UCLA

UCLA Electronic Theses and Dissertations

Title

Simulation of Deformable Objects for Sim2Real Applications in Robotics

Permalink

https://escholarship.org/uc/item/9259n6s8

Author

Choi, Andrew Sang-Jin

Publication Date

2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Simulation of Deformable Objects for Sim2Real Applications in Robotics

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Andrew Sang-Jin Choi

2023

© Copyright by

Andrew Sang-Jin Choi

2023

ABSTRACT OF THE DISSERTATION

Simulation of Deformable Objects for Sim2Real Applications in Robotics

by

Andrew Sang-Jin Choi
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2023
Professor Demetri Terzopoulos, Co-Chair
Professor Mohammad Khalid Jawed, Co-Chair

From manipulators to self-driving cars, training robots in the real world is often tedious, expensive, and results in hardware wear-and-tear. Due to the aforementioned reasons, the concept of transferring useful skills from "sim2real" has become an extremely attractive avenue for robotics researchers. Training purely (or partially) from simulation boasts numerous immense benefits such as allowing researchers to explore dangerous state spaces, learn faster than real-time, and even employ multiple agents to learn in parallel. Despite this, there currently exists a prominent "sim2real gap", where skills and/or models learned from within simulation transfer poorly to the real world due to environment misalignment. Given the scarcity of physically adequate models, this sim2real gap is especially prominent in contact-rich scenarios as well as problem spaces concerning deformables, whether it be the manipulation of deformable objects or soft robots themselves. In this thesis, I present a culmination of our previous works tackling two key sequential research areas: 1) development of efficient, physically accurate simulators for soft robots and structures and 2) full end-to-end sim2real solutions for robotic deformable material handling for tasks such as stiff sheet folding and deformable linear object deployment. Throughout these works, we showcase the immense benefits of developing solutions with physical insight in the areas of simulation, perception, and robotic manipulation.

The dissertation of Andrew Sang-Jin Choi is approved.

Achuta Kadambi

Bolei Zhou

Jungseock Joo

Mohammad Khalid Jawed, Committee Co-Chair

Demetri Terzopoulos, Committee Co-Chair

University of California, Los Angeles

2023

To my mother and father,

who have supported me unconditionally in all aspects of life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

My four year journey at UCLA pursuing my doctorate degree has been some of the most fulfilling years of my life. Before delving deeper into the fun work I was able to accomplish, I'd like to thank some people who were instrumental in my graduate journey.

To start, I'd be remiss not to thank each and every one of my advisors. Some students are lucky to have a single advisor they get along with; I was blessed with three who have always provided me with their wisdom and attention.

In no particular order...

First, I'd like to thank Professor Demetri Terzopoulos for his continuous mentorship over the course of my Ph.D. program. I originally reached out to him to have an M.S. degree requirement signed off. Though we did not know each other, he expressed an immediate interest in my early work on discrete elastic knots and even offered to have a phone call that very day. Considering his prestigious career and undoubtedly busy schedule, I was taken aback by his kindness and attention. Since then, he has carefully edited my publications and theses, helped me transfer into the CS Ph.D. program, and offered me invaluable advice for graduating in a timely manner. Without him, my journey to a Ph.D. degree would undoubtedly have been more arduous.

I'd like to thank Professor M. Khalid Jawed for essentially being my main-PI the past few years as well as for graciously accepting me into the Structures-Computer Interaction Laboratory. Not only did I feel like I thrived as a researcher under his tutelage, I also met colleagues that I consider life-long friends. His constant generosity in the form of lab dinners and outings created bonds within our lab that I believe many envy. As a PI, he has always been very kind, compassionate, and extremely attentive. No matter what the inquiry, I could expect a response from him within a few hours. Overall, I cannot imagine a better PI to have worked for.

Finally, I'd like to thank Professor Jungseock Joo for his mentorship as my early PI and essentially kick-starting my research journey. He took a chance on an unsure and doubtful 23-year old Andrew and without him, I would not be the researcher I am today. Furthermore, his experiences and insights concerning industry have helped me tremendously through my job search.

In addition to my advisors, I'd also like to thank my dear friend and colleague Dezhong Tong with whom I had the pleasure of completing several research projects. From discussing NBA basketball to carrying out experiments in the wee hours of the night, he was an ever-present colleague and collaborator during my Ph.D. I was extremely fortunate to have a productive, symbiotic relationship where we could both grow as researchers.

I'd like to thank my dear friend Hudson for his continuous support during the past few years. Any hint of imposter syndrome was erased due to his seemingly unending supply of motivational speeches. Words cannot express just how grateful I am for his friendship.

I'd like to thank my partner Elizabeth for her continuous support and consideration. Not only would she always express interest in my work, she would also provide encouragement in times of immense stress and do whatever she could to make my life easier during those times. I could always count on being able to laugh with her and relax in spite of gloomy deadlines.

And, finally, I'd like to thank my parents. They immigrated to this country many years ago without higher educations. The fact that I was able to obtain a Ph.D. degree is a testament to the countless sacrifices they endured in granting me this opportunity. Thank you to my father for his constant encouragement of me to further my education and to my mother for always making sure I was well-fed and taken care of. Without them, none of this would have been possible.

VITA

2014–2018	B.S. in Mechanical Engineering University of California, Davis Davis, CA.
2018–2019	Control Systems Engineer Brock Solutions Los Angeles, CA.
2019–2021	M.S. in Computer Science University of California, Los Angeles Los Angeles, CA.
2021	Robotics Software Intern Vecna Robotics Waltham, MA.
2022–2023	Teaching Assistant Computer Science Department University of California, Los Angeles Los Angeles, CA.
2020–2023	Graduate Student Researcher Mechanical & Aerospace Engineering Department University of California, Los Angeles Los Angeles, CA.
2021–2023	Ph.D. Candidate Computer Science Department University of California, Los Angeles Los Angeles, CA.

PUBLICATIONS

A. Choi, D. Tong, M.K. Jawed, and J. Joo, "Implicit Contact Model for Discrete Elastic Rods in Knot Tying," *Journal of Applied Mechanics*, vol. 88(5), pp. 051010, 2021. (Choi et al., 2021)

A. Vepa, **A. Choi**, N. Nakhaei, W. Lee, N. Stier, A. Vu, G. Jenkins, X. Yang., M. Shergill, M. Desphy, K. Delao, M. Levy, C. Garduno, L. Nelson, W. Liu, F. Hung, and F. Scalzo, "Weakly-Supervised Convolutional Neural Networks for Vessel Segmentation in Cerebral Angiography," *In 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 585-594, 2022. (Vepa et al., 2022)

A. Choi, M.K. Jawed, and J. Joo, "Preemptive Motion Planning for Human-to-Robot Indirect Placement Handovers," *In 2022 International Conference on Robotics and Automation (ICRA)*, pp. 4743-4749, 2022. (Choi et al., 2022)

D. Tong, **A. Choi**, J. Joo, A. Borum, and M.K. Jawed, "Snap Buckling in Overhand Knots," *Journal of Applied Mechanics*, vol. 90(4), pp. 041008, 2023. (Tong et al., 2023a)

D. Tong*, **A. Choi***, J. Joo, and M.K. Jawed, "A Fully Implicit Method for Robust Frictional Contact Handling in Elastic Rods," *Extreme Mechanics Letters*, vol. 58, pp. 101924, 2023. (Tong et al., 2023b)

A. Choi, D. Tong, B. Park, D. Terzopoulos, J. Joo, and M.K. Jawed, "mBEST: Realtime Deformable Linear Object Detection Through Minimal Bending Energy Skeleton Pixel Traversals," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8(8), pp. 4863-4870, 2023. (Choi et al., 2023b)

D. Tong, **A. Choi**, L. Qin, W. Huang, J. Joo, and M.K. Jawed, "Sim2Real Neural Controllers for Physics-Based Robotic Deployment of Deformable Linear Objects," *The International Journal of Robotics Research (IJRR)*, 2023. (Tong et al., 2023c)

A. Choi*, D. Tong*, D. Terzopoulos, J. Joo, and M.K. Jawed, "Learning Neural Force Manifolds for Sim2Real Robotic Symmetrical Paper Folding," *arXiv and in-review for IEEE Transactions on Automation Science and Engineering*, 2023. (Choi et al., 2023c)

A. Choi, R. Jing, A. Sabelhaus, and M.K. Jawed, "DisMech: A Discrete Differential Geometry-based Physical Simulator for Soft Robots and Structures," *arXiv and in-review for IEEE Robotics and Automation Letters (RA-L)*, 2023. (Choi et al., 2023a)

* denotes equal contribution

CHAPTER 1

Introduction

With advances in both hardware and artificial intelligence, robots are increasingly expected to exhibit or surpass human-level capabilities in a variety of tasks. One of the most difficult problem spaces faced by the robotics research community are those related to deformable materials. People possess an incredible innate understanding of soft dynamics; e.g., we can use gravity to manipulate a shirt over our heads without ever having "learned" how. Whether the problem pertains to manipulators dexterously handling deformable objects or the control of soft robots themselves, instilling such intuition into robots remains an important research problem and has the potential to breed numerous applications with considerable economic and humanitarian potential.

As opposed to their rigid-body counterparts, deformable structures possess essentially an infinite number of degrees of freedom and are capable of undergoing highly nonlinear geometric deformations from even minute external forces. The complexity of such structures has resulted in a scarcity of accurate, efficient, and robust simulators for deformable objects, especially those directed towards the robotics community. These limitations have resulted in the dreaded "sim2real gap", an ongoing challenge where the performance of skills and/or models learned through simulation deteriorate when transferred to the real world.

With this in mind, we take a twofold approach to developing robust sim2real solutions. We first focus on making physically accurate continuum mechanics simulations with an emphasis on elastic contact. Then, we directly leverage these simulation tools to tackle nontrivial sim2real deformable object manipulation problems, such as paper sheet folding and deformable linear object (DLO) deployment. Developing physically insightful solutions will be a key topic in this thesis as we explore such contributions in the areas of simulation, perception, and robotic manipulation, as well as their interdependencies.

We next preview the contributions of this thesis in greater detail.

1.1 Contributions

Simulation of Deformable Objects

The first half of this thesis focuses on accurate simulation of deformable objects. One of the key approaches to reducing the sim2real gap lies in developing simulation frameworks that model the real world as accurately as possible. With this in mind, we tackle elastic contact as well as develop robust, easy-to-use solutions for soft simulation. Known for their physical accuracy and computational efficiency, we will be using discrete differential geometry-based (DDG) simulation frameworks to simulate elasticity throughout this thesis.

Elastic Contact and Friction (Choi et al., 2021; Tong et al., 2023b)

One of the key areas in which simulations will stray from the real world arises when dealing with the highly nonlinear process of frictional contact. As such, the sim2real gap is especially prominent for contact-rich scenarios. We formulate and physically validate a fully implicit frictional contact model, the Implicit Contact Model (IMC), capable of enforcing non-penetration and simulating Coulomb friction. IMC is designed to be easily integrated into discrete differential geometry-based frameworks. Using IMC, we simulate several difficult frictional contact scenarios pertaining to slender rods such as knot tying and flagella bundling. Two variations (IMCv1 and IMCv2) are discussed in detail.

General Simulation Framework for Soft Robots and Structures (Choi et al., 2023a)

Though DDG-based simulation has been employed by the robotics community, many previous efforts have primarily used "one-off" simulations. In an attempt to create an easy-to-use generalizable DDG-based simulation framework for the robotics community, we developed DisMech. DisMech offers several functionalities such as arbitrary customization of robot geometry, material, and environment, with elastic contact and self-contact capabilities provided via IMC. Furthermore, continuum actuation is provided by an intuitive natural curvature manipulation scheme. A generalizable framework for real2sim modelling is presented. To the best of our knowledge, DisMech is the first general purpose DDG-based 3D simulation framework.

Sim2Real Deformable Object Manipulation

The second half of this thesis will shift gears and focus on robust sim2real solutions to deformable object manipulation problems, where we directly leverage the simulation tools developed in the first half of the thesis for data generation. In particular, we place emphasis on developing efficient physical solutions robust against changes in material and geometry by leveraging scaling analysis and machine learning. We first formulate a solution for sim2real paper sheet folding. Next, a physics-inspired DLO detection algorithm, mBEST, is formulated. Finally, we directly utilize mBEST to formulate a solution for sim2real DLO deployment.

Sim2Real Paper Sheet Folding (Choi et al., 2023c)

A full sim2real solution for single manipulator paper sheet folding is presented. An accurate model of the external forces of a paper given a grasp position is learned from simulation. This model is then used to formulate a near real-time model-predictive control pipeline using visual-feedback. An extensive robotic case study involving a wide variety of materials from origami paper to cardboard demonstrates the accuracy and generalizability of our method.

Physics-Inspired DLO Detection (Choi et al., 2023b)

Though the state representation of a deformable object within a simulation is known, objects in the real world must be detected. Furthermore, perception algorithms must be real-time to enable high frequency visual-feedback control. Therefore, we formulate Minimal Bending Energy Skeleton Pixel Traversals (mBEST), a robust, real-time DLO instance segmentation algorithm. Using mBEST, we show that we can outperform state-of-the-art algorithms using a remarkably cheap and physically insightful optimization objective inspired by DDG-based simulation frameworks.

Sim2Real DLO Deployment (Tong et al., 2023c)

Finally, we tackle shape control of DLOs when deploying onto a 2D substrate. An optimal control policy is formulated and learned from simulation. We use this neural controller along with mBEST to deploy patterns for a wide variety of DLO and substrate materials with superhuman precision. Using our optimal deployment scheme, we are able to accomplish difficult tasks such as stiff cable placement, knot tying, and mimicking human writing.

1.2 Overview

The remainder of the thesis is organized as follows:

In Chapter 2, we perform an extensive literature review of all relevant areas covered by the thesis. We present the literature review in the same order as the thesis's core material; i.e., we cover frictional contact methods, continuum physics simulators for robotics, robotic folding works, DLO perception algorithms, and robotic DLO manipulation strategies.

Chapters 3, 4, and 5, focus on the steps necessary to construct accurate and efficient continuum physics simulators that make sim2real applications possible.

In Chapter 3, we first present a discrete differential geometry-based simulation framework for slender structures from the computer graphics community, Discrete Elastic Rods (DER), which will be a common tool used throughout the rest of the chapters. Next, we present two versions of our elastic frictional contact model, the Implicit Contact Model (IMC). In Chapter 4, we present the earliest iteration of our contact model, IMCv1, which enforces contact implicitly and friction semi-explicitly.

Finally, we present the most recent version of our contact model, IMCv2, in Chapter 5. IMCv2 supersedes IMCv1 with a fully implicit frictional contact formulation. Therefore, IMCv1 is included simply for archival purposes.

In Chapter 6, we develop and present DisMech, a fully generalizable physical simulator for soft robots and structures. This framework uses a combination of DER + IMC as the mathematical backbone. We show extensive validation with comparisons against the state of the art, several nuanced design simulations, and a generalizable real2sim strategy with a real world validation using a dual soft limb manipulator. We then switch gears from simulation to more robotics-focused projects, where we leverage the tools from earlier chapters directly to develop sim2real solutions.

In Chapter 7, we present a full end-to-end sim2real solution for the nontrivial manipulation task of single manipulator paper folding.

In Chapter 8, we present Minimal Bending Energy Skeleton Pixel Traversals (mBEST), a robust real-time DLO detection algorithm that uses a physics-based optimization objective inspired by DER.

In Chapter 9, we present a full end-to-end sim2real solution for DLO deployment and its application for knot tying and mimicking human handwriting using mBEST.

Finally, in Chapter 10, we provide concluding remarks and discuss future research directions.

CHAPTER 2

Related Work

In this chapter, we review the relevant state of the art concerning the simulation of deformable objects as well as sim2real robotic applications.

2.1 Elastic Contact and Friction

In this thesis, we mainly focus on Coulomb friction, an adequate approximation of dry friction. Note that Coulomb friction degrades when contacted surfaces are conjoined. For engineering problems in which cohesion is important (e.g., cohesive granular media simulation (Mandal et al., 2020; Bertrand et al., 2005; Thakur et al., 2014)), a more elaborated contact theory such as Johnson-Kendall-Roberts (JKR), Derjaguin–Muller–Toporov (DMT), or Maugis models (Johnson et al., 1971; Johnson and Greenwood, 1997; Maugis, 1992) are required. Some prior work uses these elastic cohesive contact models to simulate incipient sliding of cohesive contacts (Borri-Brunetto et al., 2001; Gao et al., 2006). Despite this, Coulomb friction is still the de facto friction model for non-cohesive contact due to its simplicity and high empirical accuracy, where it can be seen implemented in a wide variety of engineering applications, including, contact in elastic structures, most granular media simulations, and more. We therefore build a novel numerical framework based on Coulomb friction. We now delve into the types of Coulomb friction contact handling methods, which can generally be divided into three distinct categories: impulse methods, constraint-based optimization methods, and penalty energy methods.

2.1.1 Impulse Methods

As the name suggests, impulse methods compute contact forces based on the required impulse to keep rod segments from penetrating, with an example being the impulse force model by Spillmann and Teschner (2008). Although computationally efficient and straightforward to implement, unrealistic visual jittering often occurs when simulations use sufficiently large time steps as the generated forces are handled explicitly (Choi et al., 2021). Therefore, impulse methods often must either deal with insufficient physical accuracy or use sufficiently small time steps.

2.1.2 Constraint-Based Methods

Constraint-based methods treat frictional contact as a constrained optimization problem. Jean and Moreau (1987, 1992) implemented convex analysis to propose using unilateral constraints to solve dry friction in granular media. Alart and Curnier (1991) developed the first approach to solving constraint-based contact dynamics using Newton's method to find the root of a non-smooth function. In graphics, Daviet et al. (2011) combined an analytical solver with the complementary condition from Alart and Curnier (1991) to capture Coulomb friction in elastic fibers. In Kaufman et al. (2014), the algorithm from Daviet et al. (2011) was incorporated with a nonlinear elasticity solver to simulate frictional contacts in assemblies of DERs. Based on previous work, Daviet (2020) proposed a general constraint-based framework for simulating contact in thin nodal objects. Overall, constraint-based methods can often produce physically realistic results but are inherently more difficult to implement than impulse and penalty methods (though the growth of open source code has alleviated this considerably). Arguably the largest drawback of constraint-based methods, additional computational costs are incurred at each solving iteration as frictional contact forces must be introduced as additional degrees of freedom in order to satisfy the complementary condition between frictional contact responses and the status of contact regions. This is in contrast to impulse and penalty methods which can obtain contact responses directly based on just configuration-based degrees of

Table 2.1: Overview of simulation frameworks used by the robotics community. Exp and Imp refer to explicit and implicit, respectively.

Framework	Model	Continuum Physics	Physics Complexity	Solver Type
Gazebo (Koenig and Howard, 2004)	Rigid-body physics	✗	low	Exp
MuJoCo (Todorov et al., 2012)	Rigid-body physics	✗	low	Exp or Imp
Bullet (Coumans and Bai, 2021)	Rigid-body physics	✗	low	Exp
SoMo (Graule et al., 2021)	Rigid-body physics	✓−	low	Exp
SOFA (Faure et al., 2012)	FEA	✓	high	Exp or Imp
SoRoSim (Mathew et al., 2022)	Cosserat Rod + GVS	✓	med-high	Exp or Imp
Elastica (Gazzola et al., 2018)	Cosserat Rod	✓	med-high	Exp
DisMech (ours)	Kirchhoff Rod + DDG	✓	med-high	Exp or Imp

freedom.

2.1.3 Penalty Energy Methods

The final contact method type, penalty energy methods, utilize a formulated "contact energy" whose gradient is treated as the contact force. Due to the requirement of a smooth differentiable gradient (and Hessian for implicit formulations), such methods utilize smooth differentiable functions to best approximate the behavior of frictional contact (Li et al., 2020a; Choi et al., 2021; Patil et al., 2020; Tong et al., 2023b). These methods have become popular in recent times as they have been shown capable of generating accurate frictional contact (Choi et al., 2021; Patil et al., 2020; Tong et al., 2023b) while remaining simple to implement (relative to constraint-based methods) and computationally efficient.

2.2 Simulation of Deformable Objects for Robotics

Though soft robotics research has been a popular and growing field (Majidi, 2014), the presence of physical simulators capable of robustly capturing continuum mechanics is scarce. Popular simulation frameworks used for sim2real training of robots include Gazebo (Koenig and Howard, 2004), MuJoCo (Todorov et al., 2012), and Bullet/PyBullet (Coumans and Bai, 2021), but such simulators focus primarily on rigid body dynamics and fail to replicate physically accurate elasticity.

One avenue to remedy this involve frameworks such as SoMo (Graule et al., 2021), which models deformable structures by representing them as rigid links connected by springs. Frameworks such as this simply act as wrappers built on top of preexisting rigid-body physics simulators (Bullet). Although plausible results have been shown for soft grippers and control (Graule et al., 2022), the underlying rigid-body physics is unable to simulate more complicated deformation modes such as twisting and elastic buckling (Tong et al., 2023a). With this, accurate soft robot simulation requires simulation frameworks more dedicated to soft physics.

Another avenue for simulating the continuum mechanics of soft robots involves the classical finite element method (FEM) (Faure et al., 2012; Coevoet et al., 2017). Although such methods are accurate, their high fidelity mesh representation often results in high computational costs, requiring separate model-order reduction (MOR) techniques for online control (Thieffry et al., 2019; Alora et al., 2023). These limitations focus the use of FEM/MOR for offline tasks such as reinforcement learning (RL) (Schegg et al., 2022) and through asynchronous techniques (Largilliere et al., 2015).

Finally, there are simulation frameworks that simulate elastic rods using either the unstretchable and unshearable Kirchhoff rod model (Kirchhoff, 1859) or fully deformable Cosserat rod model (Gazzola et al., 2018). Using the Kirchhoff rod model, the computer graphics community has developed discrete differential geometry-based frameworks such as Discrete Elastic Rods (DER) (Bergou et al., 2008, 2010). Originally meant for realistic animation, DER has shown surprisingly great performance in accurately capturing the nonlinearity of rods and has been rigorously physically validated in cases such as rod deployment (Jawed et al., 2014), knot tying (Choi et al., 2021; Tong et al., 2023b), and buckling (Tong et al., 2023a, 2021). Using a DER-inspired framework, Huang et al. (2020) showed success at simulating and modeling soft robot locomotion, albeit with the omission of twisting as locomotion occurred solely along a 2D plane.

Other frameworks have opted for using the Cosserat rod model as it incorporates the influence of shearing effects. One such example involves SoRoSim (Mathew et al., 2022) which models soft links using a Geometric Variable Strain (GVS) model of Cosserat rods.

This allows for the modeling of soft links with less degrees of freedom (DOFs) compared to traditional lumped mass models (Bergou et al., 2008; Gazzola et al., 2018). Despite this mathematical elegance, the framework can be difficult for users to setup due to the prohibitively unintuitive representation of strains as the DOF.

Arguably the most prominent soft physics simulator based on Cosserat rod theory is Elastica (Gazzola et al., 2018; Zhang et al., 2019) This framework also models soft links as Cosserat rods but uses the more traditional lumped mass model similar to DER. This framework has been extensively physically validated and its potential for reinforcement learning methods has also been demonstrated (Naughton et al., 2021). Despite these results, Elastica uses an explicit integration scheme requiring prohibitively small time step sizes to maintain numerical stability. This is especially true when simulating stiff materials as we will show later in Section 6.3. A summary of popular robotics simulation frameworks can be seen in Table 2.1.

2.3 Robotic Folding of Sheets

Next, we will dive into the vast robotic folding literature. The majority of prior work addressing the folding problem can be roughly divided into four categories: mechanical design-based solutions, vision-based solutions, learning-based solutions, and model-based solutions.

2.3.1 Mechanical Design-Based Approaches

Mechanical design-based approaches typically involve tackling the folding problem using highly specialized manipulators or end effectors. Early approaches involved specialized punches and dies for sheet metal bending (Kim et al., 1998). More recently, highly specialized manipulators for robotic origami folding have also been developed (Balkcom and Mason, 2008). Such methods can reliably produce repeatable folding but are often limited to a highly specific fold, geometry, and/or material.

2.3.2 Vision-Based Approaches

Vision-based approaches involve folding deformable materials by generating folding motions purely from visual input. These techniques are commonly applied to tasks such as folding clothes, where the primary focus is on detecting the garments' shape or key grasp points. Techniques for key feature extraction involve random decision trees (Doumanoglou et al., 2016), RGB-D sensing data analysis (Maitin-Shepard et al., 2010; Twardon and Ritter, 2015), and fitting strategies where the detected state of deformed clothes are compared against precomputed shapes (Kita et al., 2011; Doumanoglou et al., 2014). Given the soft nature of clothes, subsequent manipulations are often formulated intuitively. While some prior works employ models to predict optimal manipulation sequences, these models are typically oversimplified and lack physical details (Miller et al., 2012). Such approaches can be effective and rather simple to implement, but do not transfer well to paper folding as paper possesses a much higher stiffness when compared to fabric and will attempt to restore its natural, undeformed state if not properly handled.

2.3.3 Learning-Based Approaches

Learning-based approaches involve the robot learning how to fold through training data. The most popular has been to learn control policies from human demonstrations, also known as learning from demonstrations (LfD). Prior research has demonstrated flattening and folding towels (Lee et al., 2015b,a). Teleop demonstrations are a popular avenue for training policies and have been used to learn how to manipulate deformable linear objects (DLOs) (Rambow et al., 2012) as well as folding fabric (Yang et al., 2017). To eliminate the need for expensive human-labeled data, researchers have also focused on tackling the sim2real problem for robotic folding, with reinforcement learning being used to train robots to fold fabrics and clothes completely from simulation (Petrík and Kyrki, 2019; Matas et al., 2018; Lin et al., 2021). More recently, Zheng et al. (2022) used reinforcement learning to train a robot to flip pages in a binder through tactile feedback. Pure learning-based methods have shown promising performance, but only for specific tasks whose state distribution matches the training data. Such methods tend to generalize quite poorly; e.g., when the material or geometric properties change drastically.

2.3.4 Model-Based Approaches

Model-based approaches, where the model can either be known or learned, often use model predictive control to manipulate the deformable object. Learned models involve learning the natural dynamics of deformable objects through random perturbations (Yan et al., 2020b). These models are generally fast, but they can be inaccurate when experiencing new states. Theoretical models are often formulated to be as physically accurate as possible, which enables the direct application of their predictive power in the real world. Examples of this have been published for both strip folding (Petrík et al., 2016, 2020) and garment folding (Li et al., 2015). Physical models are often constructed using energy-based formulations (Sanchez et al., 2018; Yin et al., 2021; Zhu et al., 2022), where various elastic energies are computed based on the topological properties of the simulated objects to solve their deformed shape under manipulation. For example, Wakamatsu and Hirai (2004) modeled deformable linear objects (rods) with flexure (bending), torsion, and extension (stretching), while Jia et al. (2014) introduced manipulation as a potential energy to compute the deformations of deformable planar objects. However, theoretical models are usually quite expensive to run and must often face a trade-off between accuracy and efficiency.

2.3.5 Limitations of Prior Work for Paper Folding

Despite the large quantity of prior research focusing on 2D deformable object manipulation, the majority of these efforts have limited their scope to soft materials such as towels and cloth. Such materials are highly compliant and often do not exhibit complicated nonlinear deformations, thus allowing for solutions lacking physical insight. We instead tackle the scenario of folding paper of various stiffnesses with a single manipulator in Chapter 7. Because of its relatively high bending stiffness and slippery surface, paper is significantly more difficult to manipulate since large deformations will cause sliding of the paper on the substrate. Such an example can be observed in Figure 7.1, where intuitive folding trajectories that may work on towels and cloth fail for paper due to undesired sliding.

However, a few researchers have attempted to solve the paper folding problem. For example, Elbrechter et al. (2012) demonstrated paper folding using visual tracking and real-time physics-based modeling, with impressive results, but they required expensive end effectors (two Shadow Dexterous Hands), one end effector to hold the paper down while folding at all times, and the paper to have AR tags for visual tracking. Similarly, Namiki and Yokosawa (2015) also achieved paper folding through dynamic motion primitives and used physics-based simulations to estimate the deformation of the paper sheet, also requiring highly specialized manipulators and an end effector to hold the paper down while folding. By contrast, our method can fold papers reliably without any need for holding down the paper during the folding operation and requires only a simple 3D printed gripper.

Other researchers have also attempted to fold with a single manipulator while minimizing sliding (Petrík et al., 2016; Petrík and Kyrki, 2019; Petrík et al., 2020), but their methods focused on fabrics whose ends were taped down to the substrate. Though these methods have achieved favorable folding accuracy using a physical model for garments and fabric, we have observed in our experiments that their generated trajectories perform poorly when applied to paper folding. We believe that this is due to their local optimization strategy of solving the subsequent grasp pose using only the current grasp. In contrast, we generate our folding trajectories through global optimization, thus showcasing the importance of considering both current and future deformation states during the paper manipulation process.

2.4 DLO Perception

Although research into manipulation skills for DLOs has been prevalent, the perception algorithms used in support of these efforts remain underdeveloped. For example, in the work of Tong et al. (2021), attached markers are required to determine the configuration of the manipulated DLO. Zhu et al. (2018) carefully adjusted the workspace to increase the contrast between the manipulated DLOs (cables) and their background. Although these prior efforts successfully completed their target manipulation tasks, the simplistic perception algorithms restrict real world applicability.

Consequently, DLO detection algorithms featuring various methodologies have been proposed. Keipour et al. (2022) evaluated both curvatures and distances to fit a continuous DLO. Using data-driven methods, Yan et al. (2020a) trained a neural network to reconstruct the topology of a DLO based on a coarse-to-fine nodal representation. Though these methods achieve good results for some datasets, they work under the strict assumption that only one DLO exists within the scene, which dramatically restricts their applicability.

2.4.1 Multi-DLO Instance Segmentation

One of the first perception algorithms capable of detecting multiple DLOs, *Ariadne* (De Gregorio et al., 2018), segments images into superpixels and traverses the superpixels belonging to DLOs in order to produce paths. The ambiguity of intersections is handled using a multi-faceted cost function that takes into consideration color, distance, and curvature. Despite its satisfactory performance, this early approach suffers from a large number of hyperparameters, an overreliance on DLOs being a uniform color, and the tedious requirement that the user manually select the ends of DLOs. Furthermore, the processing speed of *Ariadne* is on the order of seconds, precluding real-time operation.

2.4.2 Deep Learning Models for DLO Segmentation

In recent years, data-driven computer vision methods have attracted increasing attention and researchers have shown that image segmentation problems can be tackled efficiently and accurately using Deep Convolutional Neural Networks (DCNNs), particularly instance segmentation (Bolya et al., 2019, 2020; Chen et al., 2020; Tian et al., 2020). Furthermore, techniques have been introduced to help synthetically generate large quantities of photorealistic data in order to adequately train such models (Denninger et al., 2019; Qiu and Yuille, 2016; Caporali et al., 2023b). Using DCNNs, Zanella et al. (2021) created segmentations of DLOs such as wires; however, the segmentations did not distinguish between each DLO.

2.4.3 Achieving Realtime Performance

Improving upon *Ariadne*, *Ariadne+* (Caporali et al., 2022b) also utilizes a DCNN model to extract an initial binary mask of the DLOs. This allows the algorithm to then apply superpixel segmentation purely on the binary mask itself, significantly reducing the computation time. Paths are then generated in a similar fashion to the original *Ariadne* algorithm by traversing superpixels while intersections are handled using a neural network to predict the most probable paths. Despite these improvements, *Ariadne+* is sub-realtime; i.e., less than 3 FPS.

Another algorithm, *FASTDLO* (Caporali et al., 2022a) improves upon the speed of *Ariadne+* by forgoing superpixel segmentation altogether. Instead, it uses a skeleton pixel representation of the DLO binary mask for path traversals. Intersections are then also handled by a neural network. By replacing superpixel segmentation with skeletonization, *FASTDLO* is able to achieve a real-time performance of 20 FPS for images of size 640×360 pixels.

More recently, *RT-DLO* (Caporali et al., 2023a) detects DLOs by representing them as sparse graphs where nodes are sampled from DLO centerlines and edges are selected based on topological reasoning. This results in increased runtime efficiency and accuracy compared to *Ariadne+* and *FASTDLO*, but requires sampling along the centerlines of the DLO to remain computationally competitive, often resulting in noisy segmentations. Furthermore, several hyperparameters must be set.

Table 2.2: Overview of state-of-the-art 2D DLO perception algorithms.

Algorithm	Intersection Rule	DLO Representation	Real-time
Ariadne (De Gregorio et al., 2018)	color, distance, curvature	superpixels	✗
Ariadne+ (Caporali et al., 2022b)	DNN prediction	superpixels	✓$^{-}$
FASTDLO (Caporali et al., 2022a)	DNN prediction	skeleton pixels	✓
RT-DLO (Caporali et al., 2023a)	cosine similarity	sparse graph	✓
mBEST (ours)	curvature	skeleton pixels	✓

2.4.4 Limitations of Prior Work for DLO Detection

Ariadne+, *FASTDLO*, and *RT-DLO* are considered state-of-the-art DLO perception algorithms, but they have been evaluated only on scenes containing DLOs with relatively smooth curvatures and minimal self-loops. Our experiments in Chapter 8 will show that these algorithms struggle to resolve nontrivial configurations (e.g., DLOs with highly variable curvatures resulting in many crossings and tangles and/or nearly-parallel intersections). We argue that a physically principled approach can outperform both sparse graphs and black box neural network approaches when dealing with intersections. Our *mBEST* algorithm robustly solves complex scenes using the simple notion that the most probable path is the one that minimizes cumulative bending energy. Not only does *mBEST* outperform in accuracy, it also achieves real-time performance with a 50% improvement over the next best algorithm and it has no hyperparameters to set. Table 2.2 summarizes the key algorithmic differences between *mBEST* and competing algorithms.

2.5 Robotic Deployment of DLOs

Constructing a mapping relationship from observations of a manipulated DLO to the robot's action space is the primary basis of controlling DLOs. To uncover this mapping relationship, prior works usually implemented models to predict or perception systems to observe the deformations of DLOs under various manipulations. Manipulation schemes are then generated based on the predicted or sensed data. Therefore, model-based and perception-based methods can be considered two of the main categories for tackling manipulation problems of deformable objects. Given the impressive performance of machine learning algorithms for processing and generalizing data from models and perceptions, learning-based approaches have become another mainstream solution. In fact, many prior works take advantage of a combination of these three methods to develop hybrid schemes for different manipulation tasks. Here, we carry out a systematic review of prior scholarly contributions that have utilized techniques based on the three delineated categories to manipulate DLOs and other deformable objects.

