J. M. P. Geraedts, *Towards behavior design of a 3d-printed soft robotic hand,* in *Soft Robotics: Trends, Applications and Challenges* (Springer, 2017) pp. 23–29.

[25] N. El-Atab, R. B. Mishra, F. Al-Modaf, L. Joharji, A. A. Alsharif, H. Alamoudi, M. Diaz, N. Qaiser, and M. M. Hussain, *Soft actuators for soft robotic applications: a review,* Advanced Intelligent Systems **2**, 2000128 (2020).

[26] M. Follador, M. Cianchetti, A. Arienti, and C. Laschi, *A general method for the design and fabrication of shape memory alloy active spring actuators,* Smart Materials and Structures **21**, 115029 (2012).

[27] A. T. Asbeck, S. M. De Rossi, I. Galiana, Y. Ding, and C. J. Walsh, *Stronger, smarter, softer: next-generation wearable robots,* IEEE Robotics & Automation Magazine **21**, 22 (2014).

[28] J. Nassour, G. Zhao, and M. Grimmer, *Soft pneumatic elbow exoskeleton reduces the muscle activity, metabolic cost and fatigue during holding and carrying of loads,* Scientific Reports **11**, 1 (2021).

[29] M. Runciman, A. Darzi, and G. P. Mylonas, *Soft robotics in minimally invasive surgery,* Soft robotics **6**, 423 (2019).

[30] G. Fang, M. C. Chow, J. D. Ho, Z. He, K. Wang, T. Ng, J. K. Tsoi, P.-L. Chan, H.-C. Chang, D. T.-M. Chan, *et al.*, *Soft robotic manipulator for intraoperative mri-guided transoral laser microsurgery,* Science Robotics **6**, eabg5575 (2021).

[31] S. Coyle, C. Majidi, P. LeDuc, and K. J. Hsia, *Bio-inspired soft robotics: Material selection, actuation, and design,* Extreme Mechanics Letters **22**, 51 (2018).

[32] F. Chen and M. Y. Wang, *Design optimization of soft robots: A review of the state of the art,* IEEE Robotics & Automation Magazine (2020).

[33] F. Schmitt, O. Piccin, L. Barbé, and B. Bayle, *Soft robots manufacturing: A review,* Frontiers in Robotics and AI **5**, 84 (2018).

[34] S. S. Rao, *The finite element method in engineering* (Butterworth-heinemann, 2017).

[35] 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 **29**, 1661 (2010).

[36] C. Duriez, E. Coevoet, F. Largilliere, T. Morales-Bieze, Z. Zhang, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, and J. Dequidt, *Framework for online simulation of soft robots with optimization-based inverse model,* in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (IEEE, 2016) pp. 111–118.

[37] C. Armanini, I. Hussain, M. Z. Iqbal, D. Gan, D. Prattichizzo, and F. Renda, *Discrete cosserat approach for closed-chain soft robots: Application to the fin-ray finger,* IEEE Transactions on Robotics **37**, 2083 (2021).

[38] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, *SOFA: A Multi-Model Framework for Interactive Physical Simulation,* in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery* (Springer, 2012) pp. 283–321.

[39] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, *Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with non-constant curvature,* IEEE Transactions on Robotics **31**, 823 (2015).

[40] R. B. N. Scharff, R. M. Doornbusch, E. L. Doubrovski, J. Wu, J. M. P. Geraedts, and C. C. Wang, *Color-based proprioception of soft actuators interacting with objects,* IEEE/ASME Transactions on Mechatronics **24**, 1964 (2019).

[41] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, *A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space,* in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013) pp. 5033–5039.

[42] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, *Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with non-constant curvature,* IEEE Trans. Robot. **31**, 823 (2015).

[43] R. Grassmann, V. Modes, and J. Burgner-Kahrs, *Learning the forward and inverse kinematics of a 6-dof concentric tube continuum robot in se(3),* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2018) pp. 5125–5132.

[44] G. Fang, Y. Tian, Z.-X. Yang, J. M. P. Geraedts, and C. C. Wang, *Efficient jacobian-based inverse kinematics of soft robots by learning,* IEEE/ASME Transactions on Mechatronics, accepted (2022).

[45] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, *Continuum robots for medical applications: A survey,* IEEE Trans. Robot. **31**, 1261 (2015).

[46] R. Katzschmann, A. D. Marchese, and D. Rus, *Autonomous object manipulation using a soft planar grasping manipulator,* Soft Robotics. **2**, 155 (2015).

[47] A. D. Marchese and D. Rus, *Design, kinematics, and control of a soft spatial fluidic elastomer manipulator,* Int. J. Robot. Res. **35**, 840 (2016).

[48] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross, *Computational design of actuated deformable characters,* ACM Trans. Graph. **32** (2013).

[49] N. Bartlett, M. Tolley, J. Overvelde, J. Weaver, B. Mosadegh, K. Bertoldi, G. M Whitesides, and R. Wood, *A 3d-printed, functionally graded soft robot powered by combustion,* Science. **349** (2015).