2.5.1 Vision-Based Approaches

Perception-based approaches involve utilizing sensors such as tactile sensors (She et al., 2021) and cameras (Tang et al., 2018; Yan et al., 2020a; Lee et al., 2014; Maitin-Shepard et al., 2010) to generate motions based on detected deformations. While sensors can capture the deformations as the manipulation proceeds, perception-based methods are usually not robust against the material and geometrical differences of the manipulated objects. In Tang et al. (2018), a learning-based perception framework is presented based on the Coherent Point Drift algorithm, which is able to register states of manipulated DLOs with captured images. Yan et al. (2020a) developed state estimation algorithms for DLOs based on images so that a robot can perform pick-and-place manipulation on the detected configuration. However, those perception systems based on cameras fail to extract accurate results when occlusions happen. To overcome this shortcoming, tactile sensors have become prevalent in the robotics community. For example, She et al. (2021) implements GelSight, a force feedback tactile sensor, to perform robotic cable management. Since sensing data by itself cannot predict future deformations of the manipulated objects, pure perception-based methods are typically insufficient for complex deformable material manipulation tasks.

2.5.2 Model-Based Approaches

Model-based methods usually construct a physically accurate model to predict the behavior of manipulated DLOs. Multiple methods exist for modeling DLOs (Yin et al., 2021; Sanchez et al., 2018). A simple and widely-used model, mass-spring systems, are often used to model deformable objects including ropes (Schulman et al., 2013; Kita et al., 2011; Macklin et al., 2014), fabrics (Macklin et al., 2016; Guler et al., 2015), etc. However, due to the simplification of mass-spring systems, such models usually suffer from inaccuracies when undergoing large deformations and lack of physical interpretability. Position-based dynamics is another type of modeling method that usually represents DLOs as chains of rigid bodies (Servin and Lacoursiere, 2008; Terzopoulos and Qin, 1994; Müller et al., 2007) and introduces constraints between the positions of those rigid bodies to simulate deformations. Though this method is straightforward and fast, physical interpretability is also lacking.

Finite element methods (FEM) are also popular for modeling deformable objects (Haouchine et al., 2018; Kaufmann et al., 2009; Buckham et al., 2004). However, FEM usually requires considerable computation resources and is hardly suitable for online predictions. More recently, fast simulation tools from the computer graphics community have attracted researchers' attention. For example, Discrete Elastic Rods (DER) (Bergou et al., 2008, 2010) has arisen as a robust and efficient algorithm for simulating flexible rods. Lv et al. (2022) used DER as a predictive modeling tool and achieved promising performance in DLO manipulation tasks. Though various ways to model deformable objects exists, each has their respective strengths and weaknesses and often possesses a trade-off between computational efficiency and accuracy.

2.5.3 Learning-Based Approaches

Finally, learning-based approaches have become prevalent as they are capable of not only predicting the shape of the deformable object but also higher-level information such as forces (Choi et al., 2023c). Most prior works use human demonstrations or robot explorations to train controlling policies for different tasks. Nair et al. (2017), Sundaresan et al. (2020), and Lee et al. (2021) fed human-made demonstrations to robots for learning control policies for shape control and knot-tying. Due to the tedium of constructing manual demonstrations, some researchers take advantage of the robots' automation to learn a policy purely from robotic exploration (Yu et al., 2022a; Wang et al., 2019). To acquire training data more efficiently, researchers have also looked into training policies purely from simulation (Matas et al., 2018). Although learning-based methods have shown promising performance for manipulating deformable objects, the trained policies are often only valid for specific tasks whose state distribution matches that of the training set. In other words, learning-based approaches often fail when parameters such as the material and geometrical properties of the manipulated object change.

More relevant to the deployment task itself, Takizawa et al. (2015) implemented the intuitive control method shown in Figure 9.2(a) for controlling the shape of a rope to make a clove hitch knot. They achieve a success rate of 60% but require empirical hardcoded adjustments to their controlling scheme, indicating the intuitive approach's unsuitability for extreme precision deployment. Additionally, Lv et al. (2022) uses a precise physical numerical model to predict the DLO's configuration during deployment. However, they use a trial-and-error method to exhaustively solve the optimal deployment path, which is computationally expensive and slow.

2.5.4 Limitations of Prior Work for DLO Deployment

Although the three discussed types of methods are suitable to be combined when solving deformable manipulation problems given the complementariness of their pros and cons, how to develop a combined approach to take advantage of different types of approaches is still an open problem in the robotic community. We find that combining physically accurate simulations and machine learning can endow the learned model with excellent accuracy from the simulations and real-time performance because of the inference speed of the neural network. In addition, scaled physics analysis, which is a vital tool from the mathematical physics community, is valuable for augmenting the model with high generality. In this thesis, we show how physical analysis can extract the true contributing factors of the deployment problem and how a learning-based approach can generalize the information from physics to offer real-time computation speed for the manipulation task.

CHAPTER 3

Discrete Elastic Rods (DER)

Note: As almost every part of this dissertation involves or is in part inspired by DER, it would be remiss not to give a brief overview of the framework. Before delving further, the reader can peruse this section to understand DER, which will provide a foundation for understanding later chapters.

3.1 Introduction

Subsequent to the the pioneering work on physics-based modeling and simulation of deformable curves, surfaces, and solids in computer graphics (Terzopoulos et al., 1987; Terzopoulos and Fleischer, 1988b,a), the community has shown impressive results using discrete differential geometry-based (DDG-based) simulation frameworks. In particular, DDG-based simulations have shown surprisingly successful performance in capturing the nonlinear mechanical behaviors of slender structures, e.g. rods (Bergou et al., 2008, 2010; Audoly and Pomeau, 2010; Jawed et al., 2018b), viscous threads (Audoly et al., 2013; Bergou et al., 2010), ribbons (Shen et al., 2015), and plates/shells (Baraff and Witkin, 1998; Grinspun et al., 2003; Batty et al., 2012). Discrete Elastic Rods (DER) algorithm (Bergou et al., 2010; Jawed et al., 2018b) – originally developed in the computer graphics community to simulate hair, fur, and other filamentary structures in the movies – has been borrowed by the engineering community to solve a variety of problems involving deployment of rods (Jawed et al., 2014; Jawed and Reis, 2014; Jawed et al., 2015b; Tong et al., 2023c), propulsion of bacterial flagella (Jawed et al., 2015c; Jawed and Reis, 2016, 2017), elastic gridshells (Panetta et al., 2019; Baek et al., 2018; Baek and Reis, 2019),

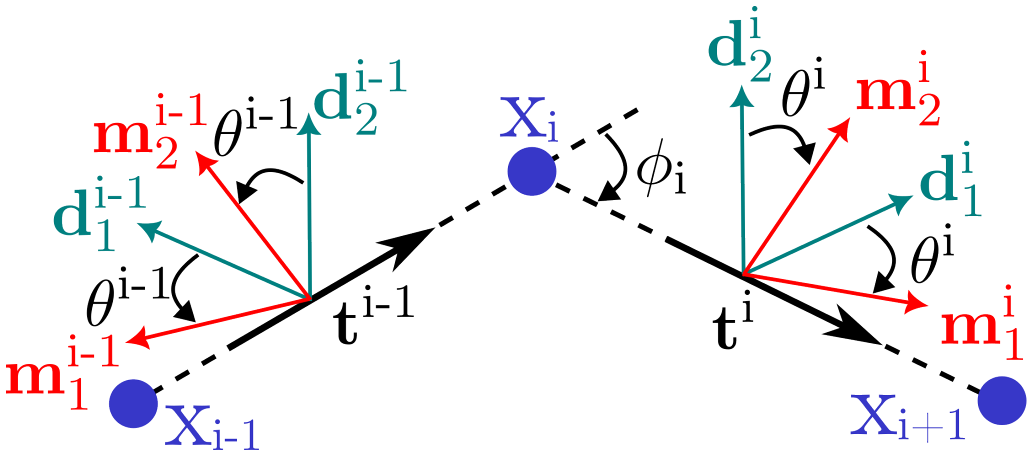


Figure 3.1: Discrete schematic of an elastic rod showcasing nodes $\mathbf{x}_i$, reference frames $\{\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{t}^i\}$, and material frames $\{\mathbf{m}_1^i, \mathbf{m}_2^i, \mathbf{t}^i\}$.

self assembly of carbon nanotubes (Jawed et al., 2018a), helix bifurcation (Tong et al., 2021), overhand knot tightening (Choi et al., 2021; Tong et al., 2023b), overhand knot buckling (Tong et al., 2023a), flagella buckling (Jawed et al., 2015c; Tong et al., 2023b), and dynamic cantilever beams (Choi et al., 2023a). Given the vast literature validating its physical accuracy, DER will be used extensively to produce realistic simulations as well as sim2real solutions in this thesis. We now briefly go over its formulation.

3.2 Reduced-Order Model and Degrees of Freedom

Discrete Elastic Rods is a reduced-order model requiring only the discretized centerline of the elastic rod. It expresses the centerline of an elastic rod with N discrete nodes: $\mathbf{x}_0, \mathbf{x}_1, ...\mathbf{x}_{N-2}, \mathbf{x}_{N-1}$. This results in a total of $N-1$ edges where $\mathbf{e}^i = \mathbf{x}_{i+1} - \mathbf{x}_i$. Note that for DER, we use subscripts to denote indices for quantities associated with nodes and superscripts for indices for quantities associated with edges. Following this, each edge $\mathbf{e}^i$ is described using two orthogonal frames: a reference frame $\{\mathbf{t}^i, \mathbf{d}_1^i, \mathbf{d}_2^i\}$ and a material frame $\{\mathbf{t}^i, \mathbf{m}_1^i, \mathbf{m}_2^i\}$ as shown in Figure 3.1. The reference frame is arbitrarily predefined at initial time $t = 0$s and is updated between time steps via time parallel transport (Bergou et al., 2010). The material frame shares the same director $\mathbf{t}^i = \mathbf{e}^i / \|\mathbf{e}^i\|$ as the reference frame and is obtainable through a twist angle θ^i with respect to the reference frame. A total of

N nodes, each represented by a Cartesian coordinate $\mathbf{x}_i \in \mathbb{R}^3$, and $N-1$ twist angles constitute a total of $4N-1$ degrees of freedom: $\mathbf{q} = [\mathbf{x}_0, \theta^0, \mathbf{x}_1, ..., \mathbf{x}_{N-2}, \theta^{N-2}, \mathbf{x}_{N-1}]^T$; here, ^{T} denotes the transposition operator.

3.3 Elastic Energies

We now define the constitutive laws of the elastic energies whose gradients and Hessians will be used to generate the inner elastic forces and Jacobians, respectively. With DER based on the Kirchhoff rod model (Kirchhoff, 1859), the three main modes of deformation are stretching, bending, and twisting.

3.3.1 Stretching Energy

The stretching strain of an edge $\mathbf{e}^i$ is expressed simply by the ratio of elongation / compression with respect to its undeformed state. Moving forwards, all quantities with a $\bar{}$ represent those respective quantities in their natural undeformed state. With this, stretching strain can be defined as

$$\epsilon^i = \frac{\|\mathbf{e}^i\|}{\|\bar{\mathbf{e}}^i\|} - 1. \tag{3.1}$$

Stretching energy is then be computed as

$$E_s = \frac{1}{2} EA \sum_{i=0}^{N-2} \left(\epsilon^i\right)^2 \|\bar{\mathbf{e}}^i\|, \tag{3.2}$$

where E is Young's modulus and A is the cross-sectional area.

3.3.2 Bending Energy

Bending deformations occur between two adjacent edges and therefore, their respective strain involves computing the curvature at each interior node. The bending strain can be evaluated through a curvature binormal which captures the misalignment between two edges:

$$(\kappa \mathbf{b})_i = \frac{2\mathbf{t}^{i-1} \times \mathbf{t}^i}{1 + \mathbf{t}^{i-1} \cdot \mathbf{t}^i}. \tag{3.3}$$

Using this curvature binormal, the integrated curvature vector at node $\mathbf{x}_i$ can be evaluated using the the material frames of each adjacent edge as

$$\begin{aligned} \kappa_{1,i} &= \frac{1}{2}(\kappa \mathbf{b})_i \cdot \left(\mathbf{m}_2^{i-1} + \mathbf{m}_2^i\right), \\ \kappa_{2,i} &= -\frac{1}{2}(\kappa \mathbf{b})_i \cdot \left(\mathbf{m}_1^{i-1} + \mathbf{m}_1^i\right), \\ \boldsymbol{\kappa}_i &= [\kappa_{1,i}, \kappa_{2,i}] \end{aligned} \tag{3.4}$$

Note that $\boldsymbol{\kappa}_i = 2\tan(\boldsymbol{\phi}_i/2)$ where $\boldsymbol{\phi}_i = [\phi_{1,i}, \phi_{2,i}]$ is the turning angle vector (Figure 3.1). With this, we compute the bending energy of the rod as

$$E_b = \frac{1}{2} EI \sum_{i=1}^{N-2} (\boldsymbol{\kappa}_i - \bar{\boldsymbol{\kappa}}_i)^2 \frac{1}{V_i}, \tag{3.5}$$

where $I = \pi h^4/4$ is the moment of inertia; h is the rod radius, and $V_i = (\|\mathbf{e}^i\| + \|\mathbf{e}^{i-1}\|)/2$ is the Voronoi length. Later, we will show how to manipulate the natural curvature $\bar{\boldsymbol{\kappa}}$ to intuitively actuate soft robot limbs (Section 6.4.2).

3.3.3 Twisting Energy

Like bending, twisting is also a deformation mode that occurs between two adjacent edges. Therefore the twisting strain at an interior node $\mathbf{q}_i$ is computed as

$$\tau_i = \theta^i - \theta^{i-1} + \beta^i, \tag{3.6}$$

where β^i is the signed angular difference between the two consecutive references frames of the i and $i-1$-th edges.

Finally, twisting energy is then defined as

$$E_t = \frac{1}{2} GJ \sum_{i=1}^{N-2} (\tau_i - \bar{\tau}_i)^2 \frac{1}{V_i}, \tag{3.7}$$

where G is the shear modulus and $J = \pi h^4/2$ is the polar second moment of area.

3.3.4 Elastic Forces

With each of the elastic energies defined, we can now compute the inner elastic forces (for nodal positions $\mathbf{x}_i$) and elastic moments (for twist angles θ^i) as the negative gradient of elastic energy:

$$\mathbf{F}^{\text{int}} = -\frac{\partial}{\partial \mathbf{q}} E^{\text{elastic}}, \tag{3.8}$$

where $E^{\text{elastic}} = E_s + E_b + E_t$.

3.4 Equations of Motion

Following this, we can write the system of equations of motions as the sum of inertial terms, internal forces, and external forces (e.g. contact, friction, gravity). This results in the equation

$$\mathbf{F} \equiv \mathbb{M}\ddot{\mathbf{q}} - \mathbf{F}^{\text{int}} - \mathbf{F}^{\text{ext}} = 0, \tag{3.9}$$

where $\mathbb{M}$ is the diagonal mass matrix, $\ddot{\mathbf{q}}$ is the second derivative of the DOFs with respect to time, $\mathbf{F}^{\text{ext}}$ is the external force vector, and $\mathbf{F}$ is the total force. The four external forces used in this thesis are (1) contact forces $\mathbf{F}^{\text{c}}$, (2) friction forces $\mathbf{F}^{\text{fr}}$, (3) viscous forces $\mathbf{F}^{\text{v}}$, and (4) hydrodynamic forces $\mathbf{F}^{\text{hy}}$; and therefore we have $\mathbf{F}^{\text{ext}} = \mathbf{F}^{\text{c}} + \mathbf{F}^{\text{fr}} + \mathbf{F}^{\text{v}} + \mathbf{F}^{\text{hy}}$. The formulation of contact and friction will be a key topic for Chapters 4 and 5 while viscous and hydrodynamic forces are formulated in Sections 4.6.6 and the supplementary material of Tong et al. (2023b), respectively.

3.5 Time Stepping Scheme

In DER, backward Euler is used to solve the $4N-1$ equations of motion in order to update the DOF vector $\mathbf{q}$. For the march from time t_i to $t_{i+1} = t_i + \Delta t$ where Δt is the time step size, (3.9) can be rewritten as

$$\frac{\mathbb{M}}{\Delta t}\left[\frac{\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)}{\Delta t} - \dot{\mathbf{q}}(t_i)\right] - \mathbf{F}^{\text{int}}(t_{i+1}) - \mathbf{F}^{\text{ext}}(t_{i+1}) = 0 \tag{3.10}$$

where $\mathbf{q}(t_i)$ are the DOFs at time t_i, and $\dot{\mathbf{q}}(t_i)$ are the velocities at time t_i.

The old DOFs and velocities $(\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i))$ are known and the task at hand is to compute the new DOFs and velocities $(\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1}))$. As the Jacobian for (3.10) can be computed, Newton-Raphson method is then used to solve for $\mathbf{q}(t_{i+1})$ iteratively. Each element of the Jacobian matrix $\mathbf{J}$ at row i and column j is expressed as follows:

$$\mathbf{J}_{ij} = \frac{m_i}{\Delta t^2}\delta_{ij} + \frac{\partial^2 E^{\text{elastic}}}{\partial q_i \partial q_j} - \left(\mathbf{J}^{\text{c}}\right)_{ij} - \left(\mathbf{J}^{\text{fr}}\right)_{ij} - \left(\mathbf{J}^{\text{v}}\right)_{ij}, \tag{3.11}$$

where $\mathbf{J}^{\text{c}}, \mathbf{J}^{\text{fr}}$, and $\mathbf{J}^{\text{v}}$ are square matrices representing the gradient of the three external forces – contact, friction, and viscous – with respect to the DOFs. Once $\mathbf{q}(t_{i+1})$ is known, the new velocity is simply $\dot{\mathbf{q}}(t_{i+1}) = \left(\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)\right)/\Delta t$.

Normally, if the gradient of an external force, $\partial F_i^{\text{ext}}/\partial q_j$, cannot be analytically evaluated (e.g., for hydrodynamic forces F^{hy}), this term is omitted during Newton iterations and that external force is considered "explicitly" (Euler forward). This generally requires a smaller time step size Δt, leading to larger computation time. Later, we will show that for IMC, the contact and friction Jacobians are analytically obtainable and allow us to take aggressive time steps. Implicit treatment of contact (IMCv1 in Chapter 4) and friction (IMCv2 in Chapter 5) is a key contribution of this thesis.

CHAPTER 4

IMCv1: A Semi-Implicit Formulation for Contact and Friction

Note: In this chapter presents my first iteration of an implicit penalty-based contact method. This method (IMCv1) uses a semi-implicit friction implementation as well as a smoothed approximation of the Lumelsky's min-distance algorithm (Lumelsky, 1985). Our later formulation (IMCv2 in Chapter 5) outperforms IMCv1 in most ways. In particular, IMCv2 uses a fully-implicit friction formulation. Additionally, the smoothed Lumelsky algorithm was found to be unnecessary as there exists an analytical piecewise smooth formulation for min-distance (Li et al., 2018). Therefore, this chapter is included only for archival purposes of my research endeavors. For interested readers, I have provided several footnotes throughout this chapter detailing "lessons" learned as well as areas of direct improvement by IMCv2. Other readers may wish to continue directly to Chapter 5 after examining Figure 4.2.

4.1 Introduction

In prior works, DER has been used to handle contact during knot tying using a contact method proposed by Spillmann and Teschner (2008). This model resolves contact by computing the contact forces that will exactly lead to the desired, collision-free state. Although computationally efficient, unrealistic visual jittering during knot tying occurs for sufficiently large time steps due to its explicit nature.

Recall that DER discretizes the elastic rod into a number of "nodes". Two consecutive nodes are connected by an "edge". To deal with contact, we define a contact energy

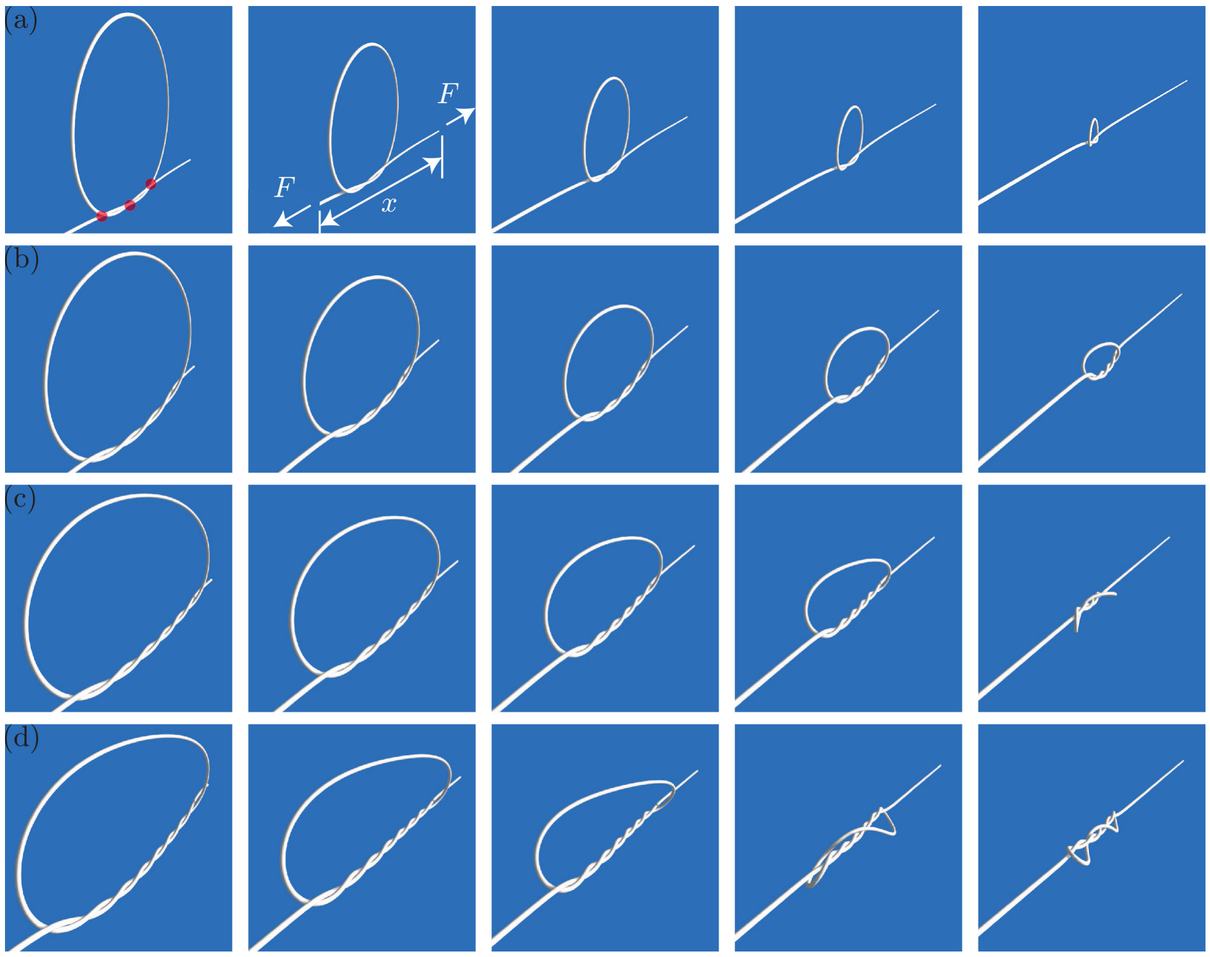


Figure 4.1: Knot tying process using DER and IMCv1 for overhand knots with unknotting numbers (a) $n = 1$, (b) $n = 2$, (c) $n = 3$, and (d) $n = 4$. Red dots in (a) represent the crossing points of the braid. Unknotting number n is equal to $\frac{1}{2}\times$ (number of crossing points - 1). The end-to-end distance of a knot is $e = L - x$, where L is the total length of the rod and x is the distance between the two ends of the knot. The knot starts off in the configuration denoted by the left most column and is gradually pulled tight from both ends leading to the configuration shown in the rightmost column. Physical parameters are detailed in Section 4.7.1.

which will be used to derive the normal contact forces (responsible for enforcing non-penetration) and Coulombic frictional forces. Instead of formulating this contact energy as a function of the distance between nodes, we formulate it as a function of the minimum distance between two edges, which results in more visually and physically realistic results. Figure 4.1 presents snapshots from our simulations of the knot tying process, where the two free ends of the "tails" of the knots are pulled .

Throughout this section, we consider open overhand knots (Jawed et al., 2015a); such

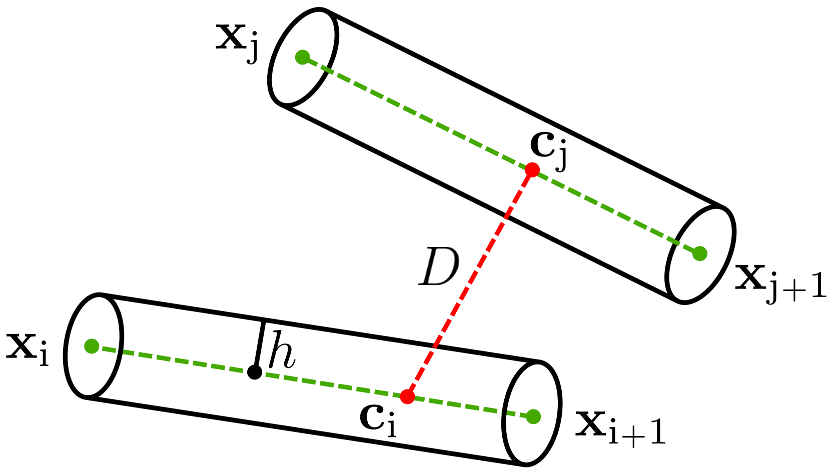


Figure 4.2: Illustration of two edges approaching contact. The green dots showcase the nodes of the edges while the green dashed lines denote the centerlines of the edges. The red dashed line denotes the vector $\vec{D}$ whose norm is the minimum distance D between the edges. $\vec{D}$ is connected to edges i and j by $\mathbf{c}_i = \mathbf{x}_i + \beta_i(\mathbf{x}_{i+1} - \mathbf{x}_i)$ and $\mathbf{c}_j = \mathbf{x}_j + \beta_j(\mathbf{x}_{j+1} - \mathbf{x}_j)$ where $\beta_i, \beta_j \in [0, 1]$.

knots can be described by the unknotting number n, related to the number of turns in the "braid" of the knot. Figures 4.1(a-d) show knots with $n = 1, \ldots, 4$. Interestingly, when the knots with $n = 3$ and $n = 4$ are sufficiently tight, they undergo snap-through buckling and the "loop" of the knot suddenly transitions from a near-circular shape to a distorted configuration. The simulation can reliably capture this behavior.

4.2 Contact Energy

Referring to Figure 4.2, denote $\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1} \in \mathbb{R}^3$ as the Cartesian nodal coordinates of the i-th and j-th edges in a rod configuration. Next, denote an edge "combination" as the following vector concatenation: $\mathbf{x}_{ij} := [\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1}]^T$. We denote the set of all valid edge combinations as $\mathcal{X}$; two consecutive edges are always in contact and those combinations are not included in this *valid* combination $\mathcal{X}$.

From here, an arbitrary edge combination is denoted as simply $\mathbf{x}$ and all edge combinations are assumed to be valid: $\mathbf{x} \in \mathcal{X}$. The contact energy E_c is then expressed as a differentiable analytical expression which takes the four nodes of the two contacting edges as inputs, $E_c(\mathbf{x}) : \mathbb{R}^{12} \rightarrow \mathbb{R}^1$. Under this formulation, we can see that the proposed contact energy is only dependent on the nodal coordinates of the discretized rod and not on the twist angles of the edges, θ, as the contact forces are computed based on the minimum distance, D, between two contacting edges.

Following this, in order to calculate D, we utilize an efficient and accurate algorithm for computing the minimum distance between two finite line segments in N dimensions proposed by Lumelsky (1985). Originally a piecewise function, the algorithm is then modified to become a twice differentiable smooth approximation. Using this completed expression, we can then obtain the negative gradient of the energy $-\nabla_{\mathbf{x}} E_c$ as well as the negative Hessian $-\nabla^2_{\mathbf{x}} E_c$ which are then used to evaluate $\mathbf{F}^c$ and $\mathbf{J}^c$ in (3.10) and (3.11). The gradient of E_c produces contact forces that act in the direction of the contact normal and whose magnitude varies with D which results in physically realistic forces when dealing with rod-rod contact. Finally, as this method is essentially a penalty method, a stiffness parameter k is then used to scale $\mathbf{F}^c$ and $\mathbf{J}^c$ appropriately.

By producing the contact forces in this way, dynamic friction can be calculated according to Coulomb's friction law. In the past, previous methods (Choe et al., 2005; Spillmann and Teschner, 2008) have been unable to simulate Coulomb friction due to the inability of obtaining its Jacobian. As we have access to the normal contact force Jacobian, we can calculate Coulombic friction forces as well as the friction Jacobian. By having access to the Jacobians of the normal and friction force, our contact model reliably converges even in complex contact states. As the cost of producing the Jacobian is relatively high and leads to having to solve a non-banded matrix, we introduce a hybrid approach which ensures computational speed by only calculating the Jacobian when necessary.

To model contact energy, we compute the minimum distance D between two edges and feed this value into a smooth inverse ReLU function

$$E_c = \frac{1}{K_1} \log\left(1 + e^{K_1(C-D)}\right), \tag{4.1}$$

whose origin is based on the contact distance C ($2h$ for self-contact where h is the cross-sectional radius of the rod) and K_1 is a stiffness term that determines how sharp the curve is.

Intuitively, this function starts to gradually increases the contact energy between two edges as D decreases while $D > C$ and then sharply increases as D approaches C. Although the gradients in the region $D > C$ are nonzero, the inclusion of these "cushioning" forces greatly aid convergence and reduce any unwanted oscillating behavior that can often occur in penalty methods. The effect of these nonzero gradients are explained in detail in Section 4.7.2. Moving on, the key component of the contact model involves obtaining a differentiable analytical expression for D which is difficult as computing the minimum distance between two line segments is a highly nonlinear and noncontinuous process. Next, we briefly describe Lumelsky's min-distance algorithm as well as a new modified smooth approximation that will be used as $D(\mathbf{x})$.

4.3 Analytical Distance via Lumelsky's Algorithm

Lumelsky's algorithm produces the minimum distance between two line segments in $\mathbb{R}^N$ and contains three noncontinuous components; we can eliminate one of them by simply taking the assumption that no edge can be reduced to a point. In other words, each edge must have a finite length greater than zero. With this condition eliminated, we now briefly layout the simplified min-distance algorithm for an arbitrary edge combination $\mathbf{x}_{ij}$. Note that here, we simply go over the steps of the algorithm. For further intuition on how exactly the algorithm is computing the min-distance, please refer to the original paper (Lumelsky, 1985).

To start off the min-distance algorithm, first, we add another "edge" vector $\mathbf{e}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ to the ones already previously formulated: $\mathbf{e}_i$ and $\mathbf{e}_j$. With these vectors, we can then calculate the necessary intermediary values as follows where i and j subscripts are left out for clarity:

$$\begin{aligned}
D_1 &= \mathbf{e}_i \cdot \mathbf{e}_i, \\
D_2 &= \mathbf{e}_j \cdot \mathbf{e}_j, \\
S_1 &= \mathbf{e}_i \cdot \mathbf{e}_{ij}, \\
S_2 &= \mathbf{e}_j \cdot \mathbf{e}_{ij}, \\
R &= \mathbf{e}_i \cdot \mathbf{e}_j, \\
\lambda &= D_1 D_2 - R^2.
\end{aligned} \tag{4.2}$$

Next, denote a fix bound function

$$F(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}, \tag{4.3}$$

where all values above 1 are 1 and all values below 0 are 0. Anything between is outputted identically. This piecewise function is the first of the two remaining noncontinuous components. The rest of the algorithm is as follows where β_1, $\beta_2 \in [0, 1]$ are the ratios that determine at which point along the length of each edge the connecting min-distance vector $\vec{D}$ lies as shown in Figure 4.2. With this in mind, the fix bound function $F(x)$ is used to ensure that these values do not go outside the appropriate range. As two edges become increasingly closer to parallel, λ approaches 0 and becomes 0 when perfectly parallel. To prevent division by zero, a piecewise function is used to describe the assignments of β_1.

$$\begin{aligned}
\beta_1 &:= \begin{cases} (S_1 D_2 - S_2 R)/\lambda & \lambda \neq 0 \\ 0 & \lambda = 0 \end{cases}, \\
\beta_1 &:= F(\beta_1), \\
\beta_2 &:= \frac{\beta_1 R - S_2}{D_2}, \\
\beta_2^f &:= F(\beta_2).
\end{aligned} \tag{4.4}$$

The last noncontinuous component is a conditional assignment where if $\beta_2^f \neq \beta_2$ (i.e. $\beta_2 < 0$ or $\beta_2 > 1$), the β values are reassigned as

$$\begin{aligned} \beta_1 &:= F\left(\frac{\beta_2^f R + S_1}{D_1}\right), \\ \beta_2 &:= \beta_2^f. \end{aligned} \tag{4.5}$$

Finally, D can be computed as

$$D = \left\| \beta_1 \mathbf{e}_i - \beta_2 \mathbf{e}_j - \mathbf{e}_{ij} \right\|. \tag{4.6}$$

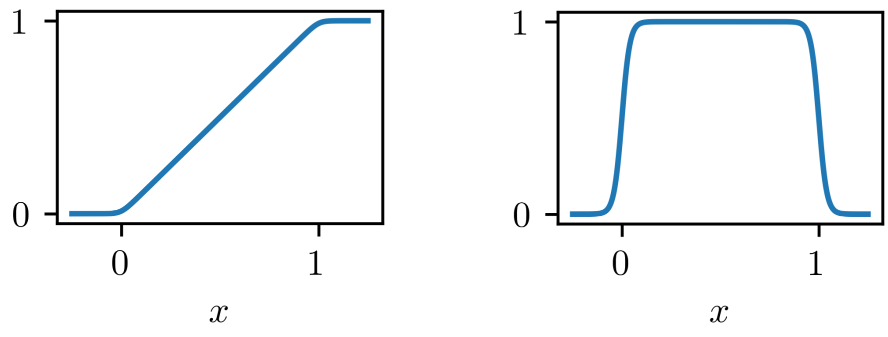
4.4 Smoothing Lumelsky's Algorithm[*]

To obtain the gradient and Hessian of the contact energy $E_c(D(\mathbf{x}))$, $D(\mathbf{x})$ must be differentiable. In the previous section, we introduced an algorithm that can compute D. Now, we modify the min-distance algorithm into a differentiable analytical expression. As (4.2) is analytical, the only necessary modifications lie in (4.4) and (4.5). Firstly, the fixbound function $F(x)$ can be modelled by the smooth approximation

$$H(x, k_h) = \frac{1}{k_h}\Big(\log\big(1 + e^{k_h x}\big) - \log\big(1 + e^{k_h (x-1)}\big)\Big), \tag{4.7}$$

where k_h is a hyperparameter which determines how stiff the curves are. A larger k_h value will result in a more accurate approximation but will result in "stiff" first and second derivatives leading to reduced convergence, thus, this value should be determined empirically. Next is the conditional reassignment in (4.5). As the reassignment only depends on whether or not $\beta_2^f \neq \beta_2$, this is equivalent to the reassignment only occurring

[*]Funny enough, a huge novelty of IMCv1 was this idea to smooth Lumelsky's algorithm. At the time, I was completely unaware of a piecewise smooth analytical solution to the edge-to-edge distance problem. After discovering this solution and testing them both out, I discovered that the formulation in Li et al. (2018) performed better. Still, smoothing Lumelsky's algorithm was indeed a fun mathematical challenge.



(a) Smooth Fixbound Function (b) Smooth Boxcar Function

Figure 4.3: (a) Smooth fixbound function $H(x)$ which models the piecewise function in (4.3). (b) Smooth boxcar function $B(x)$ which allows for an analytical conditional reassignment. Both functions are plotted with a stiffness parameter of $k_h = k_b = 50$.

when $\beta_2 < 0$ or $\beta_2 > 1$. To model this, we can use a smooth boxcar function

$$B(x, k_b) = \frac{1}{1 + e^{-k_b x}} - \frac{1}{1 + e^{-k_b (x-1)}}, \tag{4.8}$$

which consists of two compounded logistic functions. Both functions $H(x, k_h)$ and $B(x, k_b)$ can be viewed in Figure 4.3 for a value of $k_h = k_b = 50$ which is the value used to produce the simulation results.

The last noncontinuous component of the algorithm lies in the piecewise function in (4.4) which is actually left noncontinuous. Although this introduces a piecewise function into the expression, this does not hurt convergence for the following reasons. First, it should be noted that λ will almost never equal exactly 0 due to floating point arithmetic. Therefore, the piecewise function is only required to prevent simulation crashes during simulation starts with perfectly parallel rod configurations. Furthermore, the numeric stability of this algorithm is maintained as whenever λ approaches zero, the numerator $S_1 D_2 - S_2 R$ also approaches zero by a similar magnitude, effectively avoiding numeric overflow problems (Lumelsky, 1985). In terms of performance, we have found that the produced Hessian is an excellent indicator of gradient direction when approaching or passing through parallel configurations when validating against finite difference for a wide variety of edge configurations.

With these two functions and the above prevention of division by zero, we can then replace (4.4) and (4.5) with the expressions

$$
\begin{aligned}
t_1 &= \begin{cases} (S_1 D_2 - S_2 R)/\lambda & \lambda \neq 0 \\ 0 & \lambda = 0 \end{cases}, \\
t_2 &= H(t_1), \\
u_1 &= \frac{t_2 R - S_2}{D_2}, \\
u_2 &= H(u_1), \\
\beta_1 &= (1 - B(u_1)) \frac{u_2 R + S_1}{D_1} + B(u_1) t_2, \\
\beta_2 &= u_2, \\
D &= \left\| \beta_1 \mathbf{e}_i - \beta_2 \mathbf{e}_j - \mathbf{e}_{ij} \right\|, \\
E_c &= \frac{1}{K_1} \log\left(1 + e^{K_1(C-D)}\right).
\end{aligned}
\tag{4.9}
$$

As shown above, the min-distance D is fed into (4.1) which then leads to a fully differentiable analytical expression $E_c(\mathbf{x})$. It should be noted that an end to end differentiation of $E_c(\mathbf{x})$ ends up with an extremely large and complex equation when using symbolic differentiation (Meurer et al., 2017). Therefore, to greatly simplify the expression and improve computational efficiency, the gradient and Hessian for several of the intermediary algorithmic values are taken and then chain ruled together.[*] Effectively, we can define $E_c(\mathbf{x})$ by the functional

$$
f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_{ij}, D_1, D_2, S_1, S_2, R, t_2). \tag{4.10}
$$

Since the inputs for $f(\cdot)$ are all functions of $\mathbf{x}$, chain rule tells us that we can obtain the

[*]Later, we actually discovered that this chain ruling was unnecessary when switching over to more efficient symbolic differentiation tools such as SymEngine (Čertík, 2019), which IMCv2 uses.

gradient of the contact energy as

$$\begin{aligned}\nabla_{\mathbf{x}} E_c = &\frac{\partial f}{\partial \mathbf{e}_i}\frac{\partial \mathbf{e}_i}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{e}_j}\frac{\partial \mathbf{e}_j}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{e}_{ij}}\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{x}} + \\ &\frac{\partial f}{\partial D_1}\frac{\partial D_1}{\partial \mathbf{x}} + \frac{\partial f}{\partial D_2}\frac{\partial D_2}{\partial \mathbf{x}} + \frac{\partial f}{\partial R}\frac{\partial R}{\partial \mathbf{x}} + \\ &\frac{\partial f}{\partial S_1}\frac{\partial S_1}{\partial \mathbf{x}} + \frac{\partial f}{\partial S_2}\frac{\partial S_2}{\partial \mathbf{x}} + \frac{\partial f}{\partial t_2}\frac{\partial t_2}{\partial \mathbf{x}}.\end{aligned} \tag{4.11}$$

Here, we see that for any arbitrary edge combination $\mathbf{x}$, the produced force $-\nabla_{\mathbf{x}} E_c$ will be a vector of size 12 consisting of four concatenated 3-dimensional contact force vectors for every node making up $\mathbf{x}$. These 12 elements contribute to the 12 entries of the $(4N-1)$-sized $\mathbf{F}^{\mathrm{c}}$ vector located at the following positions: $4i-3, 4i-2, 4i-1, 4(i+1)-3, 4(i+1)-2, 4(i+1)-1, 4j-3, 4j-2, 4j-1, 4(j+1)-3, 4(j+1)-2, 4(j+1)-1$. Once the contact forces are computed for every contacting edge combination during a time step, the force values are added to $\mathbf{F}^{\mathrm{c}}$ and then incorporated into DER.

To obtain the Hessian, we simply take (4.11) and differentiate once again to obtain $-\nabla^2_{\mathbf{x}} E_c$. Once obtained, the Hessian is added to the $(4N-1 \times 4N-1)$ sized Jacobian matrix $\mathbf{J}^{\mathrm{c}}$ in a similar manner. The derivation of the Hessian can be done by using the product rule and its derivation is left out for brevity.

4.5 Semi-Explicit Friction

Just as we obtained contact force vectors of size $\mathbb{R}^{12}$, we produce frictional forces in a similar manner where given an edge combination $\mathbf{x}_{ij}$, we compute a friction force vector $\mathbf{F}^{\mathrm{fr}}_{ij} \in \mathbb{R}^{12}$ according to Coulomb's dynamic friction law. For each edge combination, this 12-sized vector is added to the appropriate entries of $\mathbf{F}^{\mathrm{fr}}$ (3.10).

Coulomb's friction law states the following:

1. Frictional force is independent of velocity, and
2. $\left\|\mathbf{F}^{\mathrm{fr}}\right\| = \mu_k F_n$ during sliding,

where μ_k is the dynamic friction coefficient and F_n is the normal force (more details later in this section). From the contact model, we were able to derive $-\nabla_{\mathbf{x}} E_c$ which is equivalently the normal contact forces $\mathbf{F}^{\mathrm{c}}$. As mentioned, the gradient of the contact energy will always produce forces that are along the contact normal. Therefore, we can obtain F_n at the i-th edge of the contact pair $\mathbf{x}_{ij}$ by simply summing up the contact forces on the i-th and $i+1$-th nodes: $F_n = \|\mathbf{F}^{\mathrm{c}}_i + \mathbf{F}^{\mathrm{c}}_{i+1}\|$. We can also use these contact forces to obtain the contact norm $\mathbf{n} = (\mathbf{F}^{\mathrm{c}}_i + \mathbf{F}^{\mathrm{c}}_{i+1})/F_n$. The direction of friction is then determined by the tangential relative velocity $\mathbf{u}$ of edge i with respect to edge j, which can be obtained by

$$\begin{aligned}
\mathbf{v}^{\mathrm{e}}_i &= 0.5(\mathbf{v}_i + \mathbf{v}_{i+1}),\\
\mathbf{v}^{\mathrm{e}}_j &= 0.5(\mathbf{v}_j + \mathbf{v}_{j+1}),\\
\mathbf{v}^{\mathrm{rel}} &= \mathbf{v}^{\mathrm{e}}_i - \mathbf{v}^{\mathrm{e}}_j,\\
\mathbf{u} &= \mathbf{v}^{\mathrm{rel}} - (\mathbf{v}^{\mathrm{rel}} \cdot \mathbf{n})\mathbf{n},\\
\hat{\mathbf{u}} &= \frac{\mathbf{u}}{\|\mathbf{u}\|},
\end{aligned} \tag{4.12}$$

where $\mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{v}_j$, and $\mathbf{v}_{j+1}$ are the nodal velocities of the i-th and j-th edges.

We formulate the friction force on edge i using a modified form of Coulomb's friction equation:

$$\mathbf{F}^{\mathrm{fr,e}}_i = -\mu_k \gamma \hat{\mathbf{u}} F_n, \tag{4.13}$$

where weight $\gamma \in [0, 1]$ eliminates frictional forces between edges with extremely small relative velocities that can cause unwanted behavior.* To maintain differentiability, it is obtained using a smoothed Heaviside step function with the tangential relative velocity as input:

$$\gamma = \frac{1}{1 + e^{-k_f(\|\mathbf{u}\| - c)}}, \tag{4.14}$$

where k_f is the stiffness and c determines the limit for the step transition, which must take into consideration the scaling of the model as explained in Section 4.6. In our experiments,

*Originally, IMCv1 was created with kinetic friction in mind and γ was incorporated to prevent large friction forces for small velocities. But this is essentially simulating sticking-sliding transitions! As a young graduate student at the time, I did not realize that I was indirectly starting to simulate sticking-slipping, something that is fully fleshed out in Chapter 5.

$k_f = 50$ and $c = 0.15$. We can do the same for edge j to obtain $\mathbf{F}_j^{\text{fr,e}}$. The computed frictional forces are then equally distributed to each node and then concatenated four times to form the final friction vector

$$\mathbf{F}_{ij}^{\text{fr}} = \left[0.5\mathbf{F}_i^{\text{fr,e}},\ 0.5\mathbf{F}_i^{\text{fr,e}},\ 0.5\mathbf{F}_j^{\text{fr,e}},\ 0.5\mathbf{F}_j^{\text{fr,e}}\right]^T \in \mathbb{R}^{12\times 1}. \tag{4.15}$$

Once these friction force vectors are computed for every contacting edge combination during a time step, we can then compute the friction force Jacobian matrix $\mathbf{J}_{ij}^{\text{fr}}$ as well. These are then added to $\mathbf{F}^{\text{fr}}$ and $\mathbf{J}^{\text{fr}}$ in exactly the same way as the contact energy gradient and Hessian.

It should be noted that several simplifications were made for the friction model.* Firstly, the relative velocities were computed using the midpoint of the edges rather than the contact points. Likewise, the friction forces were evenly distributed rather than being dependent on the contact points. This was done as it greatly simplifies the friction force Jacobian, leads to improved convergence, and does not have any noticeable effects so long as the rod is sufficiently discretized.

Furthermore, we treat friction semi-explicitly by using the known velocities from the previous time step. This allows the friction direction to remain constant during Newton iterations which improves convergence considerably. Although a fully implicit scheme is possible, computing the necessary contact Hessian on every iteration is costly and the overall speed of the algorithm greatly benefits from this formulation.

In terms of limitations, this method clearly does not enforce static friction and so should only be used for continuous sliding scenarios such as knot tying. Furthermore, friction occurring due to the rod twist θ is not modeled. When an edge undergoes enough twist and is in contact with a receiving edge, friction forces occur slightly off the centerline of this receiving edge. As our model assumes that all friction occurs precisely on the centerline and formulates all contact only using $\mathbf{x}$, such friction-twist coupling is neglected.

*All such simplifications are remedied in Chapter 5.

Lastly, the produced friction forces can possibly overtake the pull forces when the pull speed is very low leading to unrealistic sliding in the opposite direction. This must be remedied by pulling at a sufficiently high speed.

4.6 Algorithmic Components

In addition to the force and Jacobian generation, there are several additional steps to the IMCv1 algorithm that are explained in this section as well as several hyperparameters that must be properly tuned for optimal performance and convergence which are listed below.

1. δ^{col}, the collision limit,
2. k, the contact stiffness,
3. K_1, the contact energy stiffness, and
4. ω, the number of iterations before the hybrid algorithm computes the Jacobian

4.6.1 Scaling

First, the nodal coordinates are scaled by a scaling factor $S = 1/h$ so that the adjusted rod radius equals a unit value of 1 (i.e., $C = 2$ for self-contact). To ensure that the distance at which two edges experience a force is very close to the rod surface, the following energy function is used:

$$E_c = \frac{1}{K_1} \log\Big(1 + e^{K_1\left(2 - \frac{D}{h}\right)}\Big). \tag{4.16}$$

4.6.2 Collision Limit[*]

Following this, the collision limit δ^{col} is the threshold value used to determine when two edges are "in contact". This value is fed into a collision detection algorithm which returns

[*]Looking back, requiring the user to manually choose the collision limit δ^{col} was a rudimentary solution to say the least. IMCv2 resolves all such tedium in Chapter 5.

all edge combinations falling into this threshold which is denoted by the set

$$\mathcal{C} = \left\{ \mathbf{x}_{ij} \in \mathcal{X} \mid D_{ij}/h < 2 + \delta^{\text{col}} \right\}. \tag{4.17}$$

The minimum distance from this set is denoted by $D^{\min} = \min_{\mathbf{x} \in \mathcal{C}} D(\mathbf{x})$, which will be used later to adjust the contact stiffness k accordingly.

The collision limit δ^{col} must be chosen carefully as a higher δ^{col} value results in additional computation due to more qualifying edge combinations whereas a δ^{col} value that is too low will produce non-smooth gradients that hamper convergence.

A good way to determine a proper δ^{col} value is to observe the plotted contact energy function from (4.16) for a chosen K_1 value. By observing the contact energy curve, the point at which the generated gradients are ≈ 0 can be found when the slope of curve is nearly flattened out. Choosing a δ^{col} value that encompasses this region ensures that the generated gradients are sufficiently smooth. An example of this process can be found in Figure 4.4.

The stiffness of the contact forces is determined by the contact energy stiffness value K_1. As this value becomes higher, the region above the contact surface at which two edges experience a force decreases. This leads to stiff contact which is more physically accurate as realistically the contact energy should be zero whenever $D > C$ and shoot to ∞ as soon as $D = C$. On the other hand, a K_1 value that is too large will result in convergence issues while a K_1 value that is too low will have excessive oscillations between the contact bodies. A value that was found to be a good compromise between physical accuracy and convergence was $K_1 = 50$.

4.6.3 Adaptive Contact Stiffness

The next hyperparameter that must be specified is the contact stiffness k, a scalar value that is used to scale the contact force and Jacobian. This value is adaptively readjusted every time step to ensure that excessive hovering or penetration is minimized and so only

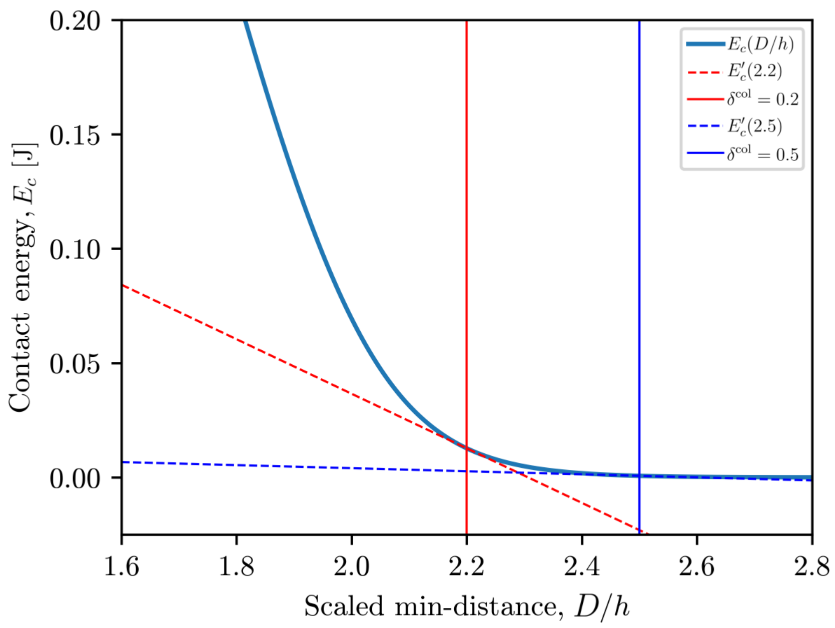


Figure 4.4: Contact energy curve of (4.16) for $K_1 = 10$. Scaling by $S = 1/h$ results in the curve being centered at a self-collision length of $C = 2.0$. As denoted by the solid blue line, the curve starts to flatten out to zero at $D/h = 2.5$ which indicates that generated gradients are ≈ 0. Therefore, a collision limit $\delta^{\rm col} = 0.5$ would be suitable. Conversely, $\delta^{\rm col} = 0.2$ as denoted by the red line would result in the Newton solver potentially failing to converge due to non-smooth force generation.

the initial value must be specified which can be found empirically. This initial k should be reasonably close to the value that would result in $D^{\rm min} = C$. In our experiments, we employed a simple algorithm that decreased or increased k by a fraction of a percent depending on whether or not $D^{\rm min}$ was $< C - \epsilon_1$ or $> C + \epsilon_2$ where ϵ_1 and ϵ_2 are limits indicating an acceptable contact range. This algorithm updates k only when $D^{\rm min}$ is deviating from the region defined by $[C - \epsilon_1, C + \epsilon_2]$ and is otherwise left constant to prevent overshooting.

4.6.4 Hybrid Formulation[*]

As mentioned previously, this algorithm applies a hybrid approach in which the Jacobian of the contact and friction forces are only computed once the number of iterations passes a limit ω. This is done because often convergence can be quickly obtained even without the contact Jacobian and with the absence of the contact Jacobian, the overall Jacobian matrix remains a banded matrix which can be solved significantly faster. Although the Jacobian results in a decrease in iteration count, the consequent increase in computational time outweighs this benefit as IMCv1 is able to reliably converge rapidly without it for a majority of time steps. With this in mind, the Jacobian is crucial for completing volatile contact states with high velocities and impacts that could otherwise end the simulation prematurely. Therefore, this hybrid approach maximizes computational speed while ensuring that the simulation can consistently reach the next time step during especially difficult contact scenarios such as inversion and the initialization phase where the rod rapidly reverts to its lowest energy state. The limit ω should be chosen empirically so that the Jacobian is only generated when necessary. In our experiments, we used an ω of 20.

4.6.5 Newton Damper[†]

Lastly, although not a hyperparameter, a damping coefficient α is used to reduce the step size of the Newton solver as the number of iterations increase for a particular time step. For this, a simple decaying algorithm is used which reduces α by a factor of 2 every other iteration. Overall, aside from the collision detection algorithm (*which is only performed on the first iteration of every time step*), the time complexity of IMCv1 with and without Jacobian generation is $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ respectively, where n is the number of collisions

[*]This hybrid formulation was created because our experiments showed that the Hessian of IMCv1 had sometimes little impact on the convergence. After the publication of IMCv1, it was discovered that our Hessian chain ruling contained an error and was off by a scale factor. After resolving this error, the Hessian aided convergence greatly and thus, the hybrid formulation was no longer needed.

[†]Replaced with Goldstein-Price line search in Section 5.5.3.

Algorithm 1: IMCv1

```
   Parameters: $k, \delta^{\text{col}}, \omega, S$
   Input: $\mathbf{x}, \mathbf{v}, n$                                  // from DER
   Output: $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}, \alpha$
1  Function IMCv1($\mathbf{x}$, $\mathbf{v}$, $n$):
2      scale $\mathbf{x}$ and $\mathbf{v}$ by $S$
3      if $n == 0$ then                                     // run only on first iter
4          $\mathcal{C}, D^{\text{min}} \leftarrow$ collisionDetection($\mathbf{x}, \delta^{\text{col}}$)
5          $k \leftarrow$ updateConStiffness($k, D^{\text{min}}$)
6      end
7      if $n < \omega$ then                                 // compute only forces
8          $\mathbf{F}^{\text{c}} \leftarrow$ genContact($\mathcal{C}$)
9          $\mathbf{F}^{\text{fr}} \leftarrow$ genFriction($\mathcal{C}, \mathbf{v}, \mathbf{F}_c$)
10         $\mathbf{J}^{\text{c}} \leftarrow$ zero square matrix
11         $\mathbf{J}^{\text{fr}} \leftarrow$ zero square matrix
12     end
13     else                                     // compute Jacobian for convergence
14         $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}} \leftarrow$ genContact($\mathcal{C}$)
15         $\mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}} \leftarrow$ genFriction($\mathcal{C}, \mathbf{v}, \mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}$)
16     end
17     scale $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}$ by $k$
18     $\alpha \leftarrow$ newtonDamper($n$)
19     return $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}, \alpha$
```

detected. The full contact algorithm as well as its implementation in DER can be seen in Alg. 1 and 2, respectively. In Alg. 2, the term "free" are the indices that correspond to the free degrees of freedom of the elastic rod. The remaining degrees of freedom are "fixed" and depends on the user defined boundary conditions.

4.6.6 Viscous Damping[*]

Since IMCv1 is inherently a penalty method, the inclusion of damping forces greatly aid the stability of the contact model. A simple viscous damping force is applied to the elastic

[*]This viscous environmental damping was originally used to reduce unwanted oscillations. Once the Hessian chain ruling error was remedied, damping was found to no longer be required. Still this damping force can be very useful for providing stability in high energy settings, as shown later in Chapter 6.

Algorithm 2: DER with IMCv1

```
Input: $\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$
Output: $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$
Require: boundary conditions $\rightarrow$ free
1  Function DER($\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$):
2      Guess: $\mathbf{q}^{(1)} \leftarrow \mathbf{q}(t_i)$
3      $n \leftarrow 0,\ \epsilon \leftarrow \infty$
4      while $\epsilon >$ tolerance do
5          $\mathbf{F}^{\text{int}} \leftarrow$ genForces($\cdot$)
6          $\mathbf{J}^{\text{int}} \leftarrow$ genJacobian($\cdot$)                    // $\frac{\partial^2 E_{\text{elastic}}}{\partial q_i \partial q_j}$ in (3.11)
7          $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}, \alpha \leftarrow$ IMCv1($\mathbf{x}^{(n)}, \dot{\mathbf{q}}(t_i), n$)          // Alg 1
8          $\mathbf{F}_{\text{der}} \leftarrow$ left side of (3.10)
9          $\mathbf{J}_{\text{der}} \leftarrow$ left side of (3.11)
10         $\mathbf{F}_{\text{free}} \leftarrow \mathbf{F}_{\text{der}}$(free)        // Downsize to only include free DOFs
11         $\mathbf{J}_{\text{free}} \leftarrow \mathbf{J}_{\text{der}}$(free, free)
12         $\Delta\mathbf{q}_{\text{free}} \leftarrow (\mathbf{J}_{\text{free}})^{-1}\, \mathbf{F}_{\text{free}}$                  // Solve $\mathbf{J}_{\text{free}}\Delta\mathbf{q}_{\text{free}} = \mathbf{F}_{\text{free}}$
13         $\mathbf{q}^{(n+1)}$(free) $\leftarrow \mathbf{q}^{(n)}$(free) $- \alpha\Delta\mathbf{q}_{\text{free}}$
14         $\epsilon \leftarrow \|\mathbf{F}_{\text{free}}\|$                                  // update error
15         $n \leftarrow n + 1$
16     end
17     $\mathbf{q}(t_{i+1}) \leftarrow \mathbf{q}^{(n)}$
18     $\dot{\mathbf{q}}(t_{i+1}) \leftarrow (\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i))/\Delta t$
19     return $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$
```

rod by applying a force

$$\mathbf{F}^{\text{v}} = -\eta \left(\frac{\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)}{\Delta t} \right) V \tag{4.18}$$

on a node where η (Pa $\cdot$ s) is a viscosity coefficient and V is the Voronoi length.

The Jacobian of this damping force is then simply

$$\mathbf{J}^{\text{v}} = -\frac{\eta V}{\Delta t}\mathbb{I}, \tag{4.19}$$

where $\mathbb{I}$ is the square identity matrix of size $(4N - 1)$. This damping force was also added to SPT for fair comparison. For all simulations, a viscosity value of $\eta = 0.01$ Pa $\cdot$ s was used unless otherwise specified.

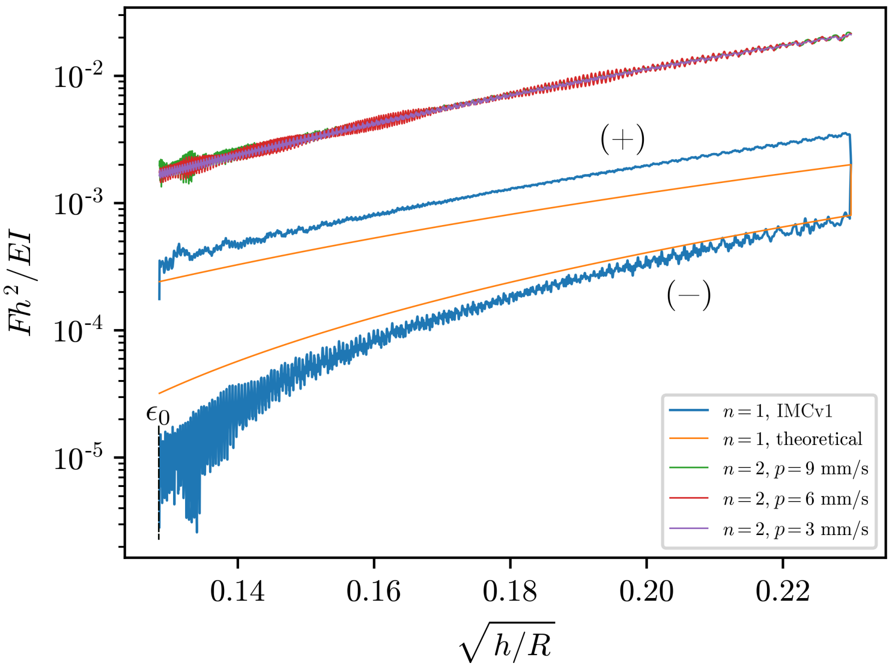


Figure 4.5: Model validation and comparison with theory. In the above half is the pull force comparison for $n = 2$ with different pull speeds p and friction $\mu_k = 0.10$. As shown, the pull forces are identical in magnitude indicating that friction is independent of velocity. In the bottom half is the $n = 1$ simulation data comparison with Audoly's predictive model (Audoly et al., 2007) shown in (4.20). Starting with an open trefoil knot, the knot is tightened and then loosened with $\mu_k = 0.10$ which is indicated by (+) and (−) respectively. A moving average of 200 steps is used for the $n = 1$ pull forces to minimize visually large variations caused by the lower end of the log scale.

4.7 Overhand Knot Tying Validation

In this section, we first validate the correctness of our contact model against theory. Afterwards, to observe the benefits of using IMCv1, we compare simulation results with the contact model (SPT) proposed by Spillmann and Teschner (2008) for knots of unknotting numbers $n \in [1, 4]$ which are shown in Figure 4.1. For both methods, the contact model is used to simulate knot tying until a mutual termination state which is shown in the rightmost column of Figure 4.1. Lastly, we compare the computational efficiency and convergence properties between the two methods. All simulations for IMCv1 used a contact energy stiffness $K_1 = 50$ and a collision limit $\delta^{\text{col}} = 0.15$ which was obtained using the method mentioned in Figure 4.4.

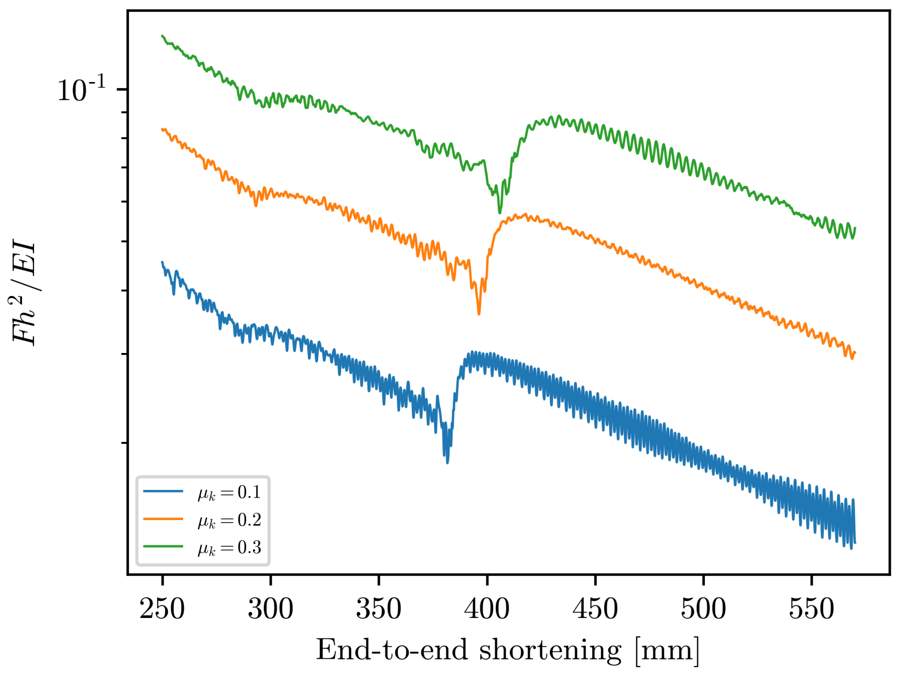


Figure 4.6: Pull force comparison for $n = 4$ and different μ_k values. There is a clear monotonically increasing relationship between pull forces and μ_k. Additionally, inversion (indicated by the sudden drop in pull force) occurs earlier for higher μ_k values as expected. A pull speed of 6, 9, and 12 mm/s was used for $\mu_k = 0.1$, 0.2, and 0.3, respectively. This was necessary as higher friction coefficients induced the limitation mentioned in Section 4.5 for higher pull speeds.

4.7.1 Theoretical Validation

Coulomb's friction law states that friction is independent of velocity. To show that our model abides by this, we plot the pull forces F in Newtons (N) when tightening a knot with unknotting number $n = 2$ with friction coefficient $\mu_k = 0.10$ for pull speeds p of 3, 6, and 9 mm/s (*pulled from both ends*). A 1 meter long rod with cross-sectional radius $h = 1.6$ mm, density $\rho = 1180\,\text{kg/m}^3$, and Young's Modulus $E = 1.8\text{e}5$ Pa is used and is discretized into 301 nodes. We plot a regularized pull force Fh^2/EI against $\sqrt{h/R}$ where $EI = \pi E h^4/4$ is the flexural modulus and R is the radius of the knot loop. The radius R can be computed using the knot circumference from Figure 4.1 as $R = e/(2\pi)$. As Figure 4.5 shows, the magnitude of the $n = 2$ pull forces are approximately the same for all pull speeds. Dynamic friction force in Coulomb's model is independent of velocity and therefore this observation supports the physical correctness of the generated friction forces.

Next, the pull forces for a trefoil knot ($n = 1$) are compared with the predictive model by Audoly et al. (2007). This model states the following theoretical equivalence

$$\frac{Fh^2}{EI} = \frac{\epsilon^4}{2} \pm \mu_k \sigma \epsilon^3, \tag{4.20}$$

where σ is a numerical constant ($\sigma = 0.492$ for trefoil knots), $\epsilon = \sqrt{h/R}$, and the $\pm$ term is the frictional component from tightening ($+$) and loosening ($-$).

Using the same rod properties as mentioned, a trefoil knot is tightened and then loosened at 3 mm/s using IMCv1. Here we reduce η to 0.0005 as loosening can be sensitive to small forces. As shown in Figure 4.5, the recorded pull forces roughly follow the curves of the predictive model albeit with some displacements when tightening increases sufficiently. This can be attributed to imperfections in the predictive model as (4.20) is an elegant lightweight solution that does not perfectly model friction when the knot becomes sufficiently tight, i.e. $\sqrt{h/R}$ becomes large. Still, aside from the displacement, the pull forces follow the rate of increase/decrease of the predictive model well which is a good indicator of correctness.

Furthermore, in Figure 4.5, during the loosening, the trefoil knot can be seen being locked by friction at a point $\epsilon_0 = \sqrt{h/R_0} \approx 0.1285$. From the right side of (4.20), we can rearrange the terms to obtain

$$\mu = \frac{1}{2\sigma}\epsilon_0 = 1.02\sqrt{h/R_0}. \tag{4.21}$$

When plugging in our obtained ϵ_0 value, we obtain $\mu_{\text{theory}} = 0.1306$ which is reasonably close to the friction coefficient used in simulation, $\mu_k = 0.10$.

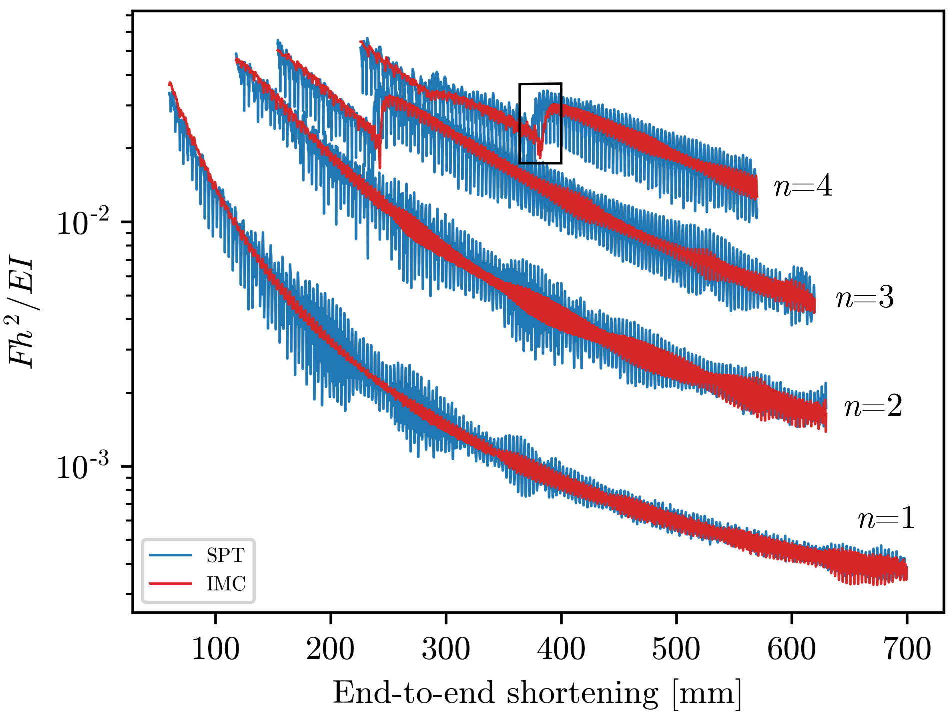
Finally, we show that the pull forces monotonically increase with μ_k in Figure 4.6. Here, we consider the $n = 4$ case. When recording the pull forces for μ_k of $0.1, 0.2$, and 0.3, we can see that the rate of increase is held constant while the magnitudes monotonically increase. Furthermore, we can see that as friction increases, the inversion point occurs sooner, which indicates that the point of inversion is highly dependent on μ_k.