[50] D. Drotman, S. Jadhav, M. Karimi, P. deZonia, and M. T. Tolley, *3d printed soft actuators for a legged robot capable of navigating unstructured terrain,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2017).

[51] R. B. Scharff, R. M. Doornbusch, X. L. Klootwijk, A. A. Doshi, E. L. Doubrovski, J. Wu, J. M. P. Geraedts, and C. C. Wang, *Color-based sensing of bending deformation on soft robots,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2018).

[52] K. H. Lee, M. C. W. Leong, M. C. K. Chow, H. C. Fu, W. Luk, K. Y. Sze, C. K. Yeung, and K. W. Kwok, *Fem-based soft robotic control framework for intracavitary navigation,* in *IEEE Int. Conf. Real-time Computing and Robotics* (2017).

[53] C. Duriez, *Control of elastic soft robots based on real-time finite element method,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2013) pp. 3982–3987.

[54] J. Cao, W. Liang, Y. Wang, H. P. Lee, J. Zhu, and Q. Ren, *Control of a soft inchworm robot with environment adaptation,* IEEE Transactions on Industrial Electronics , 1 (2019).

[55] S. Tokumoto and S. Hirai, *Deformation control of rheological food dough using a forming process model,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2002) pp. 1457–1464.

[56] E. T. Roche, M. A. Horvath, I. Wamala, A. Alazmani, S.-E. Song, W. Whyte, Z. Machaidze, C. J. Payne, J. C. Weaver, G. Fishbein, J. Kuebler, N. V. Vasilyev, D. J. Mooney, F. A. Pigula, and C. J. Walsh, *Soft robotic sleeve supports heart function,* Sci. Transl. Med. **9** (2017).

[57] G. S. Chirikjian and J. W. Burdick, *A modal approach to hyper-redundant manipulator kinematics,* IEEE Trans. Robot. Autom. **10**, 343 (1994).

[58] B. A. Jones and I. D. Walker, *Kinematics for multisection continuum robots,* IEEE Trans. Robot. **22**, 43 (2006).

[59] D. Trivedi, A. Lotfi, and C. D. Rahn, *Geometrically exact models for soft robotic manipulators,* IEEE Trans. Robot. **24**, 773 (2008).

[60] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, *A two dimensional inverse kinetics model of a cable driven manipulator inspired by the octopus arm,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2012) pp. 3819–3824.

[61] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, *Dynamic model of a multibending soft robot arm driven by cables,* IEEE Trans. Robot. **30**, 1109 (2014).

[62] P. Polygerinos, Z. Wang, J. T. B. Overvelde, K. C. Galloway, R. J. Wood, K. Bertoldi, and C. J. Walsh, *Modeling of soft fiber-reinforced bending actuators,* IEEE Trans. Robot. **31**, 778 (2015).

[63] Z. Wang and S. Hirai, *Soft gripper dynamics using a line-segment model with an optimization-based parameter identification method,* IEEE Robot. Autom. Lett. **2**, 624 (2017).

[64] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, *Pneumatic networks for soft robotics that actuate rapidly,* Adv. funct. Mater. **24**, 2163 (2014).

[65] P. Moseley, J. M. Florez, H. A. Sonar, G. Agarwal, W. Curtin, and J. Paik, *Modeling, design, and development of soft pneumatic actuators with finite element method,* Adv. Eng. Mater. **18** (2016).

[66] A. A. Stanley and A. M. Okamura, *Deformable model-based methods for shape control of a haptic jamming surface,* IEEE Trans. Vis. Comput. Graph. **23**, 1029 (2016).

[67] J. Hiller and H. Lipson, *Dynamic simulation of soft multimaterial 3d-printed objects,* Soft Robotics. **1**, 88 (2014).

[68] C. Duriez, E. Coevoet, F. Largilliere, T. Morales-Bieze, Z. Zhang, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, and J. Dequidt, *Framework for online simulation of soft robots with optimization-based inverse model,* in *Proc. IEEE Int. Conf. Simulation, Modeling, and Programming for Autonomous Robots.* (2016) pp. 111–118.

[69] O. Goury and C. Duriez, *Fast, generic, and reliable control and simulation of soft robots using model order reduction,* IEEE Trans. Robot. **34**, 1565 (2018).

[70] K.-H. Lee, D. K. Fu, M. C. Leong, M. Chow, H.-C. Fu, K. Althoefer, K. Y. Sze, C.-K. Yeung, and K.-W. Kwok, *Nonparametric online learning control for soft continuum robot: An enabling technique for effective endoscopic navigation,* Soft Robotics. **4** (2017).

[71] D. Navarro-Alarcón, Y. H. Liu, J. G. Romero, and P. Li, *Model-free visually servoed deformation control of elastic objects by robot manipulators,* IEEE Trans. Robot. **29**, 1457 (2013).