4.7.2 Pull Force Accuracy

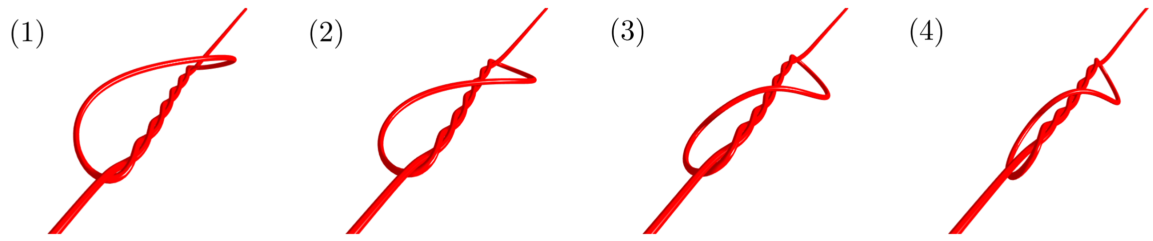
Simulations of knot pulling are performed for both IMCv1 and SPT using a pull speed $p = 6\,\text{mm/s}$, time step $\Delta t = 0.5\,\text{ms}$, friction coefficient $\mu_k = 0.10$, and the same rod properties from Section 4.7.1. With these settings, knot tying was simulated for each method with the experienced pull forces being recorded at each time step. Figure 4.7a plots these pull forces with respect to the end-to-end shortening. Here we see that for both IMCv1 and SPT, as the end-to-end shortening decreases (*knot is pulled tight*), the pull forces increase identically as expected. They also increase in magnitude as the unknotting number n goes up and for $n = 3$ and $n = 4$, inversion occurs which is indicated by the sudden drop in pull force. This is shown visually in Figure 4.7b for $n = 4$.

The largest difference between the methods can be seen in the considerable amount of force jittering by SPT which occurs due to the time step being too large. This leads to visually unrealistic results where the knot continuously "trembles" while being tied. On the other hand, IMCv1 produces much smoother pull forces which directly translate to visually smooth simulations for the same time step size. One caveat that should be noted is that SPT ensures exact non-penetration during contact whereas IMCv1 allows small penetrations and hovering to occur which is physically unrealistic. Still, through the adaptive contact stiffness and the inclusion of damping, IMCv1 contains any hovering to stay within 20 µm above the contact surface while penetrations rarely exceed 5 µm (for comparison, cross-sectional radius is $h = 1.6\,\text{mm}$). Thus, this minor error is largely indiscernible both visually and physically.

Overall, the enforcement of non-penetration at every time step limits the maximal time step that SPT can take without experiencing significant force jittering whereas IMCv1 produces smooth results in exchange for contact varying within a small region.



(a) IMCv1 vs. SPT Pull Force Comparison



(b) Inversion Snapshots for $n = 4$

Figure 4.7: Simulation results comparison. (a) Pull force comparison for unknotting numbers 1 through 4 using SPT (Spillmann and Teschner, 2008) and IMCv1 with $p = 6\,\text{mm/s}$, $\Delta t = 0.5\,\text{ms}$, and $\mu_k = 0.10$. (b) Inversion occurring for $n = 4$ and the corresponding drop in pull forces shown by the border box in (a).

4.7.3 Runtime

Next, we discuss run-time comparisons as well as convergence characteristics. Here, we show that in addition to IMCv1 producing smoother results than SPT at sufficiently large time steps, IMCv1 is also computationally competitive. Both models use the same DER implementation which is written in C++. One discrepancy is that while SPT is directly implemented into DER in C++, IMCv1 is written entirely in Python for ease

Table 4.1: IMCv1 vs. SPT (Spillmann and Teschner, 2008) run time data, $\mu_k = 0.10$, $\Delta t = 0.5\,\text{ms}$, $p = 6\,\text{mm/s}$. AIPT stands for average iterations per time step. ATPI stands for average time per iteration. Iters indicates the total number of Newton's iterations that were necessary to complete the simulation. The time is the total computational time to completion.

Model	n	AIPT	ATPI [ms]	Total Iters	Time [s]
IMCv1	1	2.23	2.24	245824	551
	2	2.67	2.20	245830	542
	3	3.04	2.18	261707	570
	4	3.24	2.12	239987	509
SPT	1	8.25	2.30	907541	2085
	2	9.28	2.26	853935	1931
	3	10.25	2.26	881907	1990
	4	10.94	2.21	810160	1790

of prototyping.[*] To minimize any performance differences arising from using different languages, we employed an LLVM-based Python JIT compiler (Lam et al., 2015) for certain computational intensive portions of the code such as the chain ruling procedure from (4.11).

The computational time for each iteration for both contact methods is recorded using the ctime library. This timing was done so that any computational time arising from IO usage recording the data and rendering the rod graphically were excluded. One thing to note is that the timings for IMCv1 include all of the shared memory overhead between the C++ and Python programs. Therefore, a significant performance increase for IMCv1 can be expected when fully implemented in C++ and compiled without this overhead. Finally, both methods used identical Newton tolerances for all simulations and all simulations were run on a single thread on an Intel Core i7-9700K 3.60GHz CPU.

In Table 4.1, the runtime, total number of iterations, average iterations per time step (AIPT), and average time per iteration (ATPI) are reported. For all knots, we can immediately see that IMCv1 converges with a noticeably smaller amount of iterations than SPT. While the ATPI between both methods are about equal, the simulations for

[*]IMCv2 is coded entirely in C++.

IMCv1 finish approximately 4$\times$ faster than SPT for all knots. For both methods, AITC increases as the knot complexity n increases as expected.

One observation was that the number of iterations to complete a time step increased for IMCv1 as the knot loop became extremely small and/or tightly inverted. Dynamically reducing the time step solves this issue but this was left out so that all reported results were for a constant time step. For the same time step size, SPT can more reliably converge when the knot loop becomes very small albeit with the large force jittering still present. Therefore, it may be worth investigating performance differences between the two methods when SPT uses a time step size that is small enough to compete with the smoothness of IMCv1 while IMCv1 starts at a larger time step and dynamically reduces as convergence becomes an issue.

Still, even for a constant time step size of 0.5 ms, IMCv1 is seen to be more computational efficient when pulling the knots close to taut and for the majority of the knot tying procedure, takes far less iterations to converge. This difference should only increase with IMCv1 being implemented directly into DER in C++.

4.8 Conclusion

In this chapter, we introduced a novel contact model for DER simulations in which the contact forces are handled implicitly using a smooth penalty force formulation. This model was shown to be able to model dynamic friction and simulate knot tying accurately. We showcased comparisons with previous methods (Spillmann and Teschner, 2008) and concluded that our method was capable of producing more visually smooth realistic results while also producing physically accurate data. Furthermore, our method was stable, computationally competitive, and took minimal iterations to converge.

Although this method has shown promising results for knot tying simulations, it is not without its drawbacks. Firstly, IMCv1 is largely unsuitable for contact scenarios that frequently involve sudden excessively large velocities and impacts as these will result in excessive penetration and possible overshoot of the contacting body without appropriate damping. Where IMCv1 shines is scenarios with constant sliding contact such as knot tying where contact forces may or may not gradually rise.

Additionally, to be absolutely physically realistic, contact forces should equal zero when $D > C$ whereas the usage of (4.1) produces a force $\mathbf{F}^c \neq 0$ when $D > C$. It has been shown that employing a smooth approximation such as this can greatly improve convergence in penalty methods (Durville, 2012) which is one of the goals for this algorithm. As the contact forces approach zero as K_1 becomes sufficiently large while $D > C$, this is not a significant problem as discussed in Section 4.7.2. Still, the fact that $\mathbf{F}^c$ is not exactly zero is something to consider. Lastly, the large amount of hyperparameters leaves something to be desired as tuning may be necessary when switching between rod properties which can be time consuming.

Some possible future research directions involve modifications that further improve the realism of the contact model. One of these pertains to the contact stiffness parameter k. A simplification that was employed is the usage of a global stiffness parameter. More realistic contact can be simulated using local stiffness parameters as shown previously by Durville (2012) in exchange for more computation. This addition may also fix the problem where friction forces overtake the pull force of the knot if the pull speed is too low as mentioned in Section 4.5.

Finally, another challenging problem that remains is the proper modelling of static friction. Although dynamic friction is adequately modelled, ultimately, subtle frictional threshold events such as the transition from sticking to sliding and vice-versa are necessary to simulate realistic contact outside the realm of constant sliding. In the next chapter, many of these concerns will be addressed with a more sophisticated frictional contact formulation, IMCv2.

CHAPTER 5

IMCv2: A Fully Implicit Formulation for Contact and Friction

5.1 Introduction

In this chapter, we formulate IMCv2, a fully-implicit penalty-based contact model for frictional contact based on our previous work (IMCv1) in Chapter 4 (Choi et al., 2021). We improve upon this iteration by (1) reformulating frictional contact to be fully-implicit for enhanced physical accuracy, (2) squaring our contact energy term for more rigid contact, (3) changing our smoothly approximated distance formulation to a more stable piecewise analytical formulation, and (4) adding a line search method for increased robustness. Our proposed numerical framework can generate contact for any rod-rod contact scenario and can also generate contact for 3D meshes with proper alterations.

Similar to Chapter 4, we denote the following vector concatenation describing an edge-to-edge contact pair: $\mathbf{x}_{ij} := [\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1}]^T \in \mathbb{R}^{12}$, where $|j-i| > 1$ to exclude consecutive edges from consideration when enforcing contact. We describe the set of all valid edge combinations as $\mathcal{X}$. In future equations, we simply denote the subscriptless $\mathbf{x}$ as an arbitrary edge combination for clarity. We design a contact energy $E_c(D(\mathbf{x}))$ to increase as the minimum distance D (Figure 4.2) between two edge centerlines approaches a contact threshold C ($2h$ for self-contact where h is the radius). With this, the contact energy gradient $-k\nabla_{\mathbf{x}} E_c \in \mathbb{R}^{12}$ is used as the contact forces while the contact energy Hessian $-k\nabla^2_{\mathbf{x}} E_c \in \mathbb{R}^{12\times 12}$ is used as the contact force Jacobian, where k is the contact stiffness which scales the contact forces appropriately to enforce non-penetration. In the upcoming sections, we will now formulate contact energy $E(D)$, minimum distance

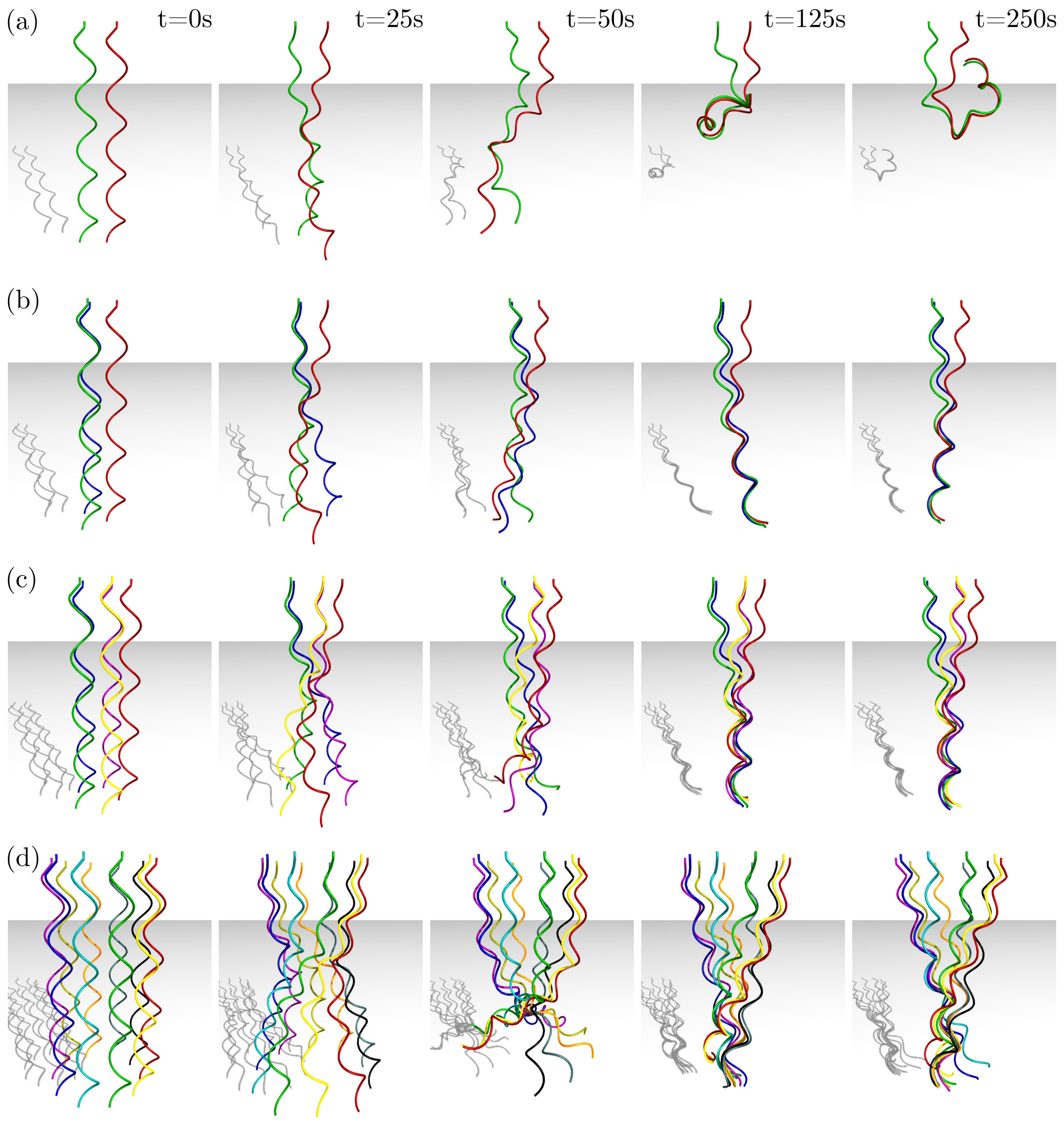


Figure 5.1: Rendered snapshots of flagella bundling with varying amounts of flagella. Rows contain (a) $M = 2$, (b) $M = 3$, (c) $M = 5$, and (d) $M = 10$ flagella. Each column indicates the flagella configuration at the moment of time indicated in the top row.

between two edges $D(\mathbf{x})$, as well as friction.

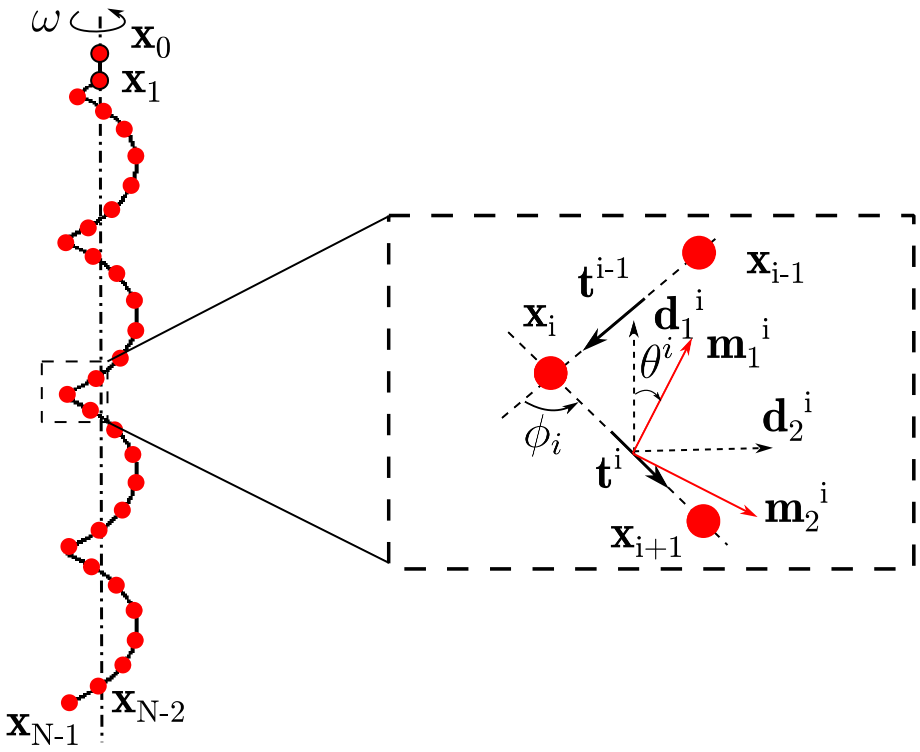


Figure 5.2: Discrete schematic of a flagellar elastic rod. Nodes $\mathbf{x}_0$, $\mathbf{x}_1$, and the edge between $\mathbf{x}_0$ and $\mathbf{x}_1$ is clamped along the dashed centerline and rotated with an angular velocity ω. The rest of the nodes constitute the helical flagellum which revolves around the centerline. A zoomed in snapshot of two edges showcasing their reference frame, material frame, turning angles, and twist angles is shown.

5.2 Improved Contact Energy

In the ideal setting, contact energy must satisfy two properties: (1) it is zero for any distance $D > C$ and (2) it is non-zero at exactly distance $D = C$. A Heaviside step function can essentially describe these properties. Such a function is non-smooth with a very sudden discontinuous change in value, and therefore, cannot be solved reliably by root-finding algorithms such as Newton's method. To remedy this, IPC (Li et al., 2018) uses the following energy formulation to smoothly approximate contact:

$$E_c^{\mathrm{IPC}}(D,\delta) = \begin{cases} -(D-(C+\delta))^2 \ln\left(\frac{D}{C+\delta}\right), & D \in (C, C+\delta) \\ 0 & D \geq C+\delta, \end{cases} \tag{5.1}$$

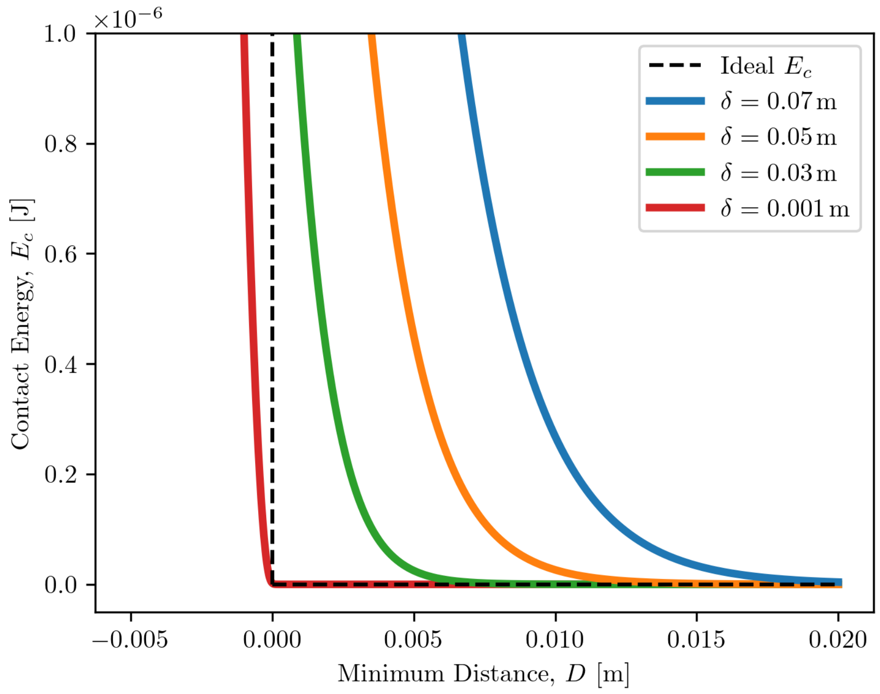


Figure 5.3: Plots for (5.2) with varying δ values. Note that some of the tolerances displayed are unrealistically large for clarity.

where δ is the distance tolerance that defines the region $(C, C+\delta)$ for which non-zero forces are experienced. This contact energy approaches ∞ when D decreasingly approaches C and is therefore undefined for the region $D \leq C$. Although this barrier formulation allows IPC to strictly enforce non-penetration, the solver must be careful never to allow any contact pairs in the penetration zone and/or venture into this undefined region during the optimization process. This is ensured by the inclusion of a custom line search method which conservatively sets an upper bound for the Newton update coefficient α.

In contrast to this, we design our energy formulation to allow for optimization into the penetrated region, thus expanding the range contact forces are experienced from $D \in (C, C+\delta)$ to $D \in (0, C+\delta)$. This in turn allows us to take advantage of more aggressive line search methods, which leads to faster convergence for the flagella contact problem. Although this in theory allows our model to be susceptible to penetration, a sufficient contact stiffness k remedies this issue. In addition, to further ensure non-penetration, we take our previous energy formulation from Choi et al. (2021) and square it so that our gradient grows exponentially instead of linearly. In the end, we use the smooth approximation

$$E_c(D,\delta) = \begin{cases} (C-D)^2 & D \in (0, C-D] \\ \left(\frac{1}{K_1}\log\left(1+e^{K_1(C-D)}\right)\right)^2 & D \in (C-\delta, C+\delta) \\ 0 & D \geq C+\delta, \end{cases} \tag{5.2}$$

where $K_1 = 15/\delta$ indicates the stiffness of the energy curve.

We incorporate the piecewise term $(C-D)^2$ for two reasons. First, this term equivalently models our energy formulation for the region $D \leq C-\delta$ and has a simpler gradient and Hessian, resulting in computational efficiency. Second, and more importantly, the piecewise term also ensures numeric stability by preventing the exponential term in (5.2) from exploding. We show our plotted energy term in Figure 5.3 for various δ values. As shown, the energy starts to increase at an exponential rate as D decreases towards the contact limit which is shown as 0 here. As δ decreases, more realistic contact is achieved (enhancing accuracy) in exchange for a stiffer equation (more difficult to converge).

5.3 Piecewise Continuous Distance

As mentioned in Li et al. (2020a), the minimum distance between two edges $(\mathbf{x}_i, \mathbf{x}_{i+1})$ and $(\mathbf{x}_j, \mathbf{x}_{j+1})$ can be formulated as the constrained optimization problem

$$D = \min_{\beta_i, \beta_j} ||\mathbf{x}_i + \beta_i(\mathbf{x}_{i+1} - \mathbf{x}_i) - (\mathbf{x}_j + \beta_j(\mathbf{x}_{j+1} - \mathbf{x}_j))|| \ni 0 \leq \beta_i, \beta_j \leq 1, \tag{5.3}$$

where β_i and β_j represent the contact point ratios along the respective edges. Minimum distance between two edges can be classified into three distinct categories: point-to-point, point-to-edge, and edge-to-edge. As the names suggest, these classifications depend on which points of the edges the minimum distance vector $\vec{D}$ lies as described by β_i and β_j shown in Figure 4.2.

In the previous chapter (Choi et al., 2021), we altered Lumelsky's edge-to-edge minimum distance algorithm (Lumelsky, 1985) (which implicitly computes the β values) to be fully differentiable through smooth approximations. In this chapter, we now change the distance formulation to use piecewise analytical functions as shown below in (5.4), (5.5), and (5.6), similar to Li et al. (2020a), as we found more stable performance compared to our smooth formulation despite the non-smooth Hessian when changing between contact categories.

We now describe the conditions for each contact type classification. First, if $\vec{D}$ lies on the ends of both edges (i.e. both β constraints are active), then the distance formulation degenerates to the point-to-point case which can easily be solved using the Euclidean distance formula

$$D^{PP} = ||\mathbf{x}_a - \mathbf{x}_b||, \tag{5.4}$$

where $\mathbf{x}_a$ and $\mathbf{x}_b$ are the nodes for first and second edges in contact, respectively.

If $\vec{D}$ only lies on one end of one rod (i.e. only one β constraint is active), then the contact type degenerates to point-to-edge. This can be solved as

$$D^{PE} = \frac{||(\mathbf{x}_a - \mathbf{x}_b) \times (\mathbf{x}_b - \mathbf{x}_c)||}{||\mathbf{x}_a - \mathbf{x}_b||}, \tag{5.5}$$

where $\mathbf{x}_a$ and $\mathbf{x}_b$ are the nodes of the edge for which the minimum distance vector does not lie on an end and $\mathbf{x}_c$ is the node of the edge which the minimum distance vector does lie on. Finally, edge-to-edge distance (i.e. no active constraints) for the i-th and j-th edges can be solved as

$$\begin{aligned} \mathbf{d} &= (\mathbf{x}_{i+1} - \mathbf{x}_i) \times (\mathbf{x}_{j+1} - \mathbf{x}_j), \\ D^{EE} &= |(\mathbf{x}_i - \mathbf{x}_j) \cdot \hat{\mathbf{d}}|, \end{aligned} \tag{5.6}$$

where $\hat{}$ indicates a unit vector. With D fully defined, this concludes our contact energy formulation. To correctly classify contact pairs, we use Lumelsky's algorithm (details in Section 4.3) to compute β values.

Algorithm 3: IMCv2

```
Input: $\mathbf{x}, \mathbf{x}_0, k, \delta, \nu$
Output: $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}$
1 Function IMCv2($\mathbf{x}, \mathbf{x}_0, k, \delta, \nu$):
2     $\mathbf{v} \leftarrow (\mathbf{x} - \mathbf{x}_0)/\Delta t$                              // compute velocity
3     $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}} \leftarrow$ genContact($\mathbf{x}, \delta$)                          // (5.2)
4     $\mathbf{F}^{\text{c}} \leftarrow k\mathbf{F}^{\text{c}}$                              // scale by contact stiffness
5     $\mathbf{J}^{\text{c}} \leftarrow k\mathbf{J}^{\text{c}}$                                         // $\mathbf{J}^{\text{c}} \equiv \nabla_{\mathbf{x}}\mathbf{F}^{\text{c}}$
6     $\mathbf{F}^{\text{fr}} \leftarrow$ genFriction($\mathbf{x}, \mathbf{v}, \mathbf{F}^{\text{c}}, \nu$)                    // (5.11)
7     $\nabla_{\mathbf{x}} f, \nabla_{\mathbf{F}^{\text{c}}} f \leftarrow$ genFrictionPartials($\mathbf{x}, \mathbf{v}, \mathbf{F}^{\text{c}}, \nu$)    // $\mathbf{F}^{\text{fr}} \equiv f(\mathbf{x}, \mathbf{F}^{\text{c}})$
8     $\mathbf{J}^{\text{fr}} \leftarrow \nabla_{\mathbf{x}} f + \nabla_{\mathbf{F}^{\text{c}}} f \nabla_{\mathbf{x}}\mathbf{F}^{\text{c}}$                           // (5.14)
9     return $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}}$
```

5.4 Fully Implicit Friction

Similar to before, we model friction according to Coulomb's friction law, which describes the conditions necessary for two solids to transition between sticking and sliding. This law states that the frictional force F^{fr} is (1) equal to μF^{n} during sliding, (2) is in the region of $[0, \mu F^{\text{n}})$ when sticking, and (3) is independent of the magnitude of velocity. Here, μ is the friction coefficient and F^{n} is the normal force experienced by the body.

Let us denote the following equivalencies for clarity: $\mathbf{F}^{\text{c}} \equiv -k\nabla_{\mathbf{x}} E_c$ and $\mathbf{J}^{\text{c}} \equiv -k\nabla^2_{\mathbf{x}} E_c$. Following this, for a contact pair $\mathbf{x}_{ij}$, we can obtain the normal force on the i-th and $i+1$-th nodes as $F^{\text{n}}_i = \|\mathbf{F}^{\text{c}}_i\|$ and $F^{\text{n}}_{i+1} = \|\mathbf{F}^{\text{c}}_{i+1}\|$, respectively. This in turn allows us to obtain the contact norm vector

$$\mathbf{n}_i = \frac{\mathbf{F}^{\text{c}}_i + \mathbf{F}^{\text{c}}_{i+1}}{\|\mathbf{F}^{\text{c}}_i + \mathbf{F}^{\text{c}}_{i+1}\|}. \tag{5.7}$$

The direction of friction is then the tangential relative velocity between edges i and j. To compute this, we must first compute the relative velocities of the edges at the point of

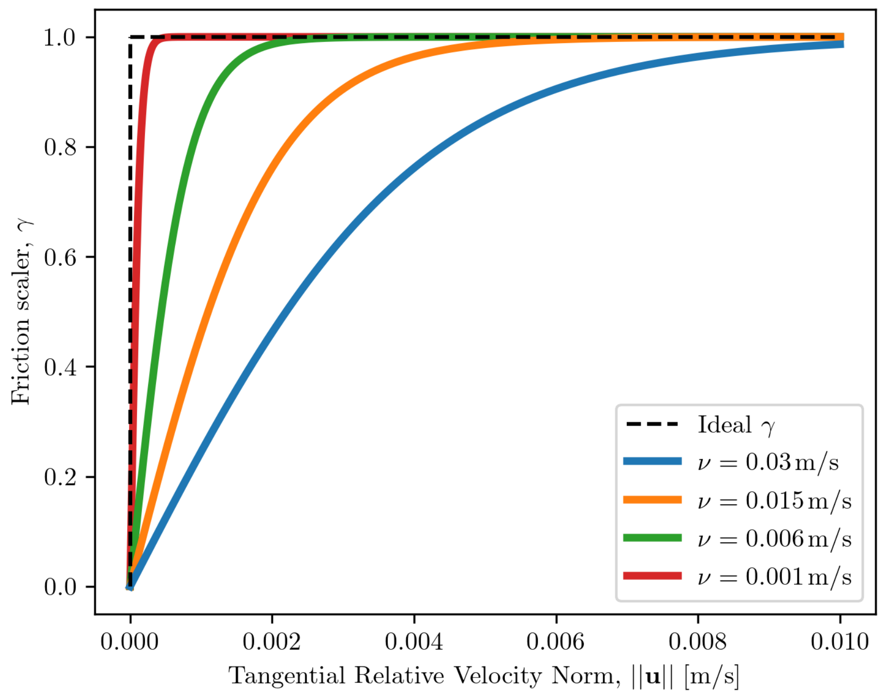


Figure 5.4: Plots for (5.10) with varying ν values. Note that some of the tolerances displayed are unrealistically large for clarity.

contact, which can be done using $\beta_i, \beta_j \in [0, 1]$ as shown below:

$$\begin{aligned} \mathbf{v}_i^{\mathrm{e}} &= (1 - \beta_i)\mathbf{v}_i + \beta_i \mathbf{v}_{i+1}, \\ \mathbf{v}_j^{\mathrm{e}} &= (1 - \beta_j)\mathbf{v}_j + \beta_j \mathbf{v}_{j+1}, \\ \mathbf{v}^{\mathrm{rel}} &= \mathbf{v}_i^{\mathrm{e}} - \mathbf{v}_j^{\mathrm{e}}, \end{aligned} \tag{5.8}$$

where $\mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{v}_j$, and $\mathbf{v}_{j+1}$ are the velocities of the i-th, $i+1$-th, j-th, and $j+1$-th nodes, respectively. The tangential relative velocity of edge i with respect to edge j can then be computed as

$$\mathbf{u} = \mathbf{v}^{\mathrm{rel}} - (\mathbf{v}^{\mathrm{rel}} \cdot \mathbf{n}_i)\mathbf{n}_i, \tag{5.9}$$

where $\hat{\mathbf{u}} = \mathbf{u}/||\mathbf{u}||$ is our friction direction.

Now, we must also make our contact model capable of simulating the transition between sticking and sliding. Coulomb's law tells us that $\|\mathbf{u}\| = 0$ during static friction and that $\|\mathbf{u}\| > 0$ for sliding friction. Sticking occurs up until the tangential force threshold μF^{n} is surpassed, after which sliding begins. This relation (similar to ideal contact energy) can also be described by a modified Heaviside step function. For the same reasons as before, we replace this step function for another smooth approximation described by

$$\gamma\left(\|\mathbf{u}\|, \nu\right) = \frac{2}{1 + e^{-K_2\|\mathbf{u}\|}} - 1, \tag{5.10}$$

where ν (m/s) is our desired slipping tolerance and $K_2(\nu) = 15/\nu$ is the stiffness parameter. As shown in Figure 5.4, $\gamma \in [0, 1]$ smoothly scales the friction force magnitude from zero to one as $\|\mathbf{u}\|$ increases from zero. The slipping tolerance describes the range of velocities $(0, \nu)$ for which a friction force $< \mu F^{\text{n}}$ is experienced. In other words, we consider velocities within this range to be "sticking".

Finally, the friction experienced by a node i for a contact pair $\mathbf{x}_{ij}$ can be described as

$$\mathbf{F}_i^{\text{fr}} = -\mu\gamma\hat{\mathbf{u}}^{\text{Trel}} F_i^{\text{n}}. \tag{5.11}$$

With friction fully defined, we can now formulate the friction Jacobian $\nabla_{\mathbf{x}}\mathbf{F}^{\text{fr}}$. Note that due to (5.8), our formulation depends on $\beta(\mathbf{x})$, which means that the gradient $\nabla_{\mathbf{x}}\beta$ is required. We can avoid this computation through the realization that the magnitudes of the contact forces $\mathbf{F}_i^{\text{c}}$ and $\mathbf{F}_{i+1}^{\text{c}}$ have an underlying linear relationship with β where

$$\begin{aligned} \mathbf{F}_i^{\text{c}} &= (1-\beta)(\mathbf{F}_i^{\text{c}} + \mathbf{F}_{i+1}^{\text{c}}), \\ \mathbf{F}_{i+1}^{\text{c}} &= \beta(\mathbf{F}_i^{\text{c}} + \mathbf{F}_{i+1}^{\text{c}}). \end{aligned} \tag{5.12}$$

Therefore, we can obtain β by simply solving

$$\beta = \frac{\|\mathbf{F}_{i+1}^{\text{c}}\|}{\|\mathbf{F}_i^{\text{c}} + \mathbf{F}_{i+1}^{\text{c}}\|}. \tag{5.13}$$

We can now treat β as a function of $\mathbf{F}^{\text{c}}$, resulting in a simplified chain ruling procedure. Let us denote (5.11) as the functional $f(\mathbf{x}, \mathbf{F}^{\text{c}}(\mathbf{x}))$. The friction Jacobian can then be computed through chain rule as

$$\nabla_{\mathbf{x}}\mathbf{F}^{\text{fr}} = \nabla_{\mathbf{x}} f + \nabla_{\mathbf{F}^{\text{c}}} f \nabla_{\mathbf{x}}\mathbf{F}^{\text{c}}. \tag{5.14}$$

This concludes our fully implicit friction scheme. Full psuedocode for the IMCv2 algorithm can be found in Alg. 3.

5.5 Algorithmic Components

In this section, we go over three key algorithmic components necessary for simulation: collision detection, adaptive contact stiffness, and the use of line search. The full pseudocode for the flagella simulation framework using IMCv2 can be seen in Alg. 5.

5.5.1 Collision Detection

For collision detection, we simply obtain the set of all edge combinations whose minimum distance is less than $C + \delta$, resulting in the contact set

$$\mathcal{C} = \{\mathbf{x}_{ij} \in \mathcal{X} \mid D_{ij} < C + \delta\}. \tag{5.15}$$

As this operation can be quite computationally expensive, we instead compute a candidate set $\hat{\mathcal{C}} = \{\mathbf{x}_{ij} \in \mathcal{X} \mid D_{ij} < C + \delta^{\text{col}}\}$ at the beginning of each time step where $\delta^{\text{col}} > \delta$ by a large enough margin. We then compute the actual contact set $\mathcal{C}$ from the candidate set $\hat{\mathcal{C}}$ at the start of each iteration, significantly reducing computational cost.

Note that if δ^{col} is not set large enough, certain edge combinations not belonging to the initial set $\hat{\mathcal{C}}$ may enter the contact zone or even penetrate by the end of the optimization. Our energy formulation in (5.2) is capable of dealing with this as minor penetrations do not lead to simulation failure and will be remedied in the next time step. This is in contrast to IPC, which may require a significantly larger δ^{col} and/or more robust collision checking (e.g., continuous collision detection) during each iteration of the optimization process.

5.5.2 Adaptive Contact Stiffness

As a penalty method, a contact stiffness k is used to scale the contact force and Jacobian. An appropriate value must be used to ensure that penetration (caused by too low of a value) or excessive hovering (caused by too large of a value) does not occur. First, let us denote the set of all i and j node indices of the contact set edge combinations $\mathbf{x}_{ij} \in \mathcal{C}$ as the set $\mathcal{C}_i$. Then, to generate an appropriate scaling for the contact stiffness, we compute the norm of the sum of forces (minus contact and friction) $\left|\left|\mathbf{F}_i^{\text{total}}\right|\right|$ experienced by each node in the contact set $\mathcal{C}$. We can denote these forces as the set

$$\mathcal{F} = \left\{ \ \left|\left|\mathbf{F}_i^{\text{total}}\right|\right| \ \mid i \in \mathcal{C}_i \ \right\}. \tag{5.16}$$

The maximum force magnitude of these forces can then be used to determine the contact stiffness

$$k = \max(\mathcal{F}) s, \tag{5.17}$$

where s is a constant scaling factor. In all our experiments, we set s to be 1×10^5. The intuition behind this contact stiffness formulation is to achieve non-penetration through force equilibrium. Furthermore, by using the maximum of $\mathcal{F}$, if non-penetration can be achieved for the edge with the largest value in $\mathcal{F}$, then this k value should be large enough to prevent penetration for all other contact pairs as well.

5.5.3 Line Search

Once the internal forces (e.g., bending force, twisting force, and stretching force), external forces (e.g., viscous dragging force and contact forces), and their respective Jacobians are computed, we can simply use Newton method to find the solution of the equations of motion. However, due to the high nonlinearity of the governing equations, convergence for Newton's method may suffer without a line search method. To rectify this, we perform

Algorithm 4: Line Search

```
Parameters: $\alpha_l, \alpha_u, m_1 = 0.1, m_2 = 0.9$          // Initial interval for $\alpha$
Input: $\mathbf{q}, \Delta\mathbf{q}$                                        // DOFs from DER
Output: $\alpha$                                                  // Newton search magnitude
Function LineSearch($\mathbf{q}, \Delta\mathbf{q}$):
    $n \leftarrow 0$
    $\alpha \leftarrow 1$
    success $\leftarrow$ False
    $\mathbf{d_0} \leftarrow \mathbf{F}^T(\partial\mathbf{F}/\partial\mathbf{q})\Delta\mathbf{q}$
    while success is False do
        if $\alpha m_2 \mathbf{d}_0 <= \frac{1}{2}\|\mathbf{F}(\mathbf{q} - \alpha\Delta\mathbf{q})\|^2 - \frac{1}{2}\|\mathbf{F}(\mathbf{q})\|^2 <= \alpha m_1 \mathbf{d}_0$ then
            success $\leftarrow$ True
        else if $\frac{1}{2}\|\mathbf{F}(\mathbf{q} - \alpha\Delta\mathbf{q})\|^2 - \frac{1}{2}\|\mathbf{F}(\mathbf{q})\|^2 < \alpha m_2 \mathbf{d}_0$ then
            $\alpha_l \leftarrow \alpha$
        else
            $\alpha_u \leftarrow \alpha$
        if $|\alpha_l - \alpha_r| <$ small value or $n >$ iterMax then
            success $\leftarrow$ True
        $\alpha \leftarrow 0.5(\alpha_l + \alpha_u)$
        $n \leftarrow n + 1$
    end
    return $\alpha$
```

Goldstein-Price line search in the Newton direction to ensure that the square of the Euclidean norm of total force $\|\mathbf{F}\|^2$ in (3.9) decreases.

We design an inner loop for each Newton iteration where we deploy the line search algorithm. This inner loop returns an optimal search step size α until $\|\mathbf{F}\|^2$ is smaller than a certain tolerance or until a maximum number of iterations is reached. As mentioned in Section 5.2, our energy formulation allows us to use a more aggressive line search strategy compared to Li et al. (2020a), resulting in larger search step sizes and faster convergence. Pseudocode for the line search method can be found in detail in Alg. 4.

5.6 Flagella Bundling Simulation Results

In this section, we showcase extensive quantitative and qualitative results for IMCv2. First, we discuss all our simulation parameters. We then conduct a detailed comparison between

Algorithm 5: Flagella Simulation Framework

```
Params: $\delta, \delta^{\text{col}}, \nu$, tolerance
Require: boundary conditions $\leftarrow$ free
Input: $\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$
Output: $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$
1  Function FlagellaSim($\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$):
2      Guess $\mathbf{q}^{(0)} \leftarrow \mathbf{q}(t_i)$
3      $n \leftarrow 0,\ \epsilon \leftarrow \infty$
4      $\mathbf{F}^{\text{hydro}} \leftarrow$ RSS($\cdot$)                                  // (Cortez, 2018)
5      $\mathbf{F}^{\text{total}} \leftarrow \mathbf{0}^{\text{DOF}}$
6      $\mathbf{J}^{\text{total}} \leftarrow \mathbf{0}^{\text{DOF}\times\text{DOF}}$
7      while $\epsilon >$ tolerance do
8          $\mathbf{F}^{\text{int}} \leftarrow$ genForces($\cdot$)                                  // (3.8)
9          $\mathbf{J}^{\text{int}} \leftarrow$ genJacobian($\cdot$)                     // $\frac{\partial^2(E_s+E_t+E_b)}{\partial q_i \partial q_j}$
10         if $n == 0$ then                                  // run only on first iter
11             $\hat{\mathcal{C}} \leftarrow$ constructCandidateSet($\mathbf{x}, \delta^{\text{col}}$)              // Section 5.5.1
12             $k \leftarrow$ updateConStiffness($\hat{\mathcal{C}}, \mathbf{F}^{\text{int}}$)              // Section 5.5.2
13         $\mathcal{C} \leftarrow$ collisionDetection($\hat{\mathcal{C}}, \delta$)
14         for $\mathbf{x}, \mathbf{x}_0 \in \mathcal{C}$ do
15             $\mathbf{F}^{\text{c}}, \mathbf{J}^{\text{c}}, \mathbf{F}^{\text{fr}}, \mathbf{J}^{\text{fr}} \leftarrow$ IMCv2($\mathbf{x}^{(n)}, \mathbf{x}_0, k, \delta, \nu$)           // Alg. 3
16             $\mathbf{F}^{\text{total}} \leftarrow \mathbf{F}^{\text{total}} + \mathbf{F}^{\text{c}} + \mathbf{F}^{\text{fr}}$
17             $\mathbf{J}^{\text{total}} \leftarrow \mathbf{J}^{\text{total}} + \mathbf{J}^{\text{c}} + \mathbf{J}^{\text{fr}}$
18         end
19         $\mathbf{F}^{\text{total}} \leftarrow \mathbf{F}^{\text{total}} + \mathbf{F}^{\text{int}} + \mathbf{F}^{\text{hydro}}$
20         $\mathbf{J}^{\text{total}} \leftarrow \mathbf{J}^{\text{total}} + \mathbf{J}^{\text{int}}$
21         $\mathbf{F}^{\text{free}} \leftarrow \mathbf{F}^{\text{total}}$(free)        // Downsize to only include free DOFs
22         $\mathbf{J}^{\text{free}} \leftarrow \mathbf{J}^{\text{total}}$(free, free)
23         $\Delta\mathbf{q}^{\text{free}} \leftarrow \left(\mathbf{J}^{\text{free}}\right)^{-1} \mathbf{F}^{\text{free}}$                // Solve $\mathbf{J}^{\text{free}}\Delta\mathbf{q}^{\text{free}} = \mathbf{F}^{\text{free}}$
24         $\alpha \leftarrow$ LineSearch($\mathbf{q}^{\text{free}}, \Delta\mathbf{q}^{\text{free}}$)                          // Alg. 4
25         $\mathbf{q}^{(n+1)}$(free) $\leftarrow \mathbf{q}^{(n)}$(free) $- \alpha\Delta\mathbf{q}_{\text{free}}$
26         $\epsilon \leftarrow \left\|\mathbf{F}^{\text{free}}\right\|$                                  // update error
27         $n \leftarrow n + 1$
28     end
29     $\mathbf{q}(t_{i+1}) \leftarrow \mathbf{q}^{(n)}$
30     $\dot{\mathbf{q}}(t_{i+1}) \leftarrow (\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i))/\Delta t$
31     return $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$
```

IMCv2 and the state-of-the-art contact handling method: Incremental Potential Contact (IPC) (Li et al., 2020a). Afterwards, we showcase comprehensive results concerning friction and display IMCv2's ability to simulate the sticking sliding transition.

5.6.1 Parameters and Setup

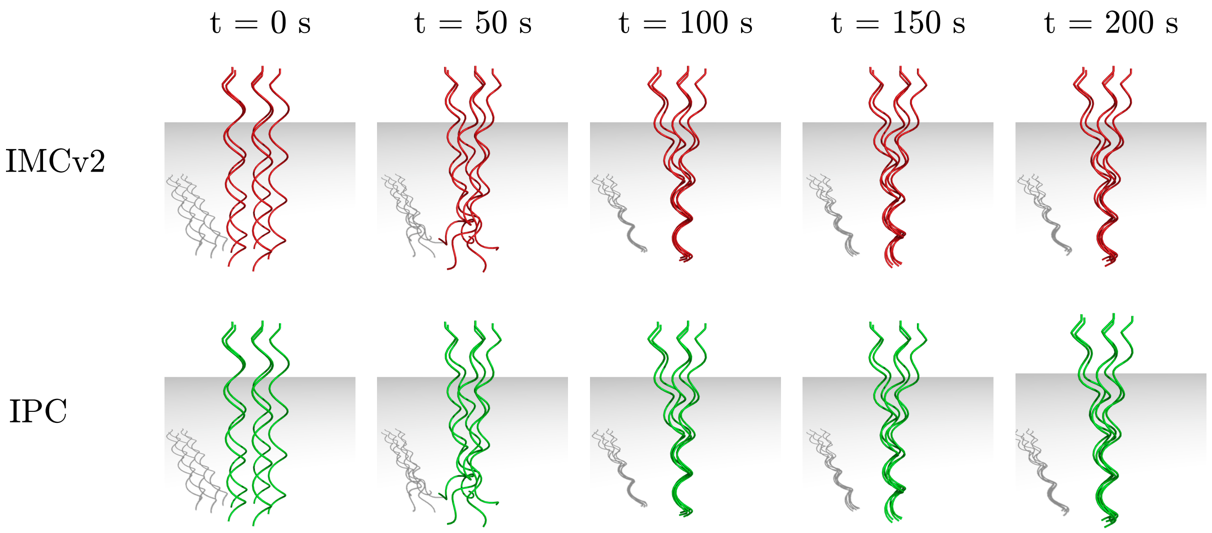
In the simulation, we design the flagella as right-handed helical rods manufactured with linear elastic material. We set the material properties as follows: Young's modulus was set to $E = 3.00\,\mathrm{MPa}$; Poisson's ratio was set to 0.5; density of the rod was set to $\rho = 1000\,\mathrm{kg/m^3}$; the cross-sectional radius was set to $h = 1\,\mathrm{mm}$, and the fluid viscosity was set to $0.1\,\mathrm{Pa{\cdot}s}$. Here, a Poisson's ratio of 0.5 was chosen to enforce the flagella to be an incompressible material. The topologies of the flagella are helices with helical radius $a = 0.01\,\mathrm{m}$, helical pitch $\lambda = 0.05\,\mathrm{m}$, and axial length $z_0 = 0.2\,\mathrm{m}$. These parameters were chosen as they best mimic the geometries of biological flagella found in nature (Jawed et al., 2015c; Jawed and Reis, 2017; Rodenborn et al., 2013; Huang and Jawed, 2021).

We explore the bundling phenomena with M flagella ($M = [2, 3, 5, 10]$) where the rotating ends of each flagella is fixed along the z-axis as shown in Figure 5.2. These rotating ends are treated as boundary conditions and are spaced out equidistantly so as to form a regular polygon with M angles with side length $\Delta L = 0.03\,\mathrm{m}$ as shown in Figure 5.5b. We set the rotation speed of the flagella ends to $\omega = 15\,\mathrm{rad/s}$ which keeps the Reynolds number in our numerical simulation to be always smaller than 4×10^{-2}, thus satisfying the Stokes flow.

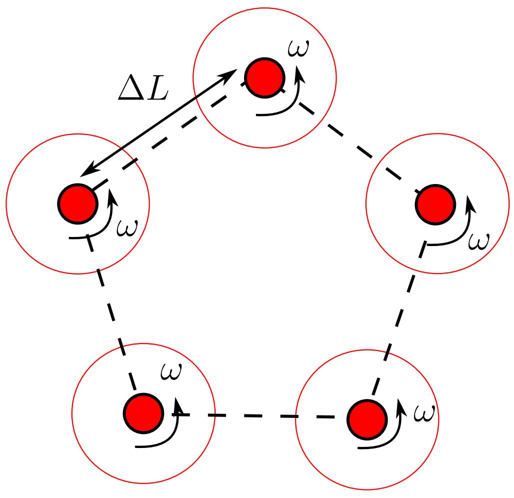
Finally, we discretize each flagella into 68 nodes for a total of 67 edges. We found this discretization to have the best trade-off between computational efficiency and accuracy. Furthermore, we set the time step size to $\Delta t = 1\,\mathrm{ms}$. As the forces generated from our fluid model are handled explicitly, we found $1\,\mathrm{ms}$ to be the largest stable time step size before convergence performance became hampered. A distance tolerance of $\delta = 1 \times 10^{-5}\,\mathrm{m}$ was used for all simulations.

5.6.2 Comparison between IMCv2 and IPC

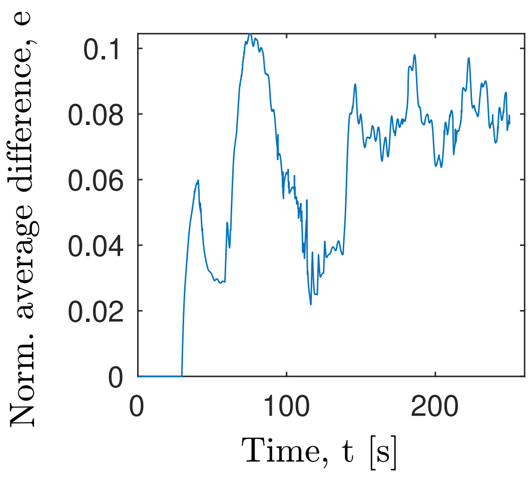
Both IMCv2 and IPC were used to simulate 250 seconds of rotation for scenarios with 2, 3, 5, and 10 flagella, as shown in Figure 5.1. As the friction coefficient between structures is usually trivial in viscous fluids, we consider purely contact without friction



(a) Flagella Bundling Comparison with IPC



(b) Boundary Conditions



(c) Average Configuration Difference

Figure 5.5: (a) Rendered snapshots for $M = 5$ flagella simulated by IMCv2 and IPC. We can observe that there is great qualitative agreement between both methods at the shown time steps. (b) A top down visualization of boundary conditions applied to the highest nodes (filled in red circles) of each flagella as well as the angular rotation ω applied to them. The larger hollow red circles represent the rest of the helical flagella. (c) The norm of the average difference in the nodal positions for the flagella simulated by IMCv2 and IPC with respect to time.

($\mu = 0$). First, a side-by-side visual comparison for $M = 5$ is shown for IMCv2 and IPC in Figure 5.5a. For various time steps, we can see that the configurations of the flagella are near identical, indicating that both methods have comparable performance. To further study this similarity, we define normalized average difference $\bar{e}$ to measure the difference

Table 5.1: IMCv2 vs. IPC (Li et al., 2020a) run time data. Simulations are run for a total of 250 seconds with a time step size of $\Delta t = 1\,\text{ms}$ and a rotation speed of $\omega = 15\,\text{rad/s}$. The contact model used can be seen in the far left column. Next to this, M indicates the number of flagella. AIPTS stands for average iterations per time step. ATPTS stands for average time per time step. Total Iters indicates the total number of Newton's iterations that were necessary to complete the simulation. The Total Run Time is the total computational time to completion. Finally, RTI stands for run time improvement and is the ratio of improvement between IMCv2's and IPC's total run time.

Model	M	AIPTS	ATPTS [ms]	Total Iters	Total Run Time [hr]	RTI
IMCv2	2	3.00	10.2	6.01×10^5	0.57	1.82
	3	3.01	21.3	6.04×10^5	1.19	1.82
	5	3.02	67.5	5.39×10^5	3.34	1.40
	10	3.12	389.4	6.56×10^5	22.77	1.22
IPC	2	4.00	18.75	7.98×10^5	1.04	N/A
	3	4.00	39.5	7.93×10^5	2.17	N/A
	5	4.01	95.3	7.09×10^5	4.68	N/A
	10	4.02	477.47	8.45×10^5	27.88	N/A

in flagella nodal configurations between IMCv2 and IPC:

$$\bar{e} = \frac{1}{MNh} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left\| \mathbf{x}_j^{i,\text{IMCv2}} - \mathbf{x}_j^{i,\text{IPC}} \right\|. \tag{5.18}$$

The relationship between normalized average difference $\bar{e}$ and time t is shown in Figure 5.5c. Here, we can find that the difference between the configurations is quite minimal, further cementing the notion that IMCv2 has comparable performance to IPC despite the loss of non-penetration guarantee.

Where IMCv2 starts to improve upon IPC is in terms of computational efficiency. Detailed metrics for all runs can be seen in Table 5.1 which showcase the average iteration per time step (AIPTS), average time per time step (ATPTS), total iterations, and total run time. All metrics were recorded using time steps with at least one contact. Here, we can see that IMCv2 was able to converge with less average iterations than IPC for all flagella cases resulting in significant reductions in total run time. These run time improvements are most drastic for $M = 2$ and $M = 3$ and start to decrease as M increases

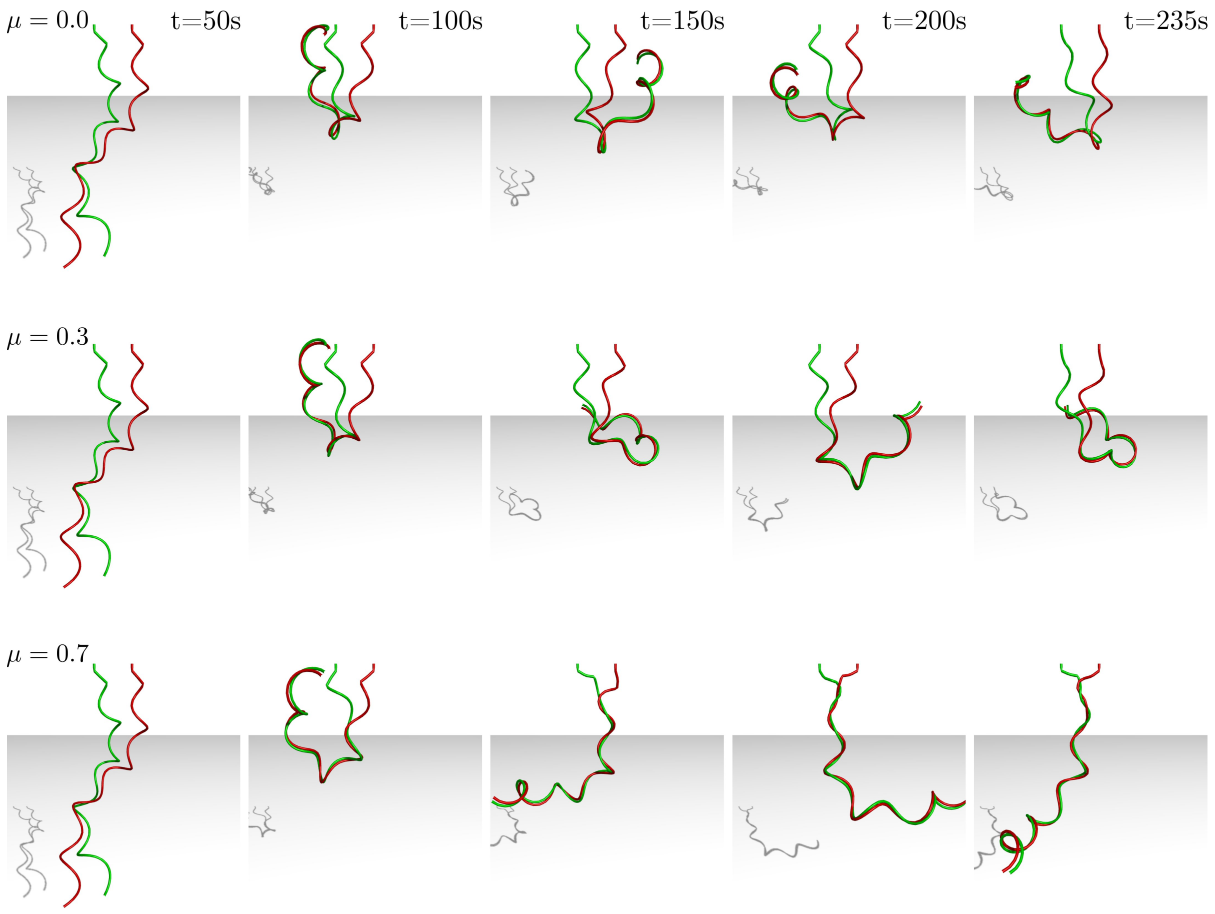


Figure 5.6: Rendered snapshots for $M = 2$ with varying friction coefficients. Each column indicates a moment in time as indicated by the time stamp in the top row. The first row shows the frictionless case $\mu = 0$ as a baseline. The second row has $\mu = 0.3$ where minor sticking can be observed as the point at where the flagella no longer contact is higher than the frictionless case. Still, $\mu = 0.3$ still has plenty of slipping allowing the flagella to not become coiled. As we increase μ to 0.7 in the third row, we can see the amount of sticking increase, ultimately resulting in the flagella becoming completing coiled.

further as the RSS force computation starts to become a bottleneck. Regardless, a clear monotonic decrease can still be seen.

5.6.3 Friction Example

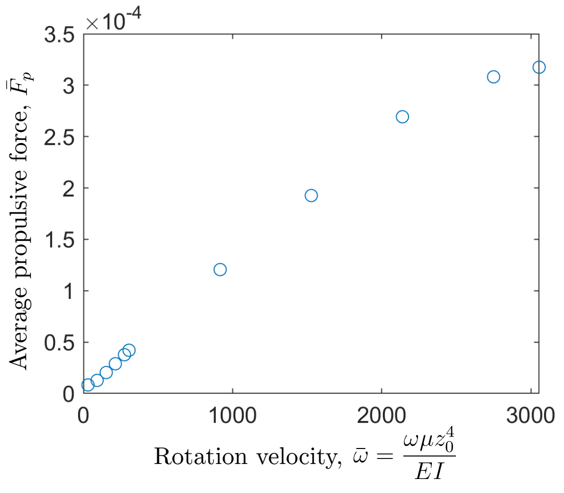
Although friction is usually negligible in a viscous fluid medium, influence of friction on flagella bundling is still intriguing since the effect of friction can become significant as the environment changes (e.g., granular medium). We assume an imaginary viscous

Table 5.2: IMCv2 friction results for varying friction coefficients. AIPTS stands for average iterations per time step. Total Iterations indicate the total number of Newton's method iterations that were necessary to complete the simulation. Sim End indicates the total *simulated* time. All simulations were set to run for 250 seconds. As can be seen, simulations with $\mu \geq 0.7$ end earlier due to excessive tangling of the flagella.

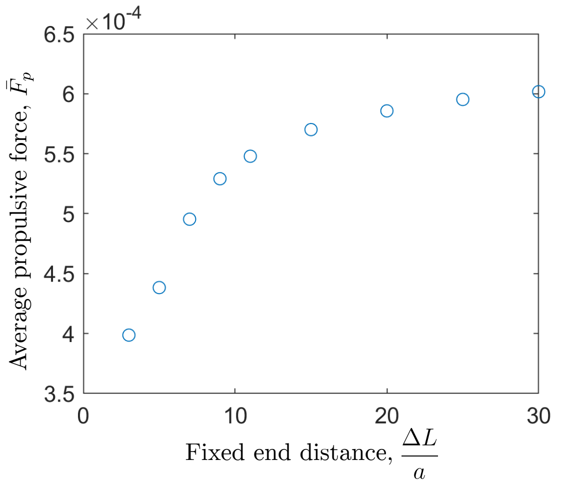
μ	AIPTS	Total Iterations	Sim End [sim s]
0.1	3.01	6.02×10^5	250
0.2	3.01	6.04×10^5	250
0.3	3.61	7.25×10^5	250
0.4	4.89	9.83×10^5	250
0.5	6.67	1.34×10^6	250
0.6	8.71	1.76×10^6	250
0.7	14.47	2.72×10^6	235.89
0.8	14.16	1.89×10^6	180.98
0.9	11.1	1.05×10^6	142.72
1.0	11.65	1.02×10^6	135.32

environment where the friction coefficient between structures is non-negligible. We present simulation data for two flagella ($M = 2$) with friction coefficients $\mu = [0.1, 0.2, ..., 1.0]$. For all simulations, a slipping tolerance of $\nu = 1 \times 10^{-4}$ m/s was used. All other parameters are kept the same as before.

We first showcase the sticking slipping phenomena with snapshots for $\mu = [0, 0.3, 0.7]$ in Figure 5.6. Intuitively, as μ increases, we also see the amount of sticking increase as well. Convergence results for all friction examples can be seen in Table 5.2 where average iterations per time step and simulation length are reported. Here, we notice two trends. First, for $\mu \geq 0.7$, the time at which the simulation ends starts to decrease from 250 seconds. This is because $\mu = 0.7$ is the point at which the flagella become completely tangled as shown in the bottom right frame of Figure 5.6. As μ increases past 0.7, the tangling happens earlier and earlier as shown. Furthermore, we observe that the number of average iterations starts to increase as μ increases. This is in line with our expectations as larger μ values result in greater sticking.



(a) $\bar{\omega}$ vs. Average $\bar{F}_p$



(b) Fixed end distance vs. Average $\bar{F}_p$

Figure 5.7: Simulation results showcasing the relationship between average propulsive force $\bar{F}_p$ and (a) rotation velocity $\bar{\omega}$ and (b) fixed end distance $\Delta L/a$ over 597 revolutions.

5.6.4 Parametric Study for $\bar{\omega}$, $\Delta L/a$, M, and, μ

In this section, we perform a parametric study of flagella bundling using our simulation framework. Motivated by scaling analysis performed in Kim et al. (2003), we choose four main dimensionless groups to examine: angular velocity $\bar{\omega} = \omega\nu z_0^4/EI$, fixed end distance $\Delta L/a$ (distance between neighboring flagella), rod number M, and friction coefficient μ. We refer to all quantities with a $^-$ as non-dimensionalized. Note that ω is the rotation velocity of the fixed end; ν is the viscosity of the fluid media; z_0 is the axial length of the flagella; EI is the bending stiffness; ΔL is the distance between the neighboring fixed ends, and a is the radius of helical geometry for one flagellum.

To evaluate the effects of our dimensionless groups, we define average propulsive force $\bar{F}_p$. When flagella bundle, propulsive forces are generated on each fixed end. We compute the average propulsive force on each fixed end during a fixed time period (defined as number of revolutions at constant ω) and average this among the M flagella to obtain F_p. With this, we can obtain

$$\bar{F}_p = \frac{F_p h^2}{EI}. \tag{5.19}$$

In Figures 5.7, 5.8, and 5.9, we showcase the influence of each dimensionless group.

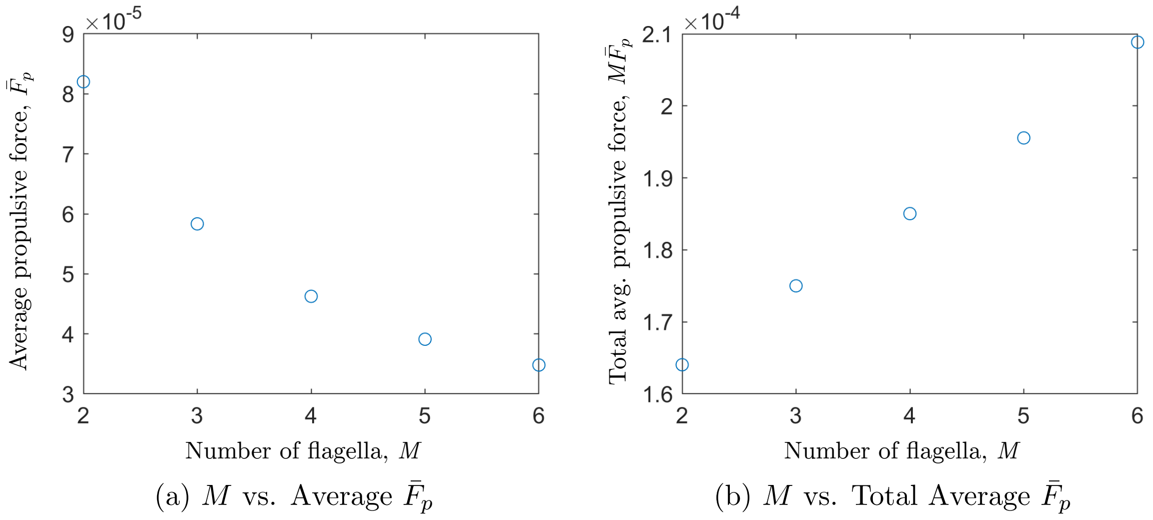


Figure 5.8: Simulation results showcasing the relationship between (a) average propulsive force $\bar{F}_p$ and number of flagella M and (b) the total average propulsive force $M\bar{F}_p$ against M over 597 revolutions.

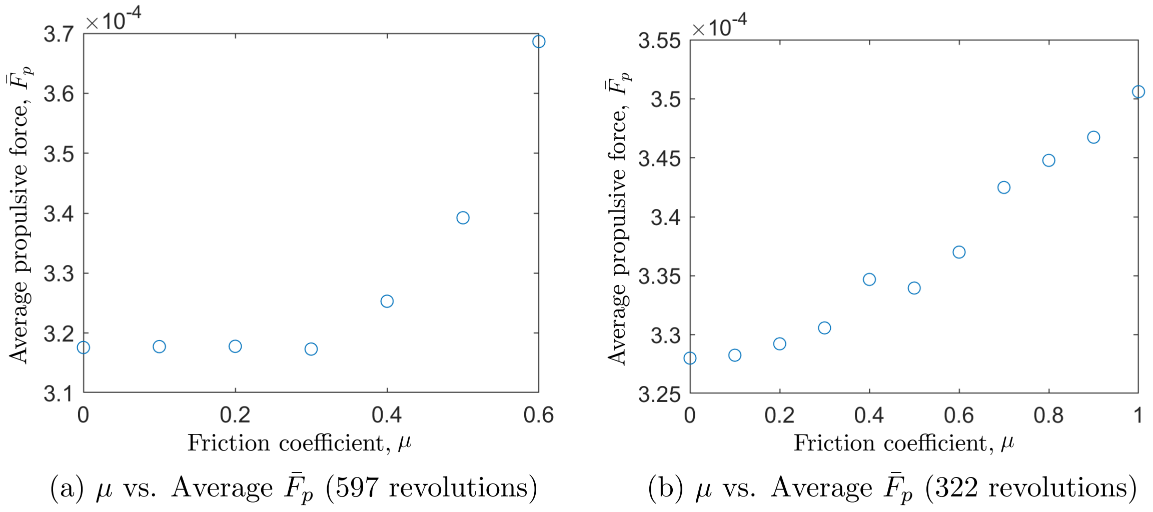


Figure 5.9: Simulation results showcasing the relationship between average propulsive force $\bar{F}_p$ and (a) friction coefficient μ over 597 revolutions. Note that (a) only includes $\mu \in [0, 0.6]$ as simulations with $\mu > 0.6$ ended before the total 597 revolutions. Therefore, we also plot (b) the average propulsive force $\bar{F}_p$ against the full friction coefficient range $\mu \in [0, 1.0]$ over 322 revolutions (the earliest exit sim time). We note that the abnormal change between $\mu = 0.4$ and $\mu = 0.5$ for plot (b) can be attributed to insufficient data as 322 revolutions is nowhere near the simulation end of 597 revolutions.

All simulations used semicoiled helices (pitch / radius $\sim$ 5) as the flagella geometry. For all simulations not studying the effects of friction coefficient μ, we consider frictionless contact ($\mu = 0$). In addition, simulations not studying number of flagella M used $M = 2$

while those not studying fixed end distance used $\Delta L/a = 3$. Figure 5.7a studies the influence of rotation velocity with the aforementioned parameters. In Figure 5.7b, the influence of fixed end distance is studied with normalized rotation velocity $\bar{\omega} = 3056$. Next, we look into the influence of flagella number in Figure 5.8 where we set normalized rotation velocity $\bar{\omega} = 611$ (to minimize inertial effects). Finally, in Figure 5.9, we study the influence of friction coefficient with rotation velocity $\bar{\omega} = 15280$ since we want to observe contact influence under extreme bundling.