[72] D. Navarro-Alarcon and Y. H. Liu, *Fourier-based shape servoing: A new feedback method to actively deform soft objects into desired 2-d image contours,* IEEE Trans. Robot. **34**, 272 (2018).

[73] M. Li, R. Kang, D. T. Branson, and J. S. Dai, *Model-free control for continuum robots based on an adaptive kalman filter,* IEEE/ASME Trans. Mechatronics. **23**, 286 (2018).

[74] Z. Zhang, T. M. Bieze, J. Dequidt, A. Kruszewski, and C. Duriez, *Visual servoing control of soft robots based on finite element model,* in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* (2017) pp. 2895–2901.

[75] O. Sorkine and M. Alexa, *As-rigid-as-possible surface modeling,* in *Proc. Eurographics Symposium on Geometry Processing* (2007).

[76] M. Shapira and A. Rappoport, *Shape blending using the star-skeleton representation,* IEEE Computer Graphics and Applications. **15**, 44 (1995).

[77] T. Hassan, M. Manti, G. Passetti, N. d'Elia, M. Cianchetti, and C. Laschi, *Design and development of a bio-inspired, under-actuated soft gripper,* in *IEEE Int. Conf. Engineering in Medicine and Biology Society* (2015) pp. 3619–3622.

[78] R. Pelrine, R. Kornbluh, Q. Pei, and J. Joseph, *High-speed electrically actuated elastomers with strain greater than 100%,* Science. **287**, 836 (2000).

[79] R. B. Scharff, J. Wu, J. M. P. Geraedts, and C. C. Wang, *Reducing out-of-plane deformation of soft robotic actuators for stable grasping,* in *IEEE Int. Conf. Soft Robotics* (2019) pp. 265–270.

[80] D. Drotman, M. Ishida, S. Jadhav, and M. T. Tolley, *Application-driven design of soft, 3d printed, pneumatic actuators with bellows,* IEEE/ASME Trans. Mechatronics. (2018), 10.1109/TMECH.2018.2879299.

[81] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly, *Shape-up: Shaping discrete geometry with projections,* Comput. Graph. Forum. **31**, 1657 (2012).

[82] *Eigen v3.3,* http://eigen.tuxfamily.org (2017).

[83] J. Cao, W. Liang, Q. Ren, U. Gupta, F. Chen, and J. Zhu, *Modelling and control of a novel soft crawling robot based on a dielectric elastomer actuator,* in *Proc. IEEE Int. Conf. Robot. Autom.* (2018) pp. 4188–4193.

[84] T. Lu, J. Huang, C. Jordi, G. Kovacs, R. Huang, D. R. Clarke, and Z. Suo, *Dielectric elastomer actuators under equal-biaxial forces, uniaxial forces, and uniaxial constraint of stiff fibers,* Soft Matter **8**, 6167 (2012).

[85] T. Kalisky, Y. Wang, B. Shih, D. Drotman, S. Jadhav, E. Aronoff-Spencer, and M. T. Tolley, *Differential pressure control of 3d printed soft fluidic actuators,* Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. , 6207 (2017).

[86] D. Rus and M. T. Tolley, *Design, fabrication and control of soft robots,* Nature **521**, 467 (2015).

[87] K. C. Galloway, K. P. Becker, B. Phillips, J. Kirby, S. Licht, D. Tchernov, R. J. Wood, and D. F. Gruber, *Soft robotic grippers for biological sampling on deep reefs,* Soft robotics **3**, 23 (2016).

[88] R. Deimel and O. Brock, *A novel type of compliant and underactuated robotic hand for dexterous grasping,* The International Journal of Robotics Research **35**, 161 (2016).

[89] A. D. Marchese and D. Rus, *Design, kinematics, and control of a soft spatial fluidic elastomer manipulator,* The International Journal of Robotics Research **35**, 840 (2016).

[90] A. D. Marchese, R. K. Katzschmann, and D. Rus, *A recipe for soft fluidic elastomer robots,* Soft robotics **2**, 7 (2015).

[91] T. Morales Bieze, A. Kruszewski, B. Carrez, and C. Duriez, *Design, implementation, and control of a deformable manipulator robot based on a compliant spine,* The International Journal of Robotics Research **39**, 1604 (2020).

[92] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, *Pneumatic networks for soft robotics that actuate rapidly,* Advanced functional materials **24**, 2163 (2014).

[93] 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 **6**, 3752 (2021).

[94] P. Moseley, J. M. Florez, H. A. Sonar, G. Agarwal, W. Curtin, and J. Paik, *Modeling, design, and development of soft pneumatic actuators with finite element method,* Advanced engineering materials **18**, 978 (2016).

[95] F. Chen, W. Xu, H. Zhang, Y. Wang, J. Cao, M. Y. Wang, H. Ren, J. Zhu, and Y. Zhang, *Topology optimized design, fabrication, and characterization of a soft cable-driven gripper,* IEEE Robotics and Automation Letters **3**, 2463 (2018).

[96] Y. Chen, Z. Xia, and Q. Zhao, *Optimal design of soft pneumatic bending actuators subjected to design-dependent pressure loads,* IEEE/ASME Transactions on Mechatronics **24**, 2873 (2019).

[97] 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 **35**, 124 (2018).

[98] G. Zhong, W. Dou, X. Zhang, and H. Yi, *Bending analysis and contact force modeling of soft pneumatic actuators with pleated structures,* International Journal of Mechanical Sciences **193**, 106150 (2021).

[99] M. Yang, L. P. Cooper, N. Liu, X. Wang, and M. P. Fok, *Twining plant inspired pneumatic soft robotic spiral gripper with a fiber optic twisting sensor,* Optics Express **28**, 35158 (2020).

[100] G. Fang, C.-D. Matte, R. B. Scharff, T.-H. Kwok, and C. C. Wang, *Kinematics of soft robots by geometric computing,* IEEE Transactions on Robotics **36**, 1272 (2020).

[101] P. Polygerinos, Z. Wang, J. T. Overvelde, K. C. Galloway, R. J. Wood, K. Bertoldi, and C. J. Walsh, *Modeling of soft fiber-reinforced bending actuators,* IEEE Transactions on Robotics **31**, 778 (2015).

[102] J. Hiller and H. Lipson, *Dynamic simulation of soft multimaterial 3d-printed objects,* Soft robotics **1**, 88 (2014).

[103] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, *Chainqueen: A real-time differentiable physical simulator for soft robotics,* in *2019 International conference on robotics and automation (ICRA)* (IEEE, 2019) pp. 6265–6271.

[104] M. A. Graule, C. B. Teeple, T. P. McCarthy, G. R. Kim, R. C. S. Louis, and R. J. Wood, *Somo: Fast and accurate simulations of continuum robots in complex environments,* in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021) pp. 3934–3941.

[105] C. Duriez, *Control of elastic soft robots based on real-time finite element method,* in *2013 IEEE international conference on robotics and automation* (IEEE, 2013) pp. 3982–3987.

[106] O. Goury and C. Duriez, *Fast, generic, and reliable control and simulation of soft robots using model order reduction,* IEEE Transactions on Robotics **34**, 1565 (2018).

[107] D. Dinev, T. Liu, J. Li, B. Thomaszewski, and L. Kavan, *Fepr: Fast energy projection for real-time simulation of deformable objects,* ACM Transactions on Graphics (TOG) **37**, 1 (2018).

[108] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, *Projective dynamics: Fusing constraint projections for fast simulation,* ACM Trans. Graph. **33** (2014).

[109] J. Li, T. Liu, and L. Kavan, *Soft articulated characters in projective dynamics,* IEEE Transactions on Visualization and Computer Graphics (2020).

[110] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Thalmann, W. Straßer, and P. Volino, *Collision detection for deformable objects,* Comput. Graph. Forum **24**, 61 (2005).

[111] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, *Volume contact constraints at arbitrary resolution,* ACM Trans. Graph. **29** (2010).

[112] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, *Elastically deformable models,* in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87 (Association for Computing Machinery, New York, NY, USA, 1987) p. 205–214.

[113] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, *Asynchronous contact mechanics,* (ACM, New York, NY, USA, 2009).

[114] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, *Fast simulation of mass-spring systems,* ACM Transactions on Graphics **32**, 209:1 (2013).

[115] C.-D. Matte and T.-H. Kwok, *Simulation of Hyperelasticity by Shape Estimation,* Journal of Computing and Information Science in Engineering **21** (2021), 050903.

[116] G. van den Bergen, *Efficient collision detection of complex deformable models using aabb trees,* J. Graph. Tools **2**, 1–13 (1998).

[117] S. Gottschalk, M. C. Lin, and D. Manocha, *Obbtree: A hierarchical structure for rapid interference detection,* in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96 (ACM, New York, NY, USA, 1996) pp. 171–180.

[118] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, and L. Liu, *Anderson acceleration for geometry optimization and physics simulation,* ACM Transactions on Graphics (TOG) **37**, 1 (2018).

[119] R. B. Scharff, G. Fang, Y. Tian, J. Wu, J. M. P. Geraedts, and C. C. Wang, *Sensing and reconstruction of 3-d deformation on pneumatic soft robots,* IEEE/ASME Transactions on Mechatronics **26**, 1877 (2021).

[120] F. Labelle and J. R. Shewchuk, *Isosurface stuffing: fast tetrahedral meshes with good dihedral angles,* in *ACM SIGGRAPH 2007 papers* (2007) pp. 57–es.

[121] S. Buss, *Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods,* IEEE Trans. Robot. Autom. **17** (2004).

[122] H. P. Moreton and C. H. Séquin, *Functional optimization for fair surface design,* SIGGRAPH Comput. Graph. **26**, 167 (1992).