Overall, we conclude the following preliminary results:

- Higher normalized rotation velocities result in higher propulsive forces.
- Larger normalized fixed end distances result in higher propulsive forces, but the influence of fixed end distance taper off as the distance grows large enough.
- More flagella result in larger total average propulsive forces but lower per flagella average propulsive forces.
- The propulsive force is nearly identical for $\mu \in [0, 0.3]$ (low friction cases) and then starts to exponentially increase as μ increases past 0.3. We observe a near monotonic increase in $\bar{F}_p$ as μ increases, barring one abnormality explained in Figure 5.9.

We provide additional insights into the frictional data in Figure 5.9. First, as μ grows, the sticking surface between flagella grows, eventually resulting in "one" flagellum. This essentially results in a single flagellum with a higher effective stiffness, which translates to higher thrust forces as a single flagellum has been shown to have higher propulsive efficiency compared to multiple flagella with equal cumulative volume (Riley et al., 2018).

Overall, we showcase interesting preliminary results for flagella bundling provided entirely by our simulation framework. We emphasize here that presented results have not been validated with experiments. However, the results are quite intuitive and in particular, $\bar{\omega}$ and $\Delta L/a$ have similar tendencies with prior works (Kim et al., 2003).

5.7 Overhand Knot Tying Validation

In this section, we provide validation for our generated frictional contact forces against theory concerning the contact scenario of knot tying. We first showcase validation against the theoretical relation for tightening trefoil knots formulated in Audoly et al. (2007), similar to what was done in Section 4.7.1. In addition to this, we also validate our contact model against the theoretical relation for tightening knots of varying unknotting numbers formulated in Jawed et al. (2015a).

5.7.1 Tightening a Trefoil Knot

Recall that Audoly et al. (2007) formulated the theoretical relationship (4.20):

$$\frac{Fh^2}{EI} = \frac{\epsilon^4}{2} + \mu\sigma\epsilon^3,$$

where F is the traction force induced by friction; h is the rod radius; EI is the bending stiffness; μ is the coefficient of friction; σ is a constant (0.492 for trefoil knots), and $\epsilon = \sqrt{h/R}$ where R is the knot loop radius.

We use IMCv2 to run a simulation of tightening a trefoil knot and record the traction forces F for the parameters $h = 0.0016\,\text{m}$, $E = 1.8\text{e}5\,\text{Pa}$ ($EI = E\pi h^4/4$), and $\mu = 0.1$. We non-dimensionalize traction force as $\bar{F} = Fh^2/EI$ and plot this against ϵ. As shown in Figure 5.10, we observe excellent agreement between simulation results and the plotted theory.

5.7.2 Tightening Knots of Various Unknotting Numbers

In addition to the validation of Audoly et al. (2007), we also provide a more rigorous validation against theory for overhand knots of various unknotting numbers (Jawed et al.,

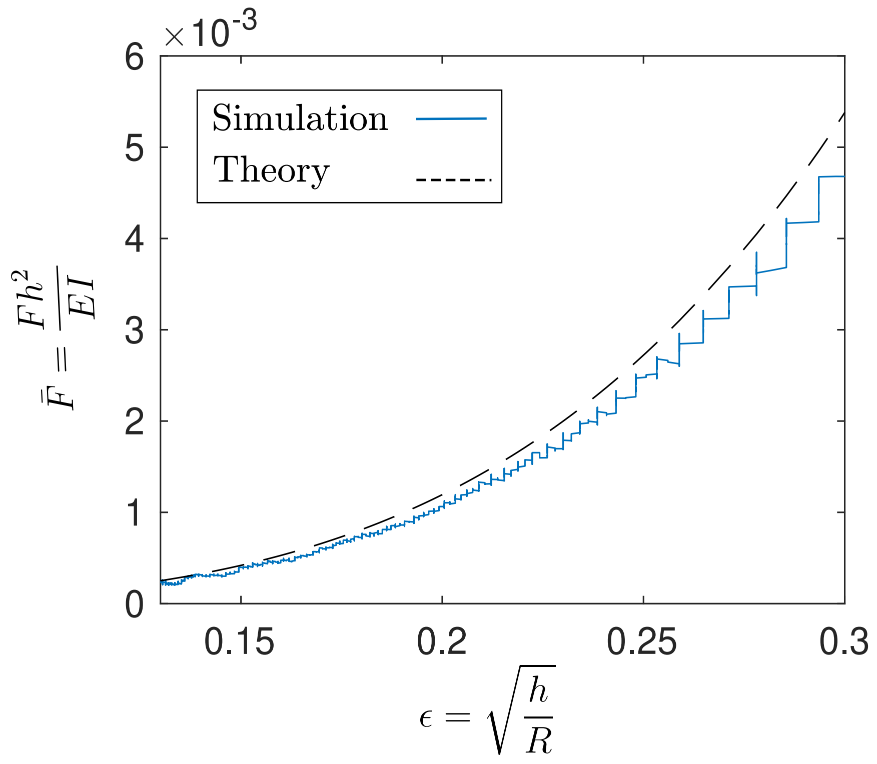


Figure 5.10: Comparison of non-dimensionalized traction force $\bar{F}$ between IMCv2 simulation results and theory for tightening trefoil knots (Audoly et al., 2007). Simulation results are colored while theory (4.20) is plotted as a dash line.

2015a). In this paper, Jawed et al. (2015a) proposed the relation

$$n^2 \frac{h}{e} = \frac{1}{8\sqrt{3}\pi^2} g\left(\left[\frac{96\sqrt{3}\pi^2}{\mu} \cdot \frac{n^2 F h^2}{EI}\right]^{\frac{1}{3}}\right), \tag{5.20}$$

where n is the unknotting number; e is the end-to-end shortening, and $g(\cdot)$ is a known nonlinear equation detailed in the paper.

We use IMCv2 to run simulations of tightening overhand knots for $n \in [1, 4]$ and $\mu = 0.07$ (all other rod parameters are the same as in Section 5.7.1. We plot the recorded non-dimensionalized traction force $n^2\bar{F}$ against the non-dimensionalized end-to-end shortening $\bar{e} = n^2 h/e$. Once again, we observe excellent agreement between simulation results and the plotted theory as shown in Figure 5.11, thus further cementing the physical accuracy of IMCv2's frictional contact forces.

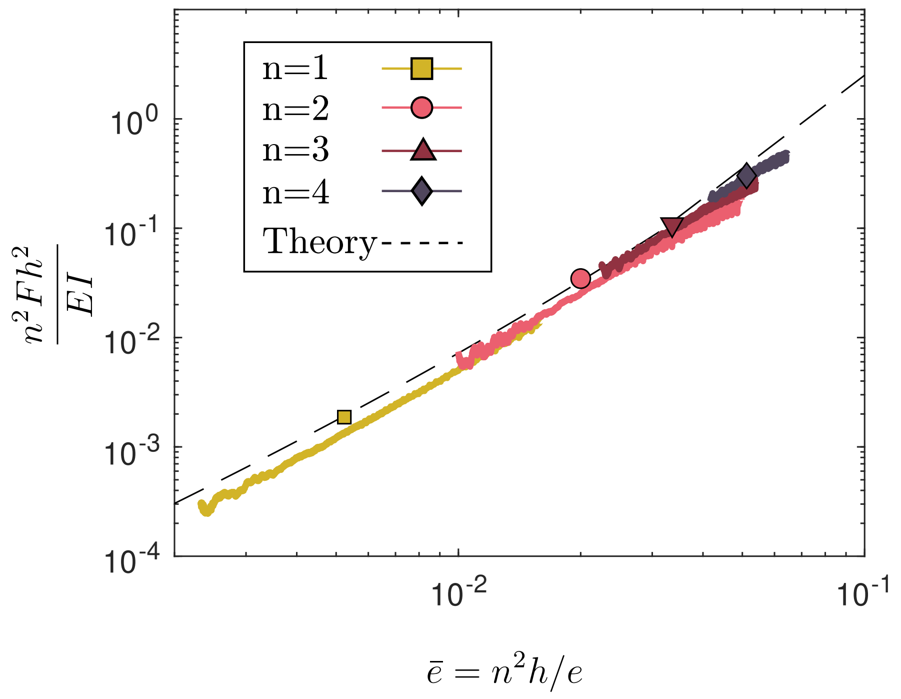


Figure 5.11: Comparison of non-dimensionalized traction force $\bar{F}$ between IMCv2 simulation results and theory for tightening overhand knots of various unknotting numbers n (Jawed et al., 2015a). Simulation results are colored while theory (5.20) is plotted as a dash line.

5.8 Granny vs. Reef Knot Validation

In this section, we study granny and reef (square) knots, as shown in Figure 5.13a. The reef knot is known famously as the one used for tying shoelaces. The granny knot differs from the reef knot by a single over/under crossing order as shown by the circled regions in Figure 5.13a. Despite the stark similarity in structure, the reef knot possesses significantly higher "knot strength", i.e. the knot stays fastened from self-friction when the ends are pulled. While reef knots tend to stay fastened until breaking when pulled on, granny knots will simply slip and unravel. We propose to study and analyze this mechanical phenomenon by utilizing physically accurate simulation frameworks.

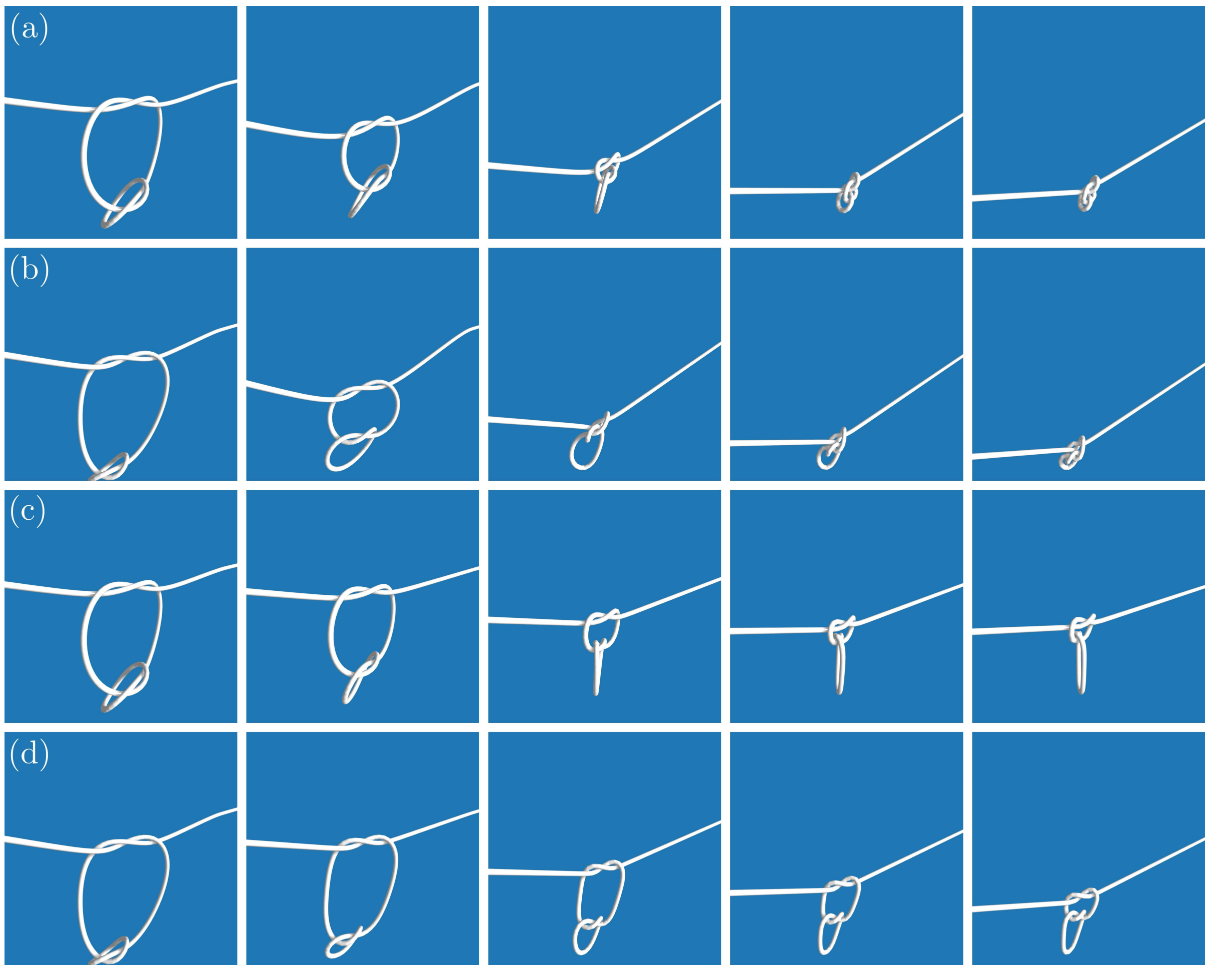
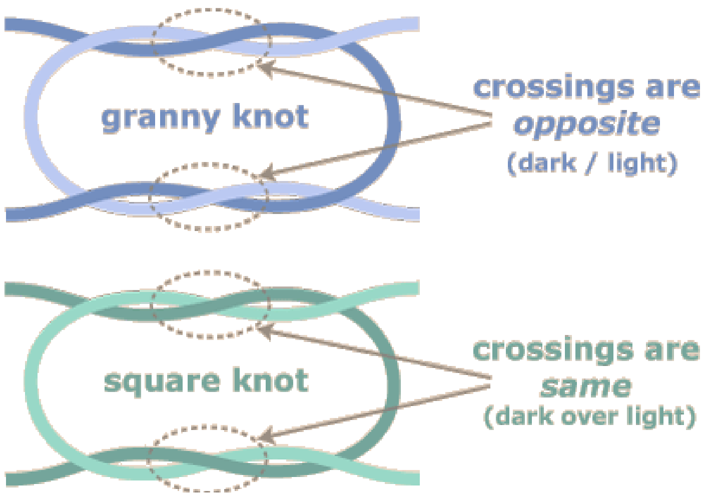


Figure 5.12: Snapshots for (a) granny knot with $\mu = 0.1$, (b) reef knot with $\mu = 0.1$, (c) granny knot with $\mu = 0.5$, and (d) reef knot with $\mu = 0.5$. Each row indicates the type of knot and μ. Each column indicates a snapshot after pulling for 0, 2, 7.5, 12, and 16 seconds from left to right.

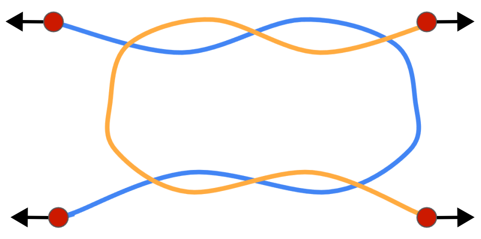
5.8.1 Boundary Conditions and Setup

Once the initial geometric configuration of a reef / granny knot is achieved, the boundary conditions for tying are relatively simple. First, boundary conditions are set on the ends at which pulling will occur, which constitutes the first and last edges of the knot. These ends will then be pulled in opposite directions and their experienced forces will be recorded as the traction forces F.

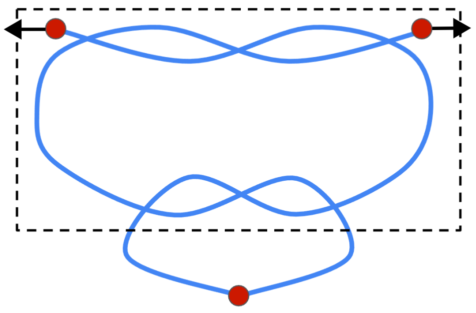
In addition to the pull ends, we apply a third static boundary condition to the bottom of the initial overhand knot, as shown in Figure 5.13c. Interestingly, the recorded traction



(a) Granny vs. Reef (Square) Knot Crossing Difference



(b) Boundary Conditions for Two Rods



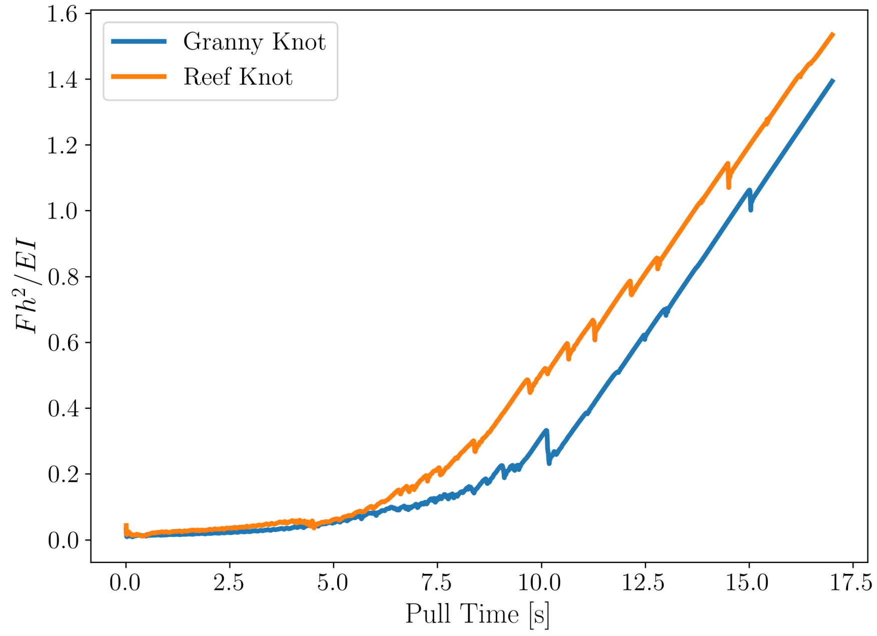
(c) Boundary Conditions for One Rod

Figure 5.13: (a) Visualization of granny and reef (square) knots. The difference between them is simply the overlay ordering of the circled crossings. (b) A traditional reef / granny knot shown tied using two separate elastic rods. (c) A reef / granny knot tied using a single elastic rod. This can be achieved by tying an initial overhand knot. Note that the portion of the knot in the boxed region is a reef / granny like the one in (b). Boundary conditions for the tying sequence are shown as red circles.

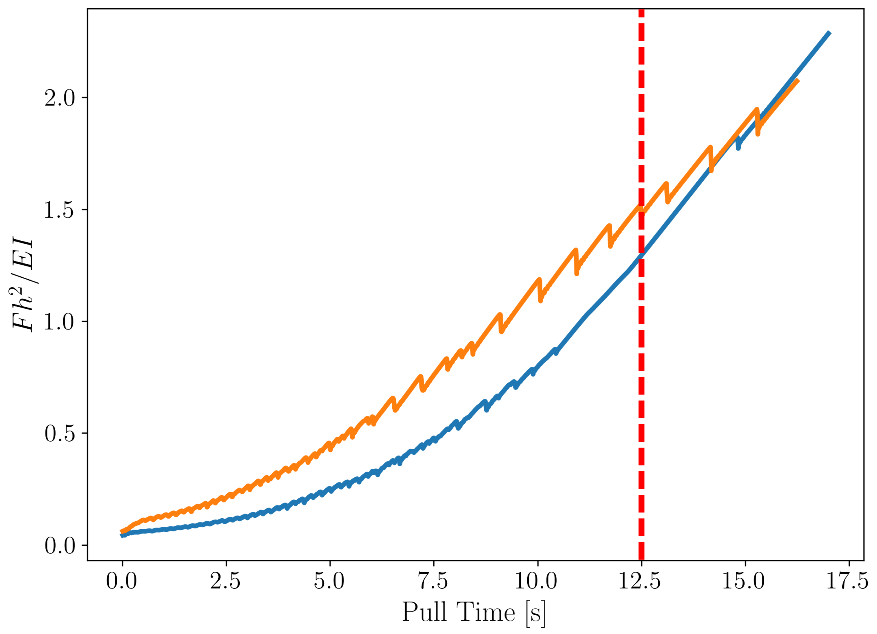
forces for both granny and reef knots without this final boundary condition are equivalent. We conclude that this is a result of the knot degenerating to a simple overhand knot case as the initial bottom overhand unravels on its own, resulting in the topology of the granny / reef knot being lost. Therefore, this static boundary condition is crucial to maintaining our desired knot topology.

5.8.2 Knot Strength Evaluation

We evaluate the knot strength as the normalized traction force Fh^2/EI experienced when pulling a granny and reef knot taut, where h is the rod radius and EI is the bending stiffness. We hypothesize that the knot strength of reef knots should be monotonically higher than granny knots for all stages of tightening unless both are taut. To test this theory, we conduct two sets of tightening simulations for both granny and reef knots. The



(a) Pull Forces at $\mu = 0.1$



(b) Pull Forces at $\mu = 0.5$

Figure 5.14: Normalized traction force comparison between granny and reef knots for (a) $\mu = 0.1$ and (b) $\mu = 0.5$. The horizontal red line in (b) refers to the moment at which the knots start to become taut.

first set of experiments involves light friction, $\mu = 0.1$, while the second pair involves high friction, $\mu = 0.5$. The mutual parameters that were used were Young's Modulus $E = 0.18\,\mathrm{MPa}$, rod radius $h = 2\,\mathrm{mm}$, rod length $L = 1\,\mathrm{m}$, density $\rho = 1180\,\mathrm{kg/m^3}$, pull speed (both ends) $u = 10\,\mathrm{cm/s}$, number of nodes $N = 301$, and time step $\Delta t = 2.5\,\mathrm{ms}$.

The traction forces for $\mu = 0.1$ can be seen in Figure 5.14a as a function of pull time (both knots have identical end-to-end shortenings at each pull time). Furthermore, the tightening sequence can be seen visualized in Figures 5.12(a-b). Here, we see that the reef knot's normalized traction force is monotonically higher than the granny knot at all stages of the tying process aside from the first five seconds. The values being the same in the first five seconds is caused by negligible friction forces as both knots slip considerably. Once the knot loop becomes tightened, the effects of friction become more prevalent ($t > 5\,\mathrm{s}$) and the traction forces start to diverge.

In comparison, for $\mu = 0.5$, the normalized traction forces diverge immediately once pulling occurs as the friction forces are non-negligible from the start (Figure 5.14b). We also note that due to the high friction, the granny and reef knots become taut much sooner as sticking becomes much more prevalent than sliding (Figure 5.12(c-d)). This occurs roughly at $t \approx 12.5\,\mathrm{s}$ as illustrated by the vertical dashed red line in Figure 5.14b. The early closing of the knot loop is also a result of the traction forces for $\mu = 0.5$ becoming equivalent with enough pulling. Overall, results for both $\mu = 0.1$ and $\mu = 0.5$ excellently match the common notion that reef knots possess higher knot strength than granny knots.

5.8.3 Parametric Study for μ

A parametric study is conducted comparing the effects of μ on the normalized traction forces. To compare the forces between different μ, the average of the recorded normalized traction forces was computed. A total pulling time of 13 seconds was used and μ was varied between values of $[0.1, 0.2, 0.3, 0.4, 0.5]$. All the rest of the parameters used were the same as the ones listed in Section 5.8.2.

Results for both the granny and reef knots can be seen in Figure 5.15. As expected,

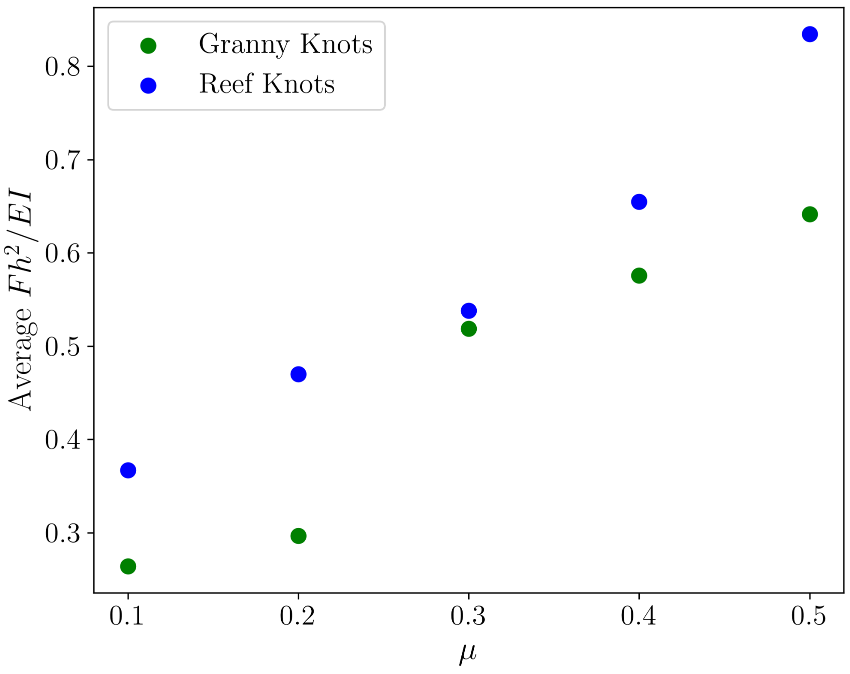


Figure 5.15: Study comparing the average normalized traction forces for friction coefficients of $\mu \in [0.1, 0.2, 0.3, 0.4, 0.5]$. Note that reef knots posses a higher average traction force regardless of μ.

we can see that reef knots possess a higher average normalized traction force than granny knots for all values of μ. Furthermore, the traction forces for both knots monotonically increase as μ increases. This matches our intuition as increasing μ increases the magnitude of the friction forces themselves.

5.8.4 Parametric Study for E

A parametric study comparing the effects of E on the average normalized traction force is also conducted. A total pulling time of 6.4 seconds was used and E was varied between values of $[1.8e4, 1.8e5, 1.8e6, 1.8e7]$ Pa. All the rest of the parameters used were the same as the ones listed in Section 5.8.2 aside from a lower time step of $\Delta t = 1\,\text{ms}$.

Results for both the granny and reef knots can be seen in Figure 5.16. Once again, we can see that reef knots possess a higher average normalized traction force than granny knots for all values of E. Furthermore, similar to μ, the traction forces for both knots

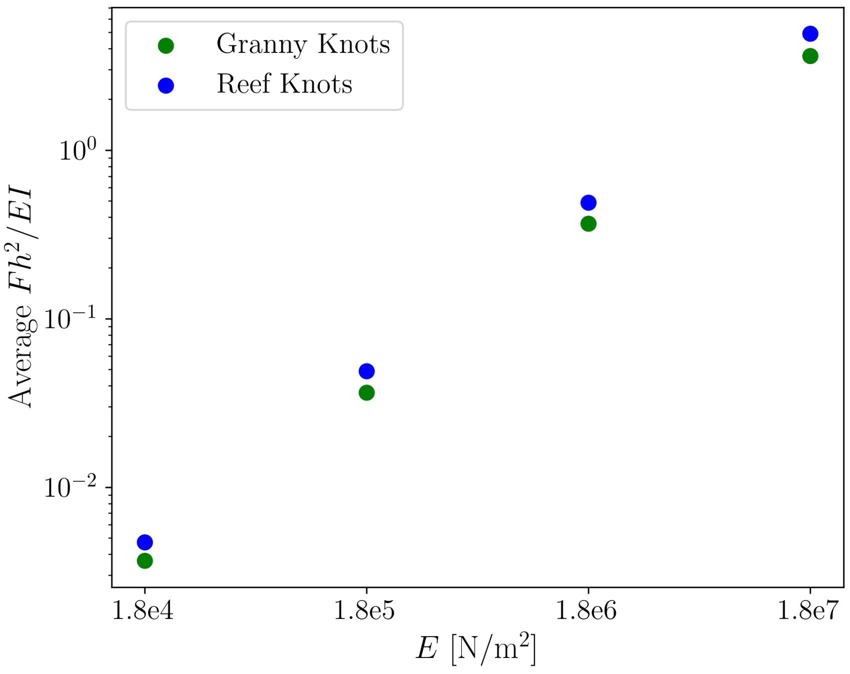


Figure 5.16: Study comparing the average normalized traction forces for Young's Moduli of $E \in [1.8\text{e}4, 1.8\text{e}5, 1.8\text{e}6, 1.8\text{e}7]$ Pa.

monotonically increase as E increases. This also matches our intuition as increasing E increases the stiffness of the material, which will indirectly increase the magnitude of the friction forces given the high curvatures of the knot loop.

5.9 Conclusion

In this chapter, we introduced an improved version of our fully-implicit and penalty-based frictional contact method, the Implicit Contact Model. To test the performance of our contact model, we formulated an end-to-end simulation framework for the novel and difficult contact scenario of flagella bundling in viscous fluid. For this contact problem, we showed that IMCv2 has comparable performance to the state of the art while maintaining faster convergence. For friction, visually convincing sticking-slipping flagella bundling results were presented and an extensive parametric study for flagella bundling was performed. Furthermore, extensive validation of IMCv2's frictional contact forces was performed for numerous complex knot tying scenarios.

For future work, we wish to improve upon the stability and robustness of our friction model. Despite the implicit formulation, the number of iterations necessary to converge starts to increase as μ increases. Another interesting avenue of research is the use of deep learning to learn physics-based dynamics for simulation. Neural networks, when properly trained, have been known to be able to generate nearly identical outputs as numerical simulations while achieving orders of magnitude reduction in computation. Thus, utilizing the computational efficiency and differentiability of neural networks while maintaining physical realism is a promising direction.

CHAPTER 6

DisMech: A Discrete Differential Geometry-Based Simulator for Soft Robots and Structures

Note: As far as pure software engineering goes, DisMech has been quite easily my most ambitious project as of writing this thesis. I am quite proud of this work in particular, as it combines my extensive experience with soft simulation (DER in Chapter 3), elastic contact (IMCv2 in Chapter 5), optimization, and robotics.

6.1 Introduction

Deformable materials are ubiquitous, from knots to clothing to new types of machines. Despite the prevalence of deformable materials in everyday life, there is a lack of physically accurate and efficient continuum mechanics simulations for complicated and arbitrarily-shaped robots, particularly in hardware. As opposed to their rigid-body counterparts, deformable structures such as rods and shells possess infinite degrees of freedom and are capable of undergoing highly nonlinear geometric deformations from even minute external forces, requiring specialized simulators.

Accurate and efficient soft physics simulators serve two key purposes in the robotics community: they allow for 1) training traditional rigid manipulators to intelligently handle deformable objects (Lin et al., 2021) with sim2real realization (Choi et al., 2023c; Tong et al., 2023c) and 2) for the modelling, training, and development of controllers for soft robots themselves (Huang et al., 2020). As the demands for efficient large-scale robot learning necessitate viable sim2real strategies, the need for such simulators becomes even more pressing. With this in mind, we introduce DisMech, a full end-to-end discrete differential geometry (DDG)-based physical simulator for both soft continuum robots and structures. Based on the Discrete Elastic Rods (DER) (Bergou et al., 2008, 2010) framework, DisMech is designed to allow users to create custom geometric configurations composed of individual elastic rods. Such configurations can be expressed quickly through an elegant API, allowing for rapid prototyping of different soft robot builds. Actuation of the soft robot is then readily achievable by manipulating the natural curvatures of the individual elastic rods (i.e., limbs), enormously simplifying sim2real control tasks.

As opposed to previous simulation frameworks focusing on soft physics (Graule et al., 2022; Mathew et al., 2022; Coevoet et al., 2017; Gazzola et al., 2018), DisMech's equations of motion are handled fully implicitly, allowing for far more aggressive time step sizes than previous formulations. This allows our simulations to reach an order of magnitude speed increase over previous state-of-the-art simulators while maintaining physical accuracy as we show later for several canonical validation cases. Though prior work implemented a DDG-based simulation for soft robot locomotion (Huang et al., 2020), this proof-of-concept operated solely in 2D, lacked elastic contact and self-contact, and was specialized to only one simple robot morphology. In comparison, DisMech is a generalizable DDG-based framework for arbitrary soft rod-like robots in contact-rich 2D and 3D environments, and includes an algorithm to map hardware motions to intuitive DDG control inputs. This chapter therefore compares a DDG simulation against the contemporary suite of soft robot simulators for the first time.

This chapter's contributions are:

1. We introduce a methodology and open-source implementation[*] of DisMech as a fully implicit simulator supporting soft physics, frictional contact, and intuitive control inputs. To the best of our knowledge, DisMech is the first general purpose DDG-based 3D simulation framework for easy use by the robotics community.
2. We numerically validate DisMech through several complicated simulations, and compare with the state-of-the-art framework Elastica (Gazzola et al., 2018), showing

[*]See `https://github.com/StructuresComp/dismech-rods`.

comparable accuracy while obtaining an order of magnitude speed increase.

3. We demonstrate generalizability of DisMech for design simulations of dynamic soft robots, including a four-legged spider and an active entanglement gripper (Becker et al., 2022).
4. Finally, we propose a generalizable gradient descent algorithm to map hardware robot data to DisMech's natural curvature parameters, validating their use as control inputs in "real2sim" of real world soft manipulator (Pacheco Garcia et al., 2023).

Despite DER's non-consideration of shearing, we argue that shearing effects are negligible so long as the radii of the rod is small enough. Though this disqualifies DisMech from accurately modeling structures such as muscle fibers, the DDG-based framework of DisMech allows for an intuitive implicit formulation resulting in order of magnitude speed increases while maintaining highly accurate results for a large array of soft structures. In addition to computational benefits, DisMech's framework also allows for the natural incorporation of shell structures via the DDG-based framework Discrete Shells (Grinspun et al., 2003), something that is rather nontrivial to accomplish in frameworks such as SoRoSim (Mathew et al., 2022) and Elastica (Gazzola et al., 2018).

6.2 Methodology

6.2.1 Elasticity, Contact, and Friction

As mentioned previously, the soft physics of DisMech is based off of the DER framework while contact and friction are handled using IMCv2. For brevity, we simply refer readers wishing to understand the mathematical formulation to Chapters 3 and 5.

6.2.2 Elastic Joints

To allow for creating connections and grid-like structures, we introduce the concept of "elastic joints", i.e., nodes along a rod that have one or more other rods connected to them. Such connections experience the same stretching, bending, and twisting energy constraints formulated in Section 3.3 as individual rods. Bending and twisting forces at joints are computed for every possible edge combination. Examples of elastic joints can be seen in Figures 6.4 and 6.6 as blue spheres.

6.2.3 Actuation via Natural Curvatures

Given how the material frames are first setup, we can actuate a soft robot by manipulating its natural curvature $\bar{\boldsymbol{\kappa}}$ or equivalently its natural turning angle $\bar{\boldsymbol{\phi}}$. This in effect results in a bending energy differential in (3.5) which produces contractions and/or relaxations of the elastic rod along the respective material directors. Later in Section 6.4.2, we show how this actuation can be framed as a control input, using a gradient descent method to calculate $\bar{\boldsymbol{\kappa}}$ values for real2sim open-loop control of a hardware robot prototype.

6.2.4 Numerical Integration Scheme

Currently, DisMech has been outfitted to support both explicit (e.g., Elastica's Verlet position (Gazzola et al., 2018)) and implicit numerical integration schemes. For implicit schemes, we offer both backward Euler and implicit midpoint. As backward Euler results in artificial damping (Huang and Jawed, 2019), we opt to use implicit midpoint for scenarios where energy loss is unwanted and backward Euler for scenarios where stability is preferred. To further maintain numerical stability, DisMech is outfitted with both a line search method (Tong et al., 2023b) (same as Section 5.5.3) as well as adaptive time stepping as optional features when an implicit scheme is chosen.