[123] *Marlin firmware,* http://marlinfw.org (2018).

[124] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, *Robotic grasping of novel objects,* in *Advances in Neural Information Processing Systems* (2007) pp. 1209–1216.

[125] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, and C. J. Walsh, *Soft robotic glove for combined assistance and at-home rehabilitation,* Robotics and Autonomous Systems **73**, 135 (2015).

[126] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, *Adaptive neural network control of a compact bionic handling arm,* IEEE/ASME Transactions on Mechatronics **20**, 2862 (2015).

[127] T. Ranzani, G. Gerboni, M. Cianchetti, and A. Menciassi, *A bioinspired soft manipulator for minimally invasive surgery,* Bioinspiration & Biomimetics **10**, 035008 (2015).

[128] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, *Control strategies for soft robotic manipulators: A survey,* Soft Robotics **5**, 149 (2018), pMID: 29297756.

[129] M. Rolf and J. J. Steil, *Constant curvature continuum kinematics as fast approximate model for the bionic handling assistant,* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012) pp. 3440–3446.

[130] P. Moseley, J. M. Florez, H. A. Sonar, G. Agarwal, W. Curtin, and J. Paik, *Modeling, design, and development of soft pneumatic actuators with finite element method,* Advanced engineering materials **18**, 978 (2016).

[131] M. S. Xavier, A. J. Fleming, and Y. K. Yong, *Finite element modeling of soft fluidic actuators: Overview and recent developments,* Advanced Intelligent Systems **3**, 2000187 (2021).

[132] D. E. Orin and W. W. Schrader, *Efficient computation of the jacobian for robot manipulators,* The International Journal of Robotics Research **3**, 66 (1984).

[133] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, *A feed forward neural network for solving the inverse kinetics of non-constant curvature soft manipulators driven by cables,* in *Dynamic Systems and Control Conference*, Vol. 56147 (American Society of Mechanical Engineers, 2013) p. V003T38A001.

[134] M. Rolf and J. J. Steil, *Efficient exploratory learning of inverse kinematics on a bionic elephant trunk,* IEEE Transactions on Neural Networks and Learning Systems **25**, 1147 (2014).

[135] J. Chen and H. Y. K. Lau, *Learning the inverse kinematics of tendon-driven soft manipulators with k-nearest neighbors regression and gaussian mixture regression,* in *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)* (2016) pp. 103–107.

[136] R. F. Reinhart, Z. Shareef, and J. J. Steil, *Hybrid analytical and data-driven modeling for feed-forward robot control,* Sensors **17**, 311 (2017).

[137] T. G. Thuruthel., E. Falotico., M. Cianchetti., F. Renda., and C. Laschi., *Learning global inverse statics solution for a redundant soft robot,* in *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,*, INSTICC (SciTePress, 2016) pp. 303–310.

[138] T. G. Thuruthel, E. Falotico, M. Cianchetti, and C. Laschi, *Learning global inverse kinematics solutions for a continuum robot,* in *Symposium on Robot Design, Dynamics and Control* (Springer, 2016) pp. 47–54.

[139] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, *Soft robot control with a learned differentiable model,* in *2020 IEEE International Conference on Soft Robotics (RoboSoft)* (2020) pp. 417–423.

[140] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, *Soft robot perception using embedded soft sensors and recurrent neural networks,* Science Robotics **4** (2019), 10.1126/scirobotics.aav1488.

[141] Y. Sun, Y. S. Song, and J. Paik, *Characterization of silicone rubber based soft pneumatic actuators,* in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013) pp. 4446–4453.

[142] D. Kubus, R. Rayyes, and J. J. Steil, *Learning forward and inverse kinematics maps efficiently,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018) pp. 5133–5140.

[143] E. S. Marquez, J. S. Hare, and M. Niranjan, *Deep cascade learning,* IEEE Transactions on Neural Networks and Learning Systems **29** (2018), 10.1109/TNNLS.2018.2805098.

[144] S. Kriegman, A. M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, and R. Kramer-Bottiglio, *Scalable sim-to-real transfer of soft robot designs,* in *IEEE International Conference on Soft Robotics* (2020) pp. 359–366.

[145] H. Park, J. Cho, J. Park, Y. Na, and J. Kim, *Sim-to-real transfer learning approach for tracking multi-dof ankle motions using soft strain sensors,* IEEE Robotics and Automation Letters **5**, 3525 (2020).

[146] H. Donat, S. Lilge, J. Burgner-Kahrs, and J. J. Steil, *Estimating tip contact forces for concentric tube continuum robots based on backbone deflection,* IEEE Trans. Med. Robot. Bionics **2** (2020), 10.1109/TMRB.2020.3034258.

[147] M. S. Malekzadeh, S. Calinon, D. Bruno, and D. G. Caldwell, *Learning by imitation with the stiff-flop surgical robot: a biomimetic approach inspired by octopus movements,* Robotics and Biomimetics **1**, 1 (2014).

[148] M. Wiese, G. Runge, B.-H. Cao, and A. Raatz, *Transfer learning for accurate modeling and control of soft actuators,* in *2021 IEEE International Conference on Soft Robotics (RoboSoft)* (2021).

[149] 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 **35**, 124 (2019).

[150] R. Grassmann and J. Burgner-Kahrs, *On the merits of joint space and orientation representations in learning the forward kinematics in se (3).* in *Robotics: science and systems* (2019).

[151] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, *A Smooth Representation of SO(3) for Deep Rotation Learning with Uncertainty,* in *Proceedings of Robotics: Science and Systems (RSS'20)* (2020).

[152] F. Chen, K. Liu, Q. Pan, S. Chen, and X. Zhu, *An integrated design and fabrication strategy for planar soft dielectric elastomer actuators,* IEEE/ASME Transactions on Mechatronics **26**, 2629 (2020).

[153] R. B. Scharff, R. M. Doornbusch, E. L. Doubrovski, J. Wu, J. M. P. Geraedts, and C. C. Wang, *Color-based proprioception of soft actuators interacting with objects,* IEEE/ASME Transactions on Mechatronics **24**, 1964 (2019).

[154] G. Dämmer, S. Gablenz, A. Hildebrandt, and Z. Major, *Design and shape optimization of polyjet bellows actuators,* in *2018 IEEE International Conference on Soft Robotics (RoboSoft)* (IEEE, 2018) pp. 282–287.

[155] S. S. Muthu and M. A. Gardetti, *Sustainability in the Textile and Apparel Industries* (Springer, 2020).

[156] J. Zhou, B. Barati, J. Wu, D. Scherer, and E. Karana, *Digital biofabrication to realize the potentials of plant roots for product design,* Bio-Design and Manufacturing **4**, 111 (2021).

[157] A. Buso, R. B. Scharff, E. L. Doubrovski, J. Wu, C. C. Wang, and P. Vink, *Soft robotic module for sensing and controlling contact force,* in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)* (IEEE, 2020) pp. 70–75.

[158] D. De Barrie, M. Pandya, H. Pandya, M. Hanheide, and K. Elgeneidy, *A deep learning method for vision based force prediction of a soft fin ray gripper using simulation data,* Frontiers in Robotics and AI **8**, 104 (2021).

[159] Z. Liu, F. Wang, S. Liu, Y. Tian, and D. Zhang, *Modeling and analysis of soft pneumatic network bending actuators,* IEEE/ASME Transactions on Mechatronics **26**, 2195 (2020).

[160] Z. Wang, S. Hirai, *et al.*, *Analytical modeling of a soft pneu-net actuator subjected to planar tip contact,* IEEE Transactions on Robotics (2022).

[161] Y. Hao, T. Wang, Z. Ren, Z. Gong, H. Wang, X. Yang, S. Guan, and L. Wen, *Modeling and experiments of a soft robotic gripper in amphibious environments,* International Journal of Advanced Robotic Systems **14**, 1729881417707148 (2017).

[162] B. W. K. Ang and C.-H. Yeow, *A learning-based approach to sensorize soft robots,* Soft Robotics (2022).

[163] Z. Zhang, A. Petit, J. Dequidt, and C. Duriez, *Calibration and external force sensing for soft robots using an rgb-d camera,* IEEE Robotics and Automation Letters **4**, 2356 (2019).

[164] 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 **39**, 490 (2020).

[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)* (IEEE, 2018) pp. 46–53.

[166] M. Azizkhani, I. S. Godage, and Y. Chen, *Dynamic control of soft robotic arm: A simulation study,* IEEE Robotics and Automation Letters **7**, 3584 (2022).

[167] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, *Dynamic simulation of articulated soft robots,* Nature communications **11**, 1 (2020).

[168] T. George Thuruthel, F. Renda, and F. Iida, *First-order dynamic modeling and control of soft robots,* Frontiers in Robotics and AI **7**, 95 (2020).

[169] T. J. Hughes, *The finite element method: linear static and dynamic finite element analysis* (Courier Corporation, 2012).

[170] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, *Sim-to-real transfer of robotic control with dynamics randomization,* in *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2018) pp. 3803–3810.

[171] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia, *Variational autoencoders for deforming 3d mesh models,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) pp. 5841–5850.

[172] G. Soter, A. Conn, H. Hauser, and J. Rossiter, *Bodily aware soft robots: integration of proprioceptive and exteroceptive sensors,* in *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2018) pp. 2448–2453.

[173] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus, *Underwater soft robot modeling and control with differentiable simulation,* IEEE Robotics and Automation Letters **6**, 4994 (2021).

Acknowledgments

My Ph.D. journey has finally come to an end, and it has been the most interesting and adventurous experience I have ever had. I have received a great deal of support from my supervision team, colleagues, family, and friends.