6.3 Theoretical Validation

In this section, we perform physical validation of our model through several canonical experiments. We compare our results with both theory (when appropriate) and the state-of-the-art framework Elastica (Gazzola et al., 2018). Efficiency metrics such as time step size and computational time for all experiments are listed in Table 6.1 while

Table 6.1: Simulation runtime comparison between DisMech and Elastica for various simulation scenarios.

Simulation Framework	Metrics [s]	Sim Validation Experiments					
		Parameters	Beam F.6.1a	Beam F.6.1b	Helix F.6.2a	Helix F.6.2b	Friction F.6.3
		N	201	201	100	100	26
		h [m]	0.020	0.020	0.005	0.001	0.025
		E [Pa]	1×10^5	1×10^7	1×10^7	1×10^9	1×10^5
		T [s]	100	20	10	2	1.5 (44 sims)
Elastica	Max Δt	—	2.5×10^{-4}	3×10^{-5}	4×10^{-5}	1×10^{-6}	1×10^{-4}
	Comp. Time	—	23.80	40.01	11.77	89.24	7.42
DisMech	Best Δt	—	5×10^{-1}	5×10^{-2}	6×10^{-3}	3×10^{-3}	5×10^{-3}
	Comp. Time	—	0.661	1.24	7.30	3.18	1.59

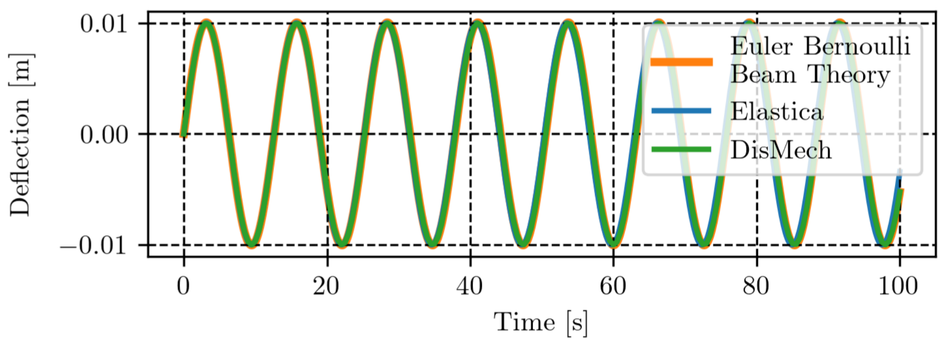
simulation results are plotted in Figures 6.1, 6.2, and 6.3. For Elastica, the largest time step size before numerical instability arose was used while for DisMech, the time step size that had the best tradeoff between number of iterations and computational time was used. For all experiments, we assume a Poisson ratio of 0.5 and a gravitational acceleration of $g = 9.8\,\mathrm{m/s^2}$ for those involving gravity. All simulations were run on a workstation containing an Intel Core i7-9700K (3.60GHz×8) processor and 32GB of RAM.

6.3.1 Dynamic Cantilever Beam

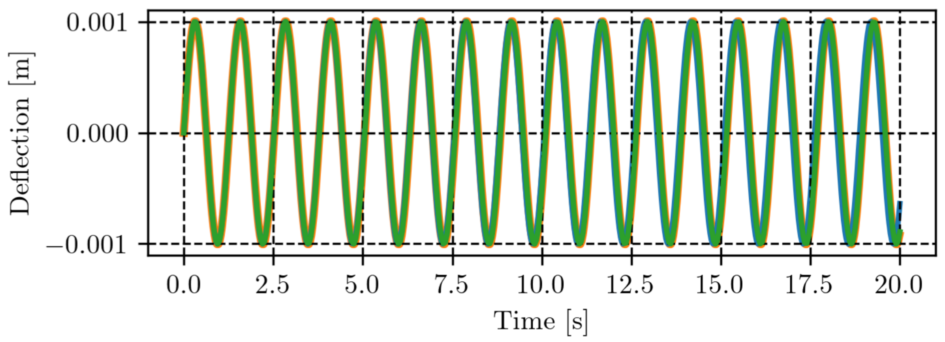
The first validation experiment we conduct is comparing the deflections of a rod modelled as a cantilever beam against Euler-Bernoulli beam theory. In absence of external loads, we can refer to the free vibration equation

$$w(s,t) = \mathrm{Re}\left[\hat{w}(s)e^{-i\omega t}\right], \tag{6.1}$$

where w is the z-direction deflection at arc length $s \in [0, L]$ at time t; ω is the frequency of the vibration, and $\hat{w}(x)$ is the natural frequency of the beam. We use the analytical solution available for cantilever beams (Han et al., 1999) for an initial tip velocity of $5\,\mathrm{mm/s}$. With this, we conduct two simulations to showcase the effect of material stiffness on time step size. For both experiments, we use a rod radius $h = 2\,\mathrm{cm}$, density $\rho = 500\,\mathrm{kg/m^3}$, and rod length $L = 1\,\mathrm{m}$. A discretization of $N = 201$ nodes is also used.



(a) $E = 0.1\,\text{MPa}$



(b) $E = 10$ MPa

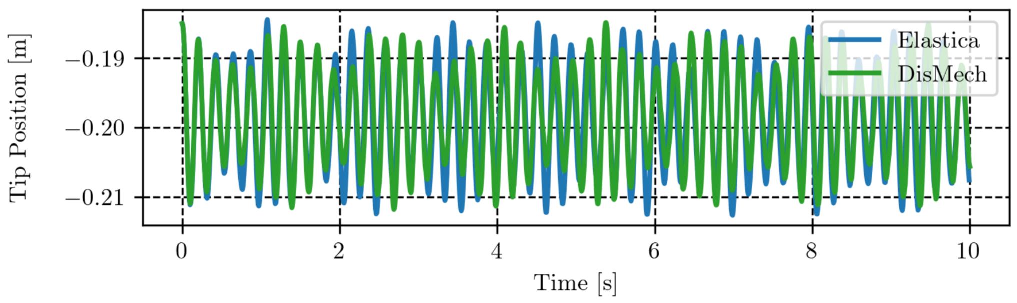
Figure 6.1: Dynamic cantilever simulation results for DisMech and Elastica compared to Euler-Bernoulli beam theory. Deflection of the tip is shown for different material properties.

For the first and second experiments, we then use a Young's modulus E of $0.1\,\text{MPa}$ and $10\,\text{MPa}$ and a sim time T of $100\,\text{s}$ and $20\,\text{s}$, respectively.

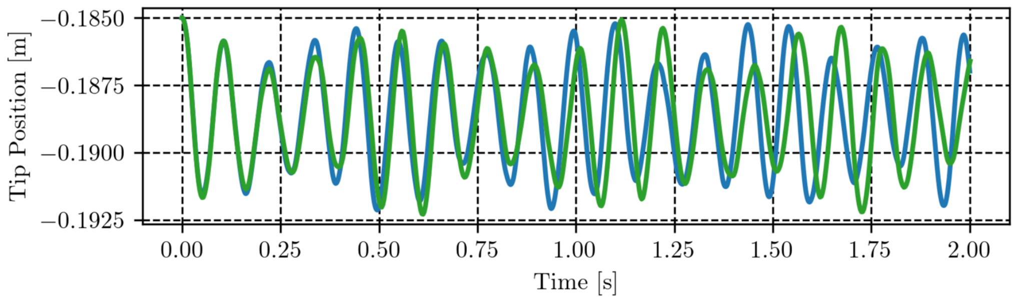
Deflections in Figure 6.1 show that both Elastica and DisMech have excellent agreement with theory (6.1), and neither suffer from artificial damping. However, DisMech achieves these results with time step sizes three orders-of-magnitude larger than Elastica, corresponding to an order of magnitude speed increase.

6.3.2 Oscillating Helix under Gravity

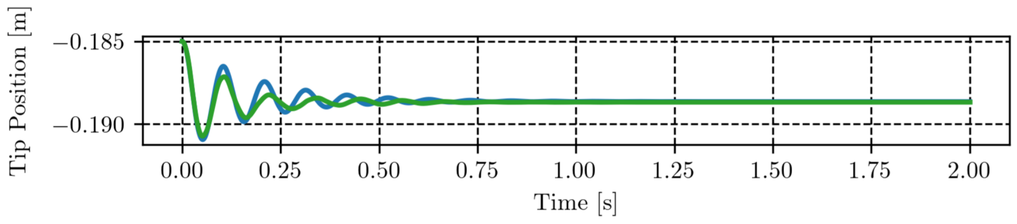
The next experiment consists of a suspended helical rod oscillating under gravity. We use the same experiment setup as Huang and Jawed (2019) with density $\rho = 1273.52\,\text{kg/m}^3$, helix radius of $2\,\text{cm}$, pitch of $5\,\text{cm}$, and a contour length of $0.5\,\text{m}$ (resulting in an axial length of $\approx 0.185\,\text{m}$). A discretization of $N = 100$ is used. Like the previous section, we



(a) $E = 10\,\mathrm{MPa}$ $h = 0.005\,\mathrm{m}$



(b) $E = 1\,\mathrm{GPa}$ $h = 0.001\,\mathrm{m}$



(c) Equilibrium Point via Damping

Figure 6.2: Helix oscillating under gravity simulation results for DisMech and Elastica. The bottom tip position of the helical rod is shown for different material and geometric properties. We also show both DisMech and Elastica reaching the same static equilibrium point when damping is introduced into the system as a sanity check.

also conduct two experiments with a Young's modulus E of $10\,\mathrm{MPa}$ and $1\,\mathrm{GPa}$, radius h of $5\,\mathrm{mm}$ and $1\,\mathrm{mm}$, and sim time T of $10\,\mathrm{s}$ and $2\,\mathrm{s}$, respectively.

As an analytical solution for comparison is not available, we simply plot the oscillating tip positions as shown in Figures 6.2a-6.2b. As shown, both simulation frameworks produce results with identical frequencies for the first experiment and near identical frequencies for the second. To confirm that the difference in results is not a result of improper system representation, we also show that both DisMech and Elastica reach the same static equilibrium point in Figure 6.2c after introducing damping. Therefore,

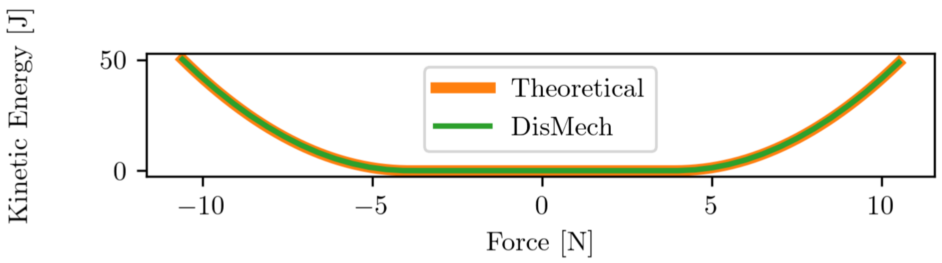


Figure 6.3: Axial friction simulation showcasing the kinetic energy of a rod as a function of an enacted external force for a friction coefficient of $\mu = 0.4$.

we presume that these slight frequency differences are numerical artifacts arising from differences in integration schemes and/or time step size. As before, we also report being able to take time step sizes of up to three orders-of-magnitude, correlating to an order of magnitude speed increase.

6.3.3 Friction Validation

For the final experiment, we test axial friction on a rod. We use a rod with radius $h = 2.5\,\text{cm}$, density $509.3\,\text{kg/m}^3$, length $L = 1\,\text{m}$, and Young's modulus $E = 0.1\,\text{MPa}$. A discretization of $N = 26$ is used. Floor contact was simulated using IMCv2 using a friction coefficient $\mu = 0.4$, distance tolerance $\delta = 5\text{e}{-4}\,\text{m}$, and slipping tolerance $\nu = 1\text{e}{-3}\,\text{m/s}$. With this, we can compute the kinetic energy of the rod when experiencing a constant uniform push/pull force $\mathbf{F}^{\text{p}}$ as

$$E_k = \begin{cases} 0 & \|\mathbf{F}^{\text{p}}\| \leq |\mu m g|, \\ \dfrac{t^2}{2m}\left(\|\mathbf{F}^{\text{p}}\| - \mu m g\right)^2 & \|\mathbf{F}^{\text{p}}\| > |\mu m g|, \end{cases} \tag{6.2}$$

where m is the mass of the rod and t is the time after starting force exertion from a resting configuration. We compute results for 44 simulations in parallel for a range of forces $\|\mathbf{F}^{\text{p}}\| \in [-10.6, 10.5]\,\text{N}$. When observing results in Figure 6.3, we see excellent agreement between DisMech's simulated kinetic energy and theory for a sim time of $T = t = 1.5\,\text{s}$. For further more rigorous validation of the contact used by DisMech, we refer readers to Sections 5.7 and 5.8.

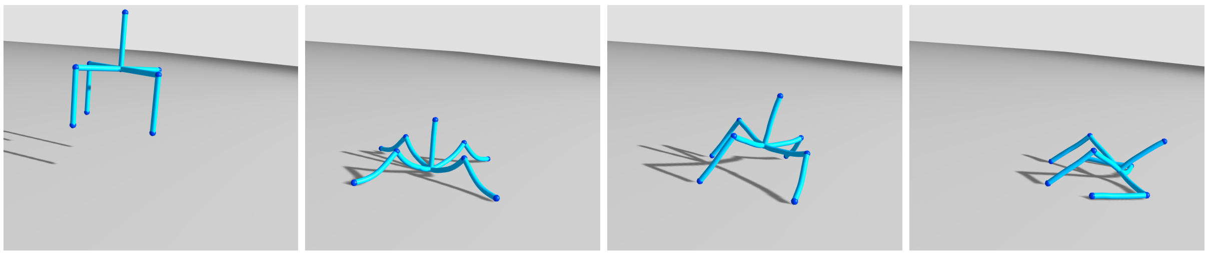


Figure 6.4: Rendering of a four-legged spider-like soft robot constructed using DisMech's API. Moving chronologically from left to right, the robot is dropped from a height where it then makes contact with an incline plane. After some initial sliding, the influence of sticking friction results in the robot's eventual equilibrium position in the rightmost column.

6.4 Practical Demonstrations for Flexible Robots

Three demonstrations below showcase the accuracy and ease-of-use of DisMech over competing frameworks in simulating complicated flexible and soft robots. We also provide an algorithm that maps hardware robot motions to DisMech's control inputs, emphasizing its generalizability.

6.4.1 Arbitrary Robot Prototyping

6.4.1.1 Spider Robot

We showcase the ease-of-use and environmental contact capabilities of DisMech by simulating a four-legged spider-like soft robot composed of interconnected rods (Figure 6.4). Individual limbs are created as rods, with joints between them, using a single line of code each. We simulate dropping the robot onto a floor (no actuation) with $\mu = 0.4$, and replicate an incline using gravity $\mathbf{g} = [0.707, 0.707, -9.8]\,\mathrm{m/s}^2$. We can observe visually plausible results of the robot colliding with the floor, rebounding, and then after an initial sliding period, coming to static equilibrium via stiction.

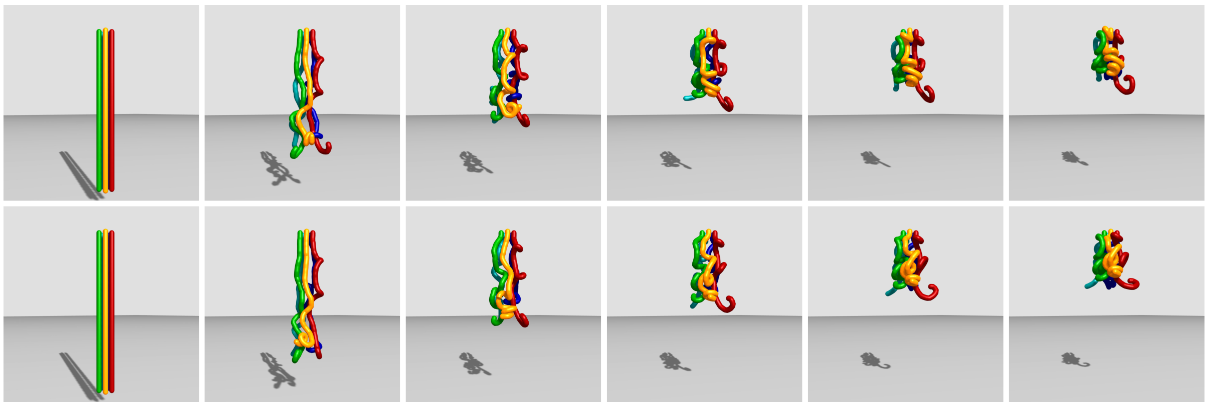


Figure 6.5: Renderings of an active entanglement gripper (Becker et al., 2022). The top row showcases contact-only entanglement whereas the bottom row showcases entanglement with friction $\mu = 0.5$. Note the influence of stiction causing more distorted helices in the latter example.

6.4.1.2 Active Entanglement Gripper

Next, we further showcase the generality of DisMech by simulating an active entanglement gripper (Becker et al., 2022), a highly nontrivial frictional contact case. To do so, we simulate five fingers (rods) placed equidistantly along a circular perimeter, each with length $L = 0.3\,\text{m}$, radius $h = 5\,\text{mm}$, density $\rho = 1200\,\text{kg/m}^3$, and Young's modulus $E = 0.3\,\text{MPa}$. Each rod is discretized using $N = 60$. Contact is simulated using $\delta = 5\text{e}{-4}\,\text{m}$ and $\nu = 1\text{e}{-3}\,\text{m/s}$. To simulate rapid entanglement, each edge is actuated via a random $\bar{\phi}$ value uniformly sampled from a range of $[0, 40]$ deg.

Results for both contact-only and frictional contact ($\mu = 0.5$) scenarios show convincing, visually plausible results (Figure 6.5). When comparing between the two, friction causes the rods to stick to each other during the coiling, resulting in more distorted helices. Simulations such as these open up many opportunities for generating control policies.

6.4.2 Real2Sim Open-Loop Control

Finally, we provide a method to use DisMech to simulate an open-loop trajectory of a real soft robot in hardware, as a form of open-loop control. Here, we use a shape memory alloy (SMA) actuated double limb soft manipulator (Pacheco Garcia et al.,

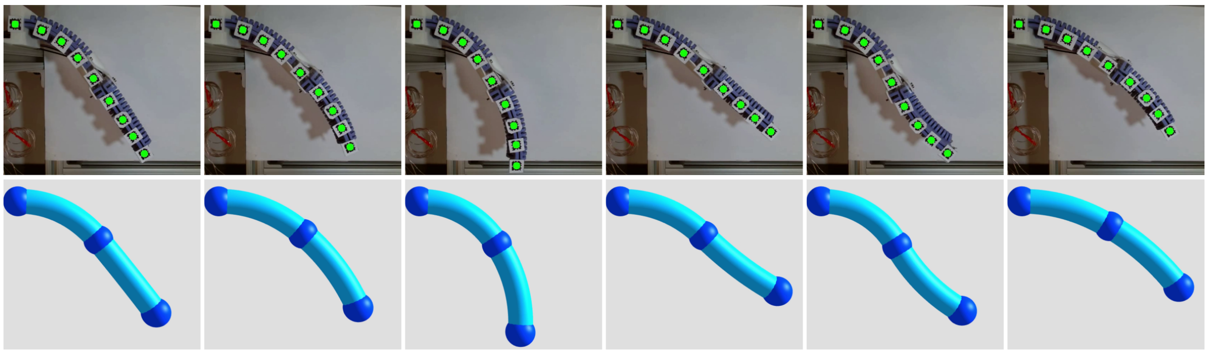


Figure 6.6: Snapshots showcasing real2sim realization of a SMA actuated dual soft limb manipulator (Pacheco Garcia et al., 2023). Using our gradient descent approach, we showcase excellent agreement between the real and simulated curvatures for a wide variety of geometric configurations.

2023) (Figure 6.6). The robot's limbs are $76 \times 49 \times 9\,\text{mm}$, and are constructed from a silicone polymer (Smooth-On Smooth-Sil) with a density of $1240\,\text{kg/m}^3$ and a Young's modulus of $1.793\,\text{MPa}$. Despite the manipulator being quite wide, we can still represent the manipulator as a rod since manipulation occurs primarily in 2D (Choi et al., 2023c). For the rod radius, we used $10\,\text{cm}$ to compensate for the ridges of the limbs. Two rods are initialized with the aforementioned parameters to represent each limb, which are then connected via an elastic joint.

6.4.2.1 Solving Natural Curvature via Gradient Descent

To achieve real2sim realization, we must first calculate the appropriate $\bar{\boldsymbol{\kappa}}$ values that generate geometric configurations of the robot corresponding to hardware data (Section 6.2.3). Given the nonlinearity of the robot's geometry, coupled with deformations produced by gravity, solving for the appropriate natural curvatures analytically is nontrivial. Therefore, we propose a gradient descent-based approach to solve for $\bar{\boldsymbol{\kappa}}$.

In our approach, we assume that the movement of the limbs is quasistatic and that each limb has more-or-less a constant curvature along its length. We use the AprilTag library to extract the arm's position from video (Figure 6.6, green dots). Using these positions, we can then compute the target curvatures $\boldsymbol{\kappa}^*$ using (3.4).

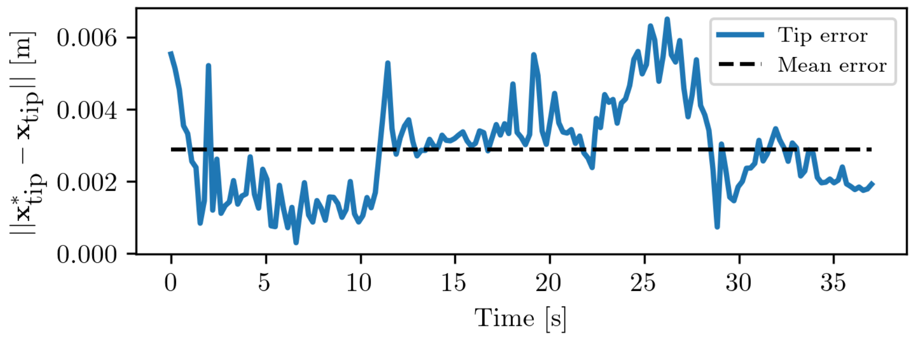


Figure 6.7: Tip error between the target tip position $\mathbf{x}^*_{\text{tip}}$ and the simulated tip position $\mathbf{x}_{\text{tip}}$ when carrying out our gradient descent natural curvature actuation.

We then define a loss

$$\lambda(\bar{\kappa}_1, \bar{\kappa}_2) = |\kappa_1^* - \kappa_1| + |\kappa_2^* - \kappa_2|, \tag{6.3}$$

where κ_1 and κ_2 are the resulting simulated curvatures of the first and second limbs when changing the natural curvatures to values $\bar{\kappa}_1$ and $\bar{\kappa}_2$, respectively. The gradient of λ with respect to $\bar{\boldsymbol{\kappa}}$ can then be obtained using a forward finite difference approach:

$$\nabla_{\bar{\boldsymbol{\kappa}}}\lambda = \begin{bmatrix} \dfrac{\partial\lambda}{\partial\bar{\kappa}_1} \\ \dfrac{\partial\lambda}{\partial\bar{\kappa}_2} \end{bmatrix} = \frac{1}{\epsilon}\begin{bmatrix} \lambda(\bar{\kappa}_1 + \epsilon, \bar{\kappa}_2) - \lambda(\bar{\kappa}_1, \bar{\kappa}_2) \\ \lambda(\bar{\kappa}_1, \bar{\kappa}_2 + \epsilon) - \lambda(\bar{\kappa}_1, \bar{\kappa}_2) \end{bmatrix}, \tag{6.4}$$

where ϵ is a small input perturbation. Using this finite difference gradient, we can then iteratively solve for the correct $\bar{\boldsymbol{\kappa}}$ by updating

$$\bar{\boldsymbol{\kappa}} = \bar{\boldsymbol{\kappa}} - \alpha\nabla_{\bar{\boldsymbol{\kappa}}}\lambda \tag{6.5}$$

until λ reaches below a set tolerance, where α is a step size. The full psuedocode for this approach can be seen in Alg. 6.

We demonstrate excellent real2sim realization of the soft limb actuator through a DisMech model both visually (Figure 6.6) and numerically (Figure 6.7), where we achieve an average tip position error of just 2.9 mm (1.5% of the robot's length). Future work, using more data, will derive separate actuator models (e.g. SMA voltages to natural

Algorithm 6: Real2Sim via Gradient Descent

```
Input: $\mathbf{D} \leftarrow$ geometric configuration of robot
Output: $\boldsymbol{\tau}_{\bar{\boldsymbol{\kappa}}} \leftarrow$ trajectory of $\bar{\boldsymbol{\kappa}}$s
Func SolveNaturalCurvatures ($\mathbf{D}$):
    $\boldsymbol{\kappa}^* \leftarrow$ ComputeCurvature($\mathbf{D}$)                    // (3.4)
    $\boldsymbol{\tau}_{\bar{\boldsymbol{\kappa}}} \leftarrow [\ ]$                                   // trajectory
    $\bar{\boldsymbol{\kappa}} \leftarrow [0, 0]$
    for $\kappa_1^*, \kappa_2^* \in \boldsymbol{\kappa}^*$ do
        $\alpha \leftarrow 0.1$                                      // step size
        $\lambda_{\text{prev}} \leftarrow \infty$
        while True do
            $\kappa_1, \kappa_2 \leftarrow \mathcal{F}(\bar{\boldsymbol{\kappa}})$                      // DisMech Sim
            $\lambda \leftarrow |\kappa_1^* - \kappa_1| + |\kappa_2^* - \kappa_2|$           // (6.3)
            if $\lambda <$ tolerance then
                break
            end
            if $\lambda > \lambda_{\text{prev}}$ then
                $\alpha \leftarrow 0.5\alpha$
            end
            $\lambda_{\text{prev}} \leftarrow \lambda$
            $\nabla_{\boldsymbol{\kappa}}\lambda \leftarrow$ finite diff with $\epsilon$                 // (6.4)
            $\bar{\boldsymbol{\kappa}} \leftarrow \bar{\boldsymbol{\kappa}} - \alpha\nabla_{\boldsymbol{\kappa}}\lambda$                       // (6.5)
        end
        $\boldsymbol{\tau}_{\bar{\boldsymbol{\kappa}}}$.append($\bar{\boldsymbol{\kappa}}$)
    end
    return $\boldsymbol{\tau}_{\bar{\boldsymbol{\kappa}}}$
```

curvatures), which in turn will allow for feedback controller design via sim2real.

6.5 Conclusion

In this chapter, we introduced DisMech, a fully generalizable, implicit, discrete differential geometry-based physical simulator capable of accurate and efficient simulation of soft robots and structures composed of elastic rods. We physically validated DisMech through several simulations, using both representative examples and new and complicated robots, showcasing an order of magnitude computational gain over previous methods. In addition, we also introduced a method for intuitively actuating soft continuum robots by bending energy manipulation via natural curvature changes, and demonstrated this actuation's use as a control input in real2sim of a soft manipulator.

Future work will involve integrating shells (Grinspun et al., 2003) into DisMech to allow for more complex deformable assemblies. In addition to this, the incorporation of shearing into DisMech will be a key research area to allow it to simulate all deformation modes similar to Cosserat rod-based frameworks (Gazzola et al., 2018; Mathew et al., 2022). In fact, shearing has previously been integrated into a DDG-based framework when simulating a Timoshenko beam (Li et al., 2020b), albeit for just a 2D framework. Finally, efficient pipelines for real2sim2real modelling and training through reinforcement learning using DisMech will be a key research area moving forward, whether it be for soft robot control or deformable object manipulation.

CHAPTER 7

Learning Neural Force Manifolds for Sim2Real Robotic Paper Folding

Note: We now transition from purely simulation-based works to more robotic applications. To start, we will be focusing on the first of two nontrivial sim2real robotic manipulation problems: single manipulator paper sheet folding.

7.1 Introduction

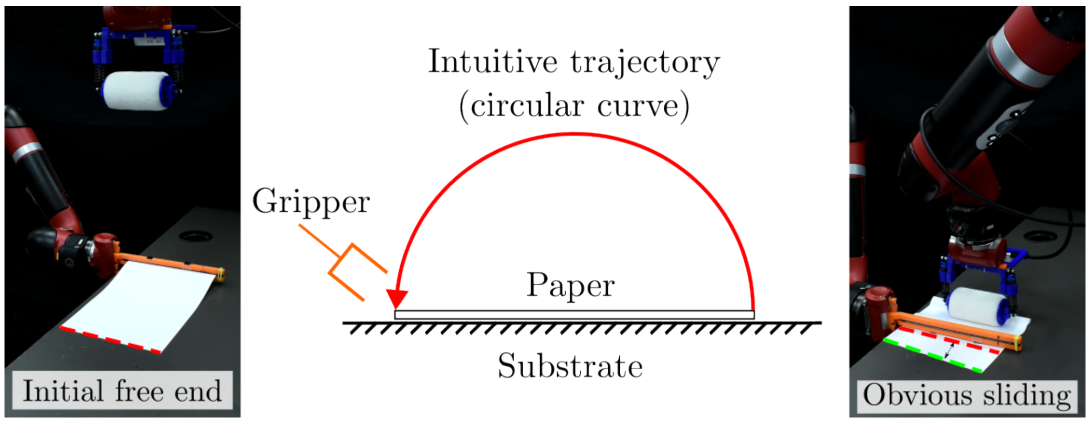
From shoelaces to clothes, we encounter flexible slender structures throughout our everyday lives. These structures are often characterized by their ability to undergo large deformations when subjected even to moderate forces, such as gravity. Therefore, the robotic manipulation of deformable objects is highly nontrivial as a robot must be able to take into account future deformations of the manipulated object in order to complete manipulation tasks successfully.

Prior research has focused primarily on manipulating either cloth or ropes (Sanchez et al., 2018; Yin et al., 2021), and, as a result, the challenge of robotically manipulating many other deformable objects still lacks robust solutions. This chapter addresses a particularly difficult deformable manipulation task — folding paper. Paper is similar to cloth but typically possesses a significantly higher bending stiffness and a slippery surface. Therefore, when compared to folding garments and fabrics, the folding of paper requires more delicate and insightful manipulations. In fact, in our experiments we observe that state-of-the-art methods for robotic fabric/cloth folding (Petrík et al., 2016; Petrík and Kyrki, 2019; Petrík et al., 2020) perform poorly when transferred to paper.

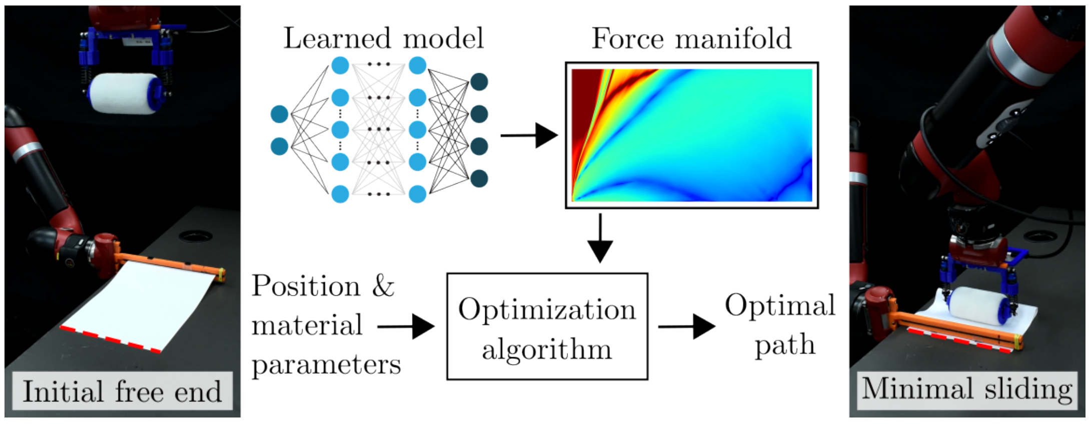
To tackle these challenges, we propose a framework that combines physically accurate simulation, scaling analysis, and machine learning to generate folding trajectories optimized to prevent sliding. With scaling analysis, we make the problem non-dimensional, resulting in both dimensionality reduction and generality. This allows us to train a single nondimensionalized neural network, whose outputs are referred to as a neural force manifold (NFM), to continuously approximate a scaled force manifold sampled purely from simulation. Compared to numerical models that require the entire geometric configuration of the paper, NFMs map the external forces of the paper given only the grasp position. Therefore, we can generate trajectories optimized to minimize forces (and thus minimize sliding) by applying path planning algorithms. Furthermore, the nondimensionality of the NFM allows us to generate trajectories for paper of various materials and geometric properties even if such parameters were not present in the training dataset. We show that our approach is capable of folding paper on extremely slick surfaces with little-to-no sliding (Figure 7.1).

Overall, our main contributions in this chapter are as follows:

1. We formulate a solution for folding materially homogeneous sheets of paper along symmetrical centerlines in a physically robust manner using scaling analysis, resulting in complete generality concerning the modulus and density of the material, size of the paper, and environmental properties (e.g., friction).
2. Next, we generate accurate non-dimensional simulation data to train a "neural force manifold" for optimal trajectory generation. We exploit the high inference speed of our trained model with a perception system to construct a robust and efficient closed-loop model-predictive control algorithm for the folding task in near real time.
3. Finally, we demonstrate full sim2real realization through an extensive robotic case study featuring 360+ folding experiments involving paper sheets of various materials and shapes. We compare our method against both natural paper folding strategies as well as the previous state of the art in robotic rectangular fabric folding (Petrík et al., 2020, 2016).



(a) Manipulation via an Intuitive Trajectory



(b) Manipulation via our Optimal Trajectory

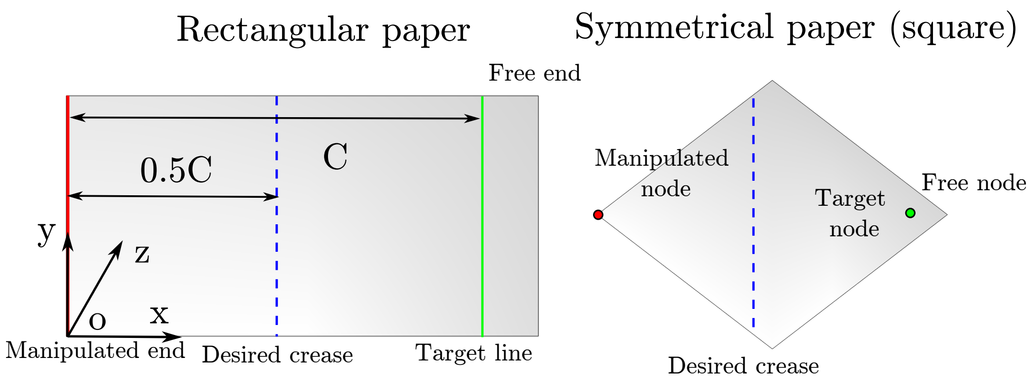
Figure 7.1: Half valley folding for A4 paper with (a) intuitive manipulation and (b) our designed optimal manipulation. An intuitive manipulation scheme such as tracing a semicircle experiences significant sliding due to the bending stiffness of the paper, resulting in a poor fold. By contrast, our optimal manipulation approach achieves an excellent fold by taking into consideration the paper's deformation to minimize sliding.