I would like to first give my greatest thanks to my promoters, Prof. Charlie Wang and Prof. Jo Geraedts. They give me the freedom to choose research directions and are always supportive when I encountered both academic and personal challenges. Charlie gives me the opportunity to start my academic career and always requires the highest standard of research. His ideas and comments are always inspiring and proves to be true. Jo's feedback on my research caused me to jump out of the comfort zone and inspired me to view the research from another perspective.

I also want to thanks Dr. Kwok Tsz-Ho for guidance during my visit to Concordia University in the summer of 2017. You are the senior in academics and acted as my supervisor at the beginning of my Ph.D. journey. I will always remember what you taught me about coding and proofing mathematics equations. These skills and knowledge make me more confident and well-prepared for the research.

Special thanks should also go to my committee members Prof. Kaspar Jansen, Prof. Andrew Weightman, Prof. Elmar Eisemann, Prof. Yu Sun, Prof. Peter Vink and Dr. Yu (Wolf) Song for all their guidance. It's also my pleasure to work together with Prof. Emily Whiting, Prof. Sylvain Lefebvre, Prof. Xiujie Wang, Prof. Yongjing Liu, Prof. Shane Xie, Prof. Zichun Zhong, and Dr. Jun Wu.

I am thankful to Dr. Chengkai Dai and Dr. Chengming Wu for the great time that we spent working together on robot-assisted additive manufacturing projects. The outcomes of these projects have resulted in the highest cited papers on my publication list and have inspired me to pursue other works. I would especially like to thank Dr. Rob Scharff, who bought me to the soft robotics research field and offered me help after I joined TU Delft. I would also like to thank Dr. Xiaoting Zhang for giving me the opportunity to work on a few SIGGRAPH projects. I also appreciate Christopher-Denny Matte and the time we worked together in Canada.

I would like to thank my lab colleagues, Tianyu Zhang, Yingjun Tian, and Xiangjia Chen for their support and the time we spent working together during

the coronavirus pandemic. Thanks should also be given to Zishun Liu, Dr. Bin Liu, Yabin Xu, Yuming Huang, Dr. Tim Kuipers, Yinan Meng, Renbo Su, Yuhu Guo and Xilong Wang for all the valuable discussions and help they provide during my PhD journey. Furthermore, I would like to thanks my collaborators, Dr. Qingqing shi, Agathe Balayn, Zeyu Zhang, Tingxiang Fan, Dr. Jiawei Cao, Dr. Zhenhong Li, and Sikai Zhong.

I also want to give my thanks for the support from my friends. I offer many thanks to Yuxuan Kang for the discussion we had when I was working on the project to develop the soft robot simulator. I also offer my thanks to Xiangwei Shi, Dr. Cha Li, Dr. Zhongchen Li, Zerui Zhang, Aoge Hu, and Zheyuan Liu for all the great time and travel experiences we shared.

I am very grateful to my family members for their financial and mental support. Thanks, Mom and Dad, for supporting my decisions and tolerating the negative emotions I have. I also want to thank to my fiancée, Ying Yu for the accomplishment of spending more than four years together even though I'm not always in the Netherlands with her. We can continue exploring the future together.

I would like to thank all the professors, colleagues, technical staffs, and friends I meet at the Chinese Academy of Science, Tsinghua University, Concordia University, TU Delft, the University of Manchester, and the Chinese University of Hong Kong. Finally, I would like to thank the *Chinese Scholarship Council* (CSC) for supporting part of my Ph.D. study.

Guoxin Fang

La Palma, Spain, May 2022

Curriculum Vitæ

Guoxin FANG

08 Aug 1996 Born in Jingdezhen, Jiangxi Province, China

Education

2016–2022 **PhD in the Materializing Futures section, IDE**
Delft University of Technology, The Netherlands
Thesis: Kinematics Computing for Soft Robots
Promotors: Prof. Charlie C.L. Wang
Prof. Jo M.P. Geraedts

2012–2016 **BEng in Mechanical Engineering**
Beijing Institute of Technology, China
Awarded as Outstanding Graduates

Research Experience

2021-2022 **Research Assistant**
Department of Mechanical, Aerospace and Civil Engineering
The University of Manchester, United Kingdom

2018-2020 **Junior Research Assistant**
Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong. Hong Kong, China

2017 **Summer Research Intern**
Department of Mechanical, Industrial and Aerospace Engineering. Concordia University, Montreal, Quebec, Canada
Supervised by *Dr. Tsz-Ho Kwok*

2015-2016 **Research Intern**
Institute of Genetics and Developmental Biology
Chinese Academy of Sciences. Beijing, China
Supervised by *Prof. Xiujie Wang*

Publications

Publications related with soft robotics

1. **Guoxin Fang**, Yingjun Tian, Zhi-Xin Yang, Jo M.P. Geraedts, and Charlie C.L. Wang, "Efficient Jacobian-based Inverse Kinematics with Sim-to-Real Transfer of soft robots by Learning", *IEEE/ASME Transactions on Mechatronics (T-MECH)*, accepted, 2022.

2. **Guoxin Fang**, Yingjun Tian, Andrew Weightman, and Charlie C.L. Wang, "Collision-Aware Fast Simulation for Soft Robots by Optimization-Based Geometric Computing", *The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, accepted, October 2022.

3. **Guoxin Fang**, Christopher-Denny Matte, Rob B.N. Scharff, Tsz-Ho Kwok, and Charlie C.L. Wang, "Kinematics of Soft Robots by Geometric Computing", *IEEE Transactions on Robotics (T-RO)*, vol.36, no.4, pp.1272-1286, August 2020.

4. **Guoxin Fang**, Rob B.N. Scharff, and Charlie C.L. Wang, "Controlling Multi-segment Soft Robot for Grasping", *IEEE International Conference on Automation Science and Engineering (CASE 2019 short paper)*, August 2019.

5. **Guoxin Fang**, Christopher-Denny Matte, Tsz-Ho Kwok, and Charlie C.L. Wang, "Geometry-based Direct Simulation for Multi-Material Soft Robots", *IEEE International Conference on Robotics and Automation*, May 2018.

6. Yingjun Tian, **Guoxin Fang**, Zhi-Xin Yang, Jo M.P. Geraedts, and Charlie C.L. Wang, "Soft Robotic Mannequin: Design and Algorithm for Deformation Control", *IEEE/ASME Transactions on Mechatronics (T-MECH)*, accepted, 2022.

7. Rob B.N. Scharff, **Guoxin Fang**, Yingjun Tian, Jun Wu, Jo M.P. Geraedts, and Charlie C.L. Wang, "Sensing and Reconstruction of 3D Deformation on Pneumatic Soft Robots", *IEEE/ASME Transactions on Mechatronics (T-MECH)*, vol.26, no.4, pp.1877-1885, August 2021.

Other publications during PhD study

1. **Guoxin Fang**, Tianyu Zhang, Sikai Zhong, Xiangjia Chen, Zichun Zhong, and Charlie C.L. Wang, "Reinforced FDM: Multi-Axis Filament Alignment with Controlled Anisotropic Strength", *ACM Transactions on Graphics*, vol.39, no.6, article no.204, November 2020.

2. Xiangjia Chen, **Guoxin Fang**, Wei-Hsin Liao, and Charlie C.L. Wang, "Field-based Toolpath Generation for 3D Printing Continuous Fibre Reinforced Thermoplastic Composites", *Additive Manufacturing Journal*, Vol. 49, January 2022, 102470.

3. Zeyu Zhang, Chenming Wu, Chengkai Dai, Qingqing Shi, **Guoxin Fang**, Dongfang Xie, Yong-Jin Liu, Charlie C.L. Wang, and Xiu-Jie Wang, "A Multi-axis robot-based bioprinting platform for bioactive artificial blood vessel and cardiac tisssue fabrication", *Bioactive Materials*, accepted, February 2022

4. Tianyu Zhang, Xiangjia Chen, **Guoxin Fang**, Yingjun Tian, and Charlie C.L. Wang, "Singularity-aware motion planning for multi-axis additive manufacturing", *IEEE Robotics*

and Automation Letters (RA-L), vol.6, no.4, pp.6172-6179, October 2021.

5. Chenming Wu, Chengkai Dai, **Guoxin Fang**, Yong-Jin Liu, and Charlie C.L. Wang, ”General support-effective decomposition for multi-directional 3-D printing”, *IEEE Transactions on Automation Science and Engineering (IEEE T-ASE)*, vol.17, no.2, pp.599-610, April 2020.

6. Xiaoting Zhang, **Guoxin Fang**, Melina Skouras, Gwenda Gieseler, Charlie C.L. Wang, and Emily Whiting,”Computational design of fabric formwork”, *ACM Transactions on Graphics*, vol.38, no.4, article no.109 (13 pages), July 2019.

7. Chengkai Dai, Charlie C.L. Wang, Chenming Wu, Sylvain Lefebvre, **Guoxin Fang**, and Yongjin Liu, ”Support-free volume printing by multi-axis motion”, *ACM Transactions on Graphics*, vol.37, no.4, article no.134 (13 pages), July 2018.

8. Chenming Wu, Chengkai Dai, **Guoxin Fang**, Yong-Jin Liu, and Charlie C.L. Wang, ”RoboFDM: a robotic system for support-free fabrication using FDM”, *IEEE International Conference on Robotics and Automation*, May 2017

9. Xiaoting Zhang, Guoxin Fang, Chengkai Dai, Jouke Verlinden, Jun Wu, Emily Whiting, and Charlie C.L. Wang, ”Thermal-comfort design of personalized casts”, *ACM Symposium on User Interface Software and Technology (UIST 2017)*, pp.243-254, Quebec City, Canada