Moreover, we offer demonstration videos and release all our code as open-source software.[*]

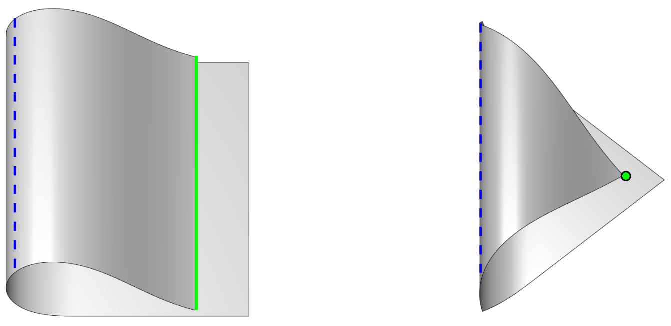
7.2 Problem Statement

This chapter studies a simple yet challenging task in robotic folding: creating a predefined crease on a sheet of paper of typical symmetrical geometry (e.g., rectangular, diamond, etc.) as illustrated in Figure 7.2. Only one end of the paper is manipulated while the

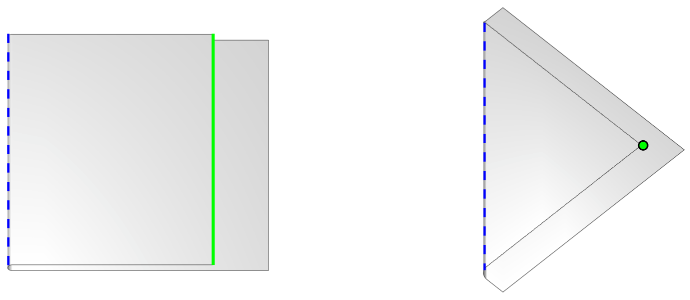
[*]See https://github.com/StructuresComp/deep-robotic-paper-folding.



(a) Initial State



(b) Folding State



(c) Folded State

Figure 7.2: States of the paper during the folding process. The manipulation process involves two steps. The first (folding) step transitions the paper from the initial state (a), where the paper lies flat on the substrate, to the folding state (b), where the manipulated end is moved to the "crease target" line C. The second (creasing) step then transitions the paper from state (b) to the final folded state (c), which involves forming the desired crease on the paper.

other end is left free. Thus, extra fixtures are unnecessary and the folding task can be completed by a single manipulator, which simplifies the workspace, but slippage of the paper against the substrate must be mitigated during manipulation, which presents a challenge.

The task can be divided into two steps. The first is manipulating one end of the paper from the initial flat state (Figure 7.2a) to the folding state (Figure 7.2b), with the goal that the manipulated edge or point should overlap precisely with the crease target line or point C as shown in the figure. In the second step, the paper is then permanently deformed to form the desired crease at $C/2$, thus achieving the final folded state (Figure 7.2c).

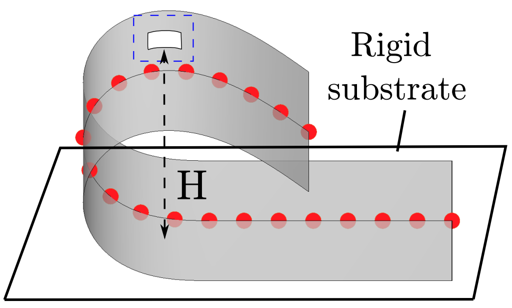
As creasing the paper is trivial, the main challenge lies in minimizing the displacement of the free end of the paper during the first step. The paper's large nonlinear deformations and slippery surface make accurate predictions of the folding paper's status crucial for minimizing displacement. Since permanent deformations are absent in the first step, we model the paper's nonlinear deformations using a 2D planar rod model with a linear elastic assumption, which is discussed in detail in the next section. This physical model is then combined with scaling analysis and machine learning to generate physically-informed folding trajectories optimized to minimize sliding. With the first step concluded, simple motion primitives are used to complete the final paper creasing.

7.3 Reduced-Order Model Representation

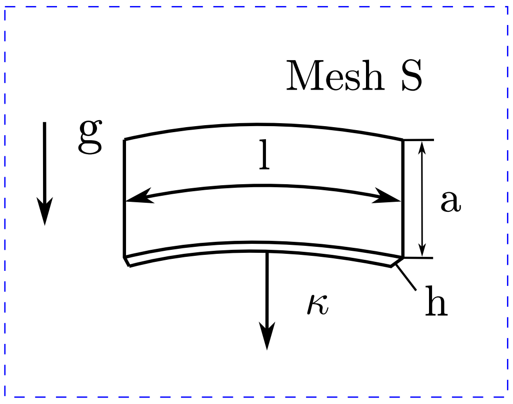
Paper is a unique deformable object. Unlike cloth, its surface is developable (Hilbert and Cohn-Vossen, 2021); i.e., the surface can bend but not stretch. Furthermore, shear deformations are not of particular importance as paper possesses a negligible thickness to length ratio. Therefore, the primary nonlinear deformation when folding paper in our scenario is bending deformation. We postulate that the nonlinear behaviors of paper arise primarily from a balance of bending and gravitational energies: $\epsilon_b \sim \epsilon_g$.

To further understand the energy balance of the manipulated paper, we analyze a finite element of the paper, as shown in Figure 7.3b. The bending energy of this piece can be written as

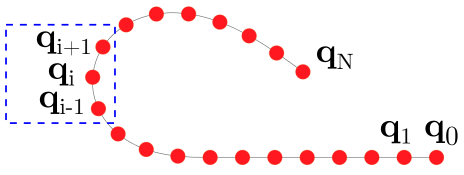
$$\epsilon_b = \frac{1}{2} k_b \kappa^2 l, \tag{7.1}$$



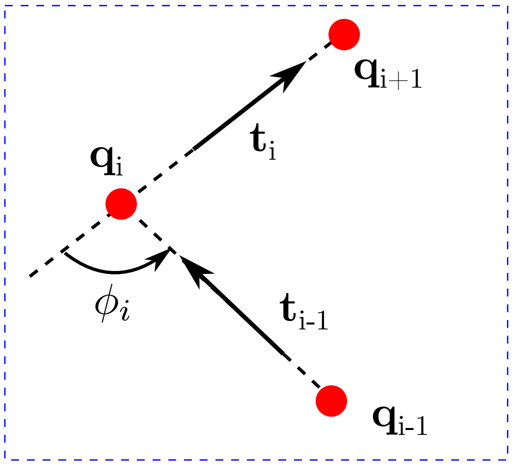
(a) Schematic of Paper during Folding



(b) Finite Mesh of Paper



(c) Reduced-Order Planar Rod Model



(d) Discrete Rod Notations

Figure 7.3: Discrete models for paper folding. The top row showcases a (a) schematic of paper folding discrete centerline with (b) a close up of the bending deformations of a finite mesh element of the paper. The bottom row shows an equivalent (c) reduced-order planar rod model (notice the overlay in (a)) with (d) DER notation.

where l is its undeformed length of the piece, κ is its curvature, and its bending stiffness is

$$k_b = \frac{1}{12} E w h^3, \tag{7.2}$$

where w is its undeformed width, h is its thickness, and E is its Young's modulus. The gravitational potential energy of the piece is

$$\epsilon_g = \rho w h l g H, \tag{7.3}$$

where ρ is its volume density and H is its vertical height above the rigid substrate.

From the above equations, we obtain a characteristic length called the gravito-bending length, which encapsulates the influence of bending and gravity:

$$L_{gb} = \left(\frac{Eh^2}{24\rho g}\right)^{\frac{1}{3}} \sim \left(\frac{H}{\kappa^2}\right)^{\frac{1}{3}}. \tag{7.4}$$

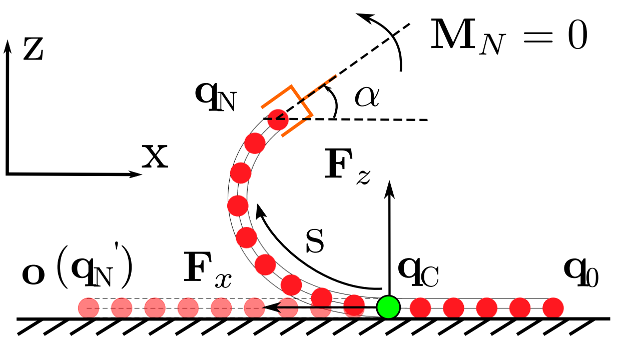
The length is in units of meters, and we can observe that it scales proportionally to the ratio of vertical height to curvature squared, which are the key quantities describing the deformed configuration of the manipulated paper. Note that the formulation of L_{gb} contains only one geometric parameter, the paper thickness h, which means that other geometric quantities (i.e., length l and width w) have no influence on the deformed configuration.

Additionally, due to the symmetrical geometry and material homogeneity of the paper, the curvature κ should be identical for all regions at the same height H. Therefore, we can simply use the centerline of the paper, as shown in Figure 7.3a, to express the paper's configuration. As deformations are limited to the x-z plane, this centerline is simply a 2D planar rod. This allows us to simulate paper folding with DER (refer to Chapter 3 for review). In the next section, we quickly formulate a simplified 2D DER formulation.

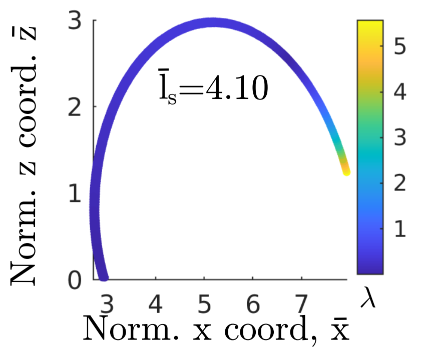
7.3.1 Simple 2D DER Formulation

As shown in Figure 7.3c, the discrete model is comprised of $N+1$ nodes, $\mathbf{q}_i$ $(0 \le i \le N)$. Each node $\mathbf{q}_i$ represents two degrees of freedom (DOF): position along the x and the z axes. This results in a $2N+2$-sized DOF column vector $\mathbf{q} = [\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_N]^T$ representing the configuration of the paper sheet. Initially, all the nodes of the paper are located in a line along the x-axis in the paper's undeformed state. As the robotic manipulator imposes boundary conditions on the end node $\mathbf{q}_N$, portions of the paper deform against the substrate, as shown in Figure 7.4a. Finally, since deformations are limited to a 2D plane, we can ignore twisting energies. The total elastic energy is therefore $E^{\text{elastic}} = E_s + E_b$.

Indeed, a ratio $k_s/k_b \sim 1/h^2 \gg 1$ indicates that stretching strains will be minimal, which matches our intuition as paper is usually easy to bend but not stretch. Therefore,



(a) Side View of Robot Paper Folding



(b) Sample λ Values

Figure 7.4: (a) Side view of a symmetrical paper during folding with coordinate frames and relevant notations. (b) Sampled λ forces for a particular $\bar{l}_s$ of 4.10. This showcases one of the sampled "partial" force manifolds that we use to train our neural network on.

the stretching energy item in (3.2) acts as a constraint to prevent obvious stretching for the modeled planar rod.

We can now write the equations of motion as a simple force balance

$$\mathbb{M}\ddot{\mathbf{q}} + \frac{\partial E^{\text{elastic}}}{\partial \mathbf{q}} - \mathbf{F}^{\text{ext}} = 0, \tag{7.5}$$

where $\mathbb{M}$ is the diagonal lumped mass matrix, the dots denote time derivatives of $\mathbf{q}$, $\frac{\partial E^{\text{elastic}}}{\partial \mathbf{q}}$ is the elastic force vector, and $\mathbf{F}^{\text{ext}}$ are the external forces acting on the paper. With the inclusion of gravity acting as the external forces, the full 2D planar rod simulation framework is now complete.

7.4 Generalized Solution and Scaling Analysis

As mentioned in Section 7.2, the core of the folding task is to manipulate the end $\mathbf{q}_N$ to the target position C starting from an initially flat state shown in Figure 7.2a. To do so, we analyze the physical system in order to achieve a solution capable of minimizing sliding during manipulation.

7.4.1 Computing the Optimal Force

We first denote several quantities to describe the deformed configuration of the paper. We introduce a point $\mathbf{q}_C$, which is the node that connects the suspended ($z > 0$) and contact regions ($z = 0$) of the paper. We focus solely on the suspended region as deformations occur primarily in this region. An origin $\mathbf{o}$ is defined for our 2D plane which is located at the initial manipulated end $\mathbf{q}_N$, as shown in Figure 7.4a. For the manipulated end, the robot end-effector imposes a position $\mathbf{q}_N = (x, z)$ and an orientation angle α to control the pose of the manipulated end, as shown in Figure 7.4a. We impose a constraint that the curvature at the manipulated end is always zero so that sharp bending deformations are prevented, which is crucial to preventing permanent deformations during the folding process. On the connective node $\mathbf{q}_C$, the tangent is always along the x-axis. With these definitions, we can now modify (7.5) with the following constraints:

$$
\begin{aligned}
\mathbb{M}\ddot{\mathbf{q}} + \frac{\partial E^{\text{elastic}}}{\partial \mathbf{q}} - \mathbf{F}^{\text{ext}} = 0&, \\
\text{such that} \quad \mathbf{q}_N &= (x, z), \\
\frac{\mathrm{d}\mathbf{q}_C}{\mathrm{d}s} &= (-1, 0), \\
M_N &= 0, \\
l_s \equiv \int_{\mathbf{q}_C}^{\mathbf{q}_N} \mathrm{d}s &= \mathbf{q}_C \cdot \hat{\mathbf{x}},
\end{aligned}
\tag{7.6}
$$

where M_N is the external moment applied on the manipulated end, s is the arc length of the paper's centerline, and l_s is the arc length of the suspended region (from $\mathbf{q}_C$ to $\mathbf{q}_N$).

We can solve (7.6) with the numerical framework presented in Section 7.3.1 resulting in a unique DOF vector $\mathbf{q}$. Note that when $\mathbf{q}$ is determined, we can then obtain the external forces from the substrate along the paper $\mathbf{F}_{\text{substrate}} = \mathbf{F}_x + \mathbf{F}_z$, orientation angle α of the manipulated end, and the suspended length l_s. Recall that through (7.4), Young's modulus E, thickness h, and density ρ were determined to be the main material and geometric properties of the paper. Therefore, we can outline the following physical relationship relating all our quantities:

$$\begin{aligned}\lambda &= \frac{\|\mathbf{F}_x\|}{\|\mathbf{F}_z\|},\\(\lambda, \alpha, l_s) &= f(E, h, \rho, x, z),\end{aligned} \tag{7.7}$$

where f is an unknown relationship. It is then trivial to see that to prevent sliding the relationship

$$\lambda \leq \mu_s \tag{7.8}$$

must be satisfied, where μ_s is the static friction coefficient between the paper and the substrate. Therefore, a trajectory that minimizes sliding is one that minimizes λ along its path.

One glaring problem remains in that the relation f must be known to generate any sort of trajectory. In the absence of an analytical solution, the numerical framework from Section 7.3.1 can be used to exhaustively find mappings between the inputs and outputs of f. However, generating tuples in this fashion requires solving the high-dimensional problem in (7.6). Such a method would be horribly inefficient and would make real-time operation infeasible. Instead, we opt to obtain a regression approximation of f by fitting a neural network on simulation data. This approach has several shortcomings, however. For one, directly learning f is time-consuming given that (7.7) is a high-dimensional mapping that depends on five parameters as input. Furthermore, since the formulation directly depends on intrinsic parameters of the paper (E, ρ, and h), an enormously exhaustive range of simulations must be run to gather enough data to accurately learn f.

7.4.2 Nondimensionalization via Buckingham π Theorem

To proceed, we reduce the dimensionality of the problem by applying scaling analysis. Buckingham π theorem is a fundamental principle in dimensional analysis, stating that a physically meaningful equation involving n physical parameters can be expressed using a reduced set of $p = n - k$ dimensionless parameters derived from the original parameters, where k represents the number of physical dimensions (e.g., mass, time, length, etc.). Therefore, according to the Buckingham π theorem, we can construct five unitless groups: $\bar{x} = x/L_{gb}$; $\bar{z} = z/L_{gb}$; $\bar{l}_s = l_s/L_{gb}$; α; and $\lambda = F_t/F_n$, where L_{gb} is the gravito-bending length (7.4). This results in the following unitless formulation of (7.7):

$$(\lambda, \alpha, \bar{l}_s) = \mathcal{F}(\bar{x}, \bar{z}). \tag{7.9}$$

Note that the mapping $\mathcal{F}$ is now independent of quantities with units; e.g., material and geometric properties of the paper. As the dimensionality of our problem has been reduced significantly, we can now express λ as a function of just two parameters $\bar{x}, \bar{z}$. Therefore, training a neural network to model $\mathcal{F}$ is now trivial as non-dimensionalized simulation data from a single type of paper can be used. Furthermore, the low dimensionality of $\mathcal{F}$ allows us easily to visualize the λ landscape along a non-dimensional 2D-plane.

We will detail the steps to model $\mathcal{F}$ in the next section.

7.5 Deep Learning and Optimization

7.5.1 Data Generation

To learn the force manifold, we solve (7.6) for many sampled (x, z) points. An example of the partial force manifold produced from this sampling can be observed for a single suspended length in Figure 7.4b. For a specific (x, z) location, we apply incremental rotations along the y-axis and find the optimal rotation angle α that results in $M_N = 0$ on the manipulated end. For a particular configuration (x, z, α), we then record the suspended length l_s as well as the tangential and normal forces experienced on the clamped end. This leads to a training dataset $\mathcal{D}$ consisting of six element tuples $(F_t, F_n, \alpha, l_s, x, z)$. We then non-dimensionalize this dataset to the form $(\lambda, \alpha, \bar{l}_s, \bar{x}, \bar{z})$.

With our simulation framework, we generated a dataset $\mathcal{D}$ comprising a total of 95,796 training samples within a normalized suspended length of $\bar{l}_s \leq 6.84$ (which adequately covers the workspace of most papers), consuming 3.54 hours of compute time on an AMD

Ryzen 7 3700X 8-core processor.

7.5.2 Learning Force and Optimal Grasp Orientation

To train a neural network model of $\mathcal{F}$,

$$(\lambda, \alpha, \bar{l}_s) = \mathcal{F}_{\text{NN}}(\bar{x}, \bar{z}), \tag{7.10}$$

we employed a simple fully-connected feed-forward nonlinear regression network with 4 hidden layers, each containing 392 units. Aside from the final output layer, each layer is followed by rectified linear unit (ReLU) activation. In addition, we preprocessed all inputs through the standardization

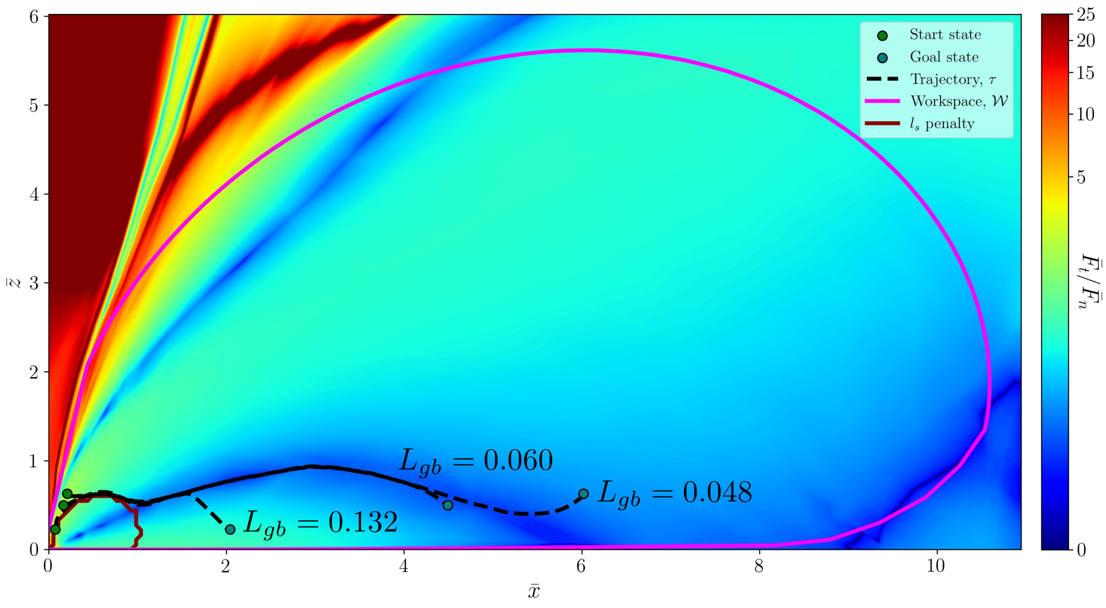
$$\mathbf{x}' = \frac{\mathbf{x} - \bar{\mathbf{x}}_{\mathcal{D}}}{\boldsymbol{\sigma}_{\mathcal{D}}}, \tag{7.11}$$

where $\mathbf{x}$ is the original input, $\bar{\mathbf{x}}_{\mathcal{D}}$ is the mean of the dataset $\mathcal{D}$, and $\boldsymbol{\sigma}_{\mathcal{D}}$ is the standard deviation of $\mathcal{D}$.

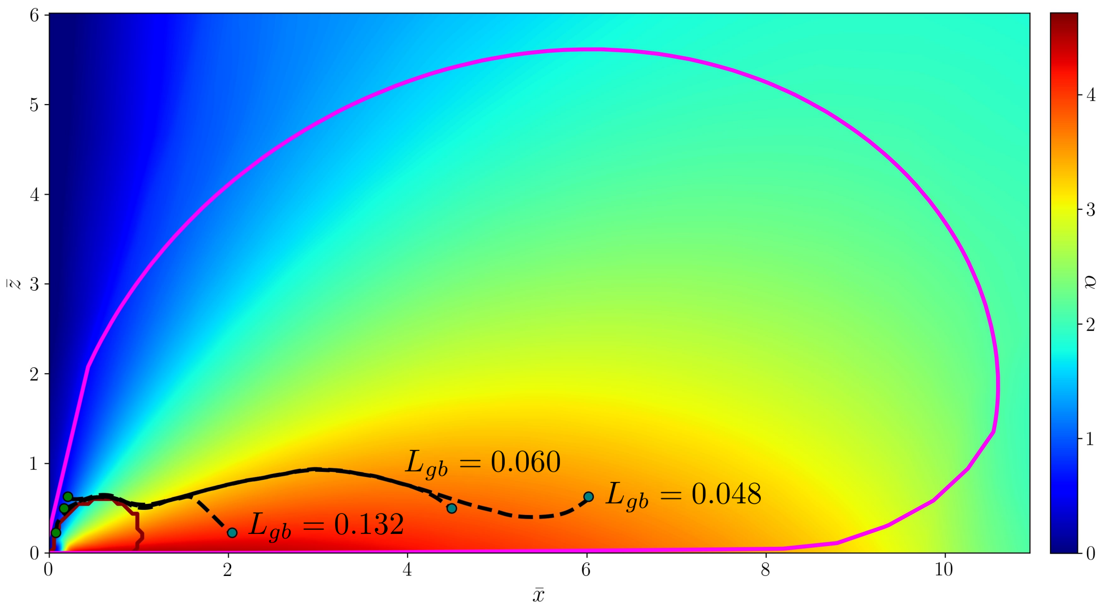
We used an initial 80-20 train-val split on the dataset $\mathcal{D}$ with a batch size of 128. Mean absolute error (MAE) was used as the training error. We alternated between stochastic gradient descent (SGD) and the Adam optimizer whenever training stalled. Furthermore, we gradually increased the batch size up to 4,096 and trained on the entire dataset once the MAE dropped below 0.001. Using this scheme, we achieved an MAE of less than 0.0005.

7.5.3 Constructing the Neural Force Manifold

The neural force manifold (i.e., λ outputs of $\mathcal{F}_{\text{NN}}$ for the workspace set) is discretized into a rectangular grid consisting of $\bar{\delta} \times \bar{\delta}$ blocks, where $\bar{\delta} = \delta / L_{gb}$. For each of the blocks, we obtain and store a single λ value using the midpoint of the block. This results in a discretized neural force manifold $\mathcal{M}$ represented as a $m \times n$ matrix. For the purposes of path planning, we add two components to our manifold. First, we do not allow exploration

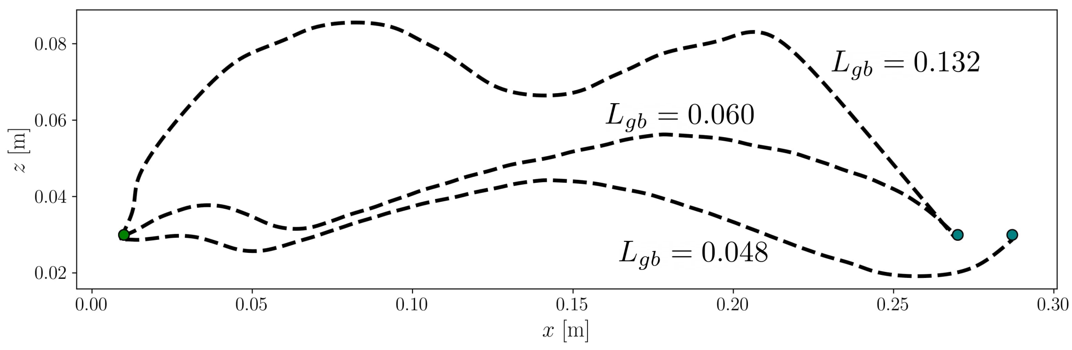


(a) Neural λ Manifold with Generated Trajectories

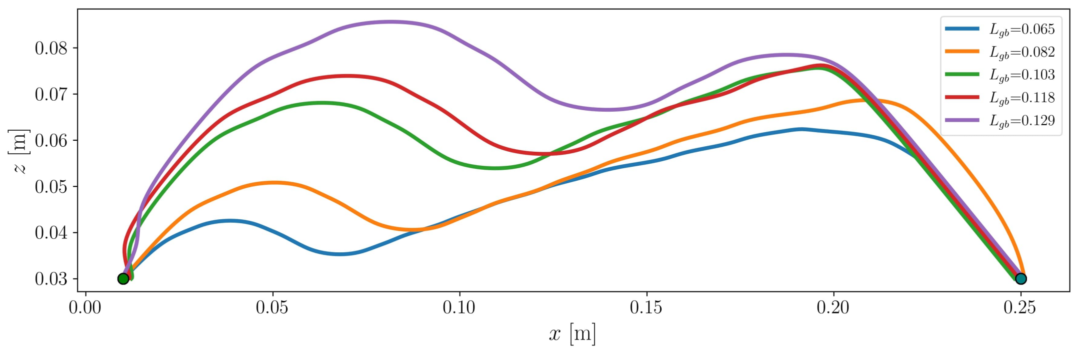


(b) Neural α Manifold with Generated Trajectories

Figure 7.5: Visualization of the trained neural network's non-dimensionalized λ force manifold $\mathcal{M}$ (a) and α manifold (b). An extremely low $\bar{\delta}$ discretization is used to showcase smoothness. For the force manifold, we observe two distinctive local minima canyons. Note that regions outside the workspace $\mathcal{W}$ are physically inaccurate, but are of no consequence as they are ignored. For the α manifold, we observe continuous smooth interpolation throughout, which is crucial for producing feasible trajectories. Both manifolds showcase the trajectories used in the experiments for folding paper in half for $L_{gb} \in [0.048, 0.060, 0.132]$ m.



(a) Redimensionalized Trajectories



(b) Folding Trajectories for Various L_{gb}

Figure 7.6: (a) The three trajectories shown in Figure 7.5 scaled back to real space. These are the actual trajectories used by the robot. (b) Arbitrary trajectories for various L_{gb} with identical start and goal states, highlighting the effect of the material property on our control policy.

into any region not covered by our training dataset ($\bar{l}_s > 6.84$). We do so by defining a workspace $\mathcal{W}$ as all $(\bar{x}, \bar{z})$ pairs within the convex hull of the input portion of the dataset $\mathcal{D}$. Secondly, we also exclude regions within a certain $\bar{l}_s$ threshold. This is done as positions with small suspended lengths and large α angles may result in high curvatures that could cause collision with our gripper and/or plastic deformation, both of which we wish to avoid. We denote this region as the penalty region $\mathcal{L}_s$. Figure 7.5a shows a visualization of $\mathcal{M}$ with the workspace $\mathcal{W}$ and penalty boundary $\mathcal{L}_s$ regions. The α values corresponding to the manifold are also shown in Figure 7.5b.

Algorithm 7: Uniform Cost Search

```
Input: $\bar{x}_s, \bar{z}_s, \bar{x}_g, \bar{z}_g, \mathcal{M}$
Output: $\tau^*$
Func UCS($\bar{x}_s, \bar{z}_s, \bar{x}_g, \bar{z}_g, \mathcal{M}$):
    $\mathcal{W} \leftarrow$ valid workspace of $\mathcal{M}$
    $\mathcal{L}_s \leftarrow l_s$ penalty region
    $\mathbf{h} \leftarrow$ initialize min heap priority queue
    $\mathbf{c} \leftarrow$ initialize empty list
    $n_s \leftarrow$ node with location $(\bar{x}_s, \bar{z}_s)$ and cost 0
    $n_g \leftarrow$ node with location $(\bar{x}_g, \bar{z}_g)$ and cost 0
    $\mathbf{h}$.push($n_s$)
    while len($\mathbf{h}$) $> 0$ do
        $n_i \leftarrow \mathbf{h}$.pop()
        if $n_i == n_g$ then
            $\tau^* \leftarrow$ path from start to goal
            break
        end
        $\mathbf{c}$.append($n_i$)
        for $(\bar{x}_j, \bar{z}_j) \in$ neighbors of $n_i$ do
            if $(\bar{x}_j, \bar{z}_j) \notin \mathcal{W} \setminus \mathcal{L}_s$ then
                continue
            end
            $n_j \leftarrow$ node with location $(\bar{x}_j, \bar{z}_j)$ and cost $\lambda_j$ from $\mathcal{M}$
            if $n_j \in \mathbf{c}$ then
                continue
            end
            if $n_j \in \mathbf{h}$ and cost of $n_j$ is higher then
                continue
            end
            $\mathbf{h}$.push($n_j$)
        end
    end
    $\tau^* \leftarrow$ perform trajectory smoothing on $\tau^*$
    return $\tau^*$
```

7.5.4 Path Planning over the Neural Force Manifold

Given the discretized manifold $\mathcal{M}$, we can now generate optimal trajectories through traditional path planning algorithms. Indeed, we can find that there exist two local minima regions (dark blue in Figure 7.5a) in the neural force manifold $\mathcal{M}$. However, note that these two minima regions are not connected, which means improper local optimization may result in undesired traversal through high force regions later. As mentioned previously, prior mechanics-based work for folding shell-like structures (cloth) have used either physical simulations or energy-based optimization to compute the optimal subsequent grasp based solely on the current status of the manipulated object (Petrík et al., 2016, 2020). We show that this local optimization approach performs poorly for paper folding in Section 7.8. In contrast, we generate globally optimized trajectories that take into account both current and future states of the paper. To do so, we define an optimal trajectory τ^* as one that reaches the goal state while minimizing the sum of λ:

$$\tau^* = \arg\min_{\tau \in \mathcal{T}} \sum_{i=0}^{L-1} \lambda_i, \tag{7.12}$$

where L is the length of the trajectory and $\mathcal{T}$ is the set of all valid trajectories from the desired start to goal state. We define a valid trajectory as one that is contained within the acceptable region

$$(x_i, z_i) \in \mathcal{W} \setminus \mathcal{L}_s \ \forall \ (x_i, z_i) \in \tau, \tag{7.13}$$

and whose consecutive states are adjacent grid locations. Given the discretization of the NFM, we can treat $\mathcal{M}$ as a graph whose edge weights consist of λ. Therefore, we use uniform cost search to obtain τ^*. Alg. 7 provides the pseudocode of the path planning algorithm.

7.6 Robotic System

7.6.1 Dual Manipulator Setup

For our experiments, we use two Rethink Robotics' Sawyer manipulators, as shown in Figure 7.7. One arm has an elongated gripper designed for folding, while the other arm has a spring-compliant roller for creasing and an Intel Realsense D435 camera for visual feedback. The elongated gripper has rubber attached to the insides of the fingers for tight gripping.

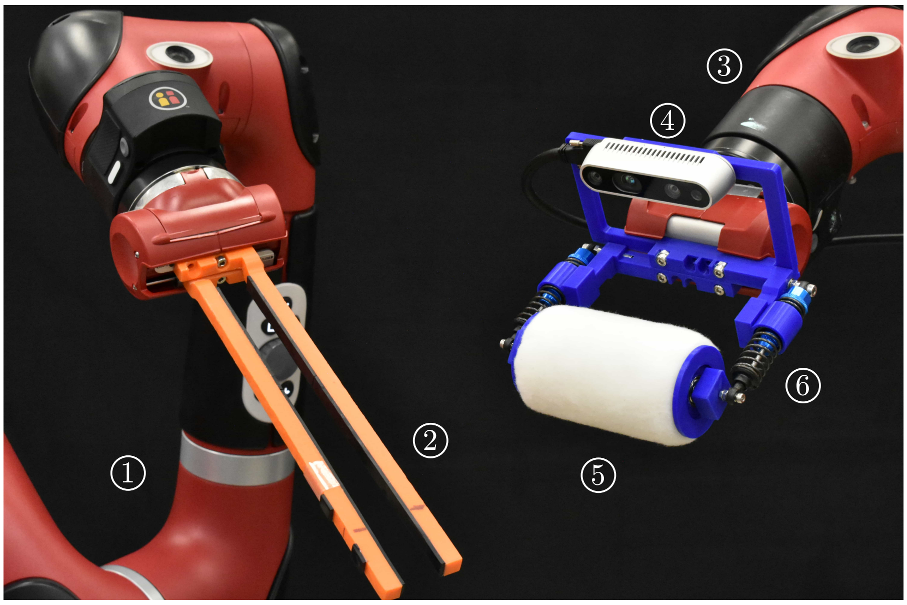


Figure 7.7: Experimental apparatus: Two robot manipulators, one for folding (1) and the other for creasing (3). An elongated gripper (2) is used to grab the manipulated end of the paper. A roller (5) with compliant springs (6) is used to form the crease. An Intel RealSense D435 camera (4) attached to the creasing arm offers visual feedback during the folding procedure. All gripper attachments were 3D printed.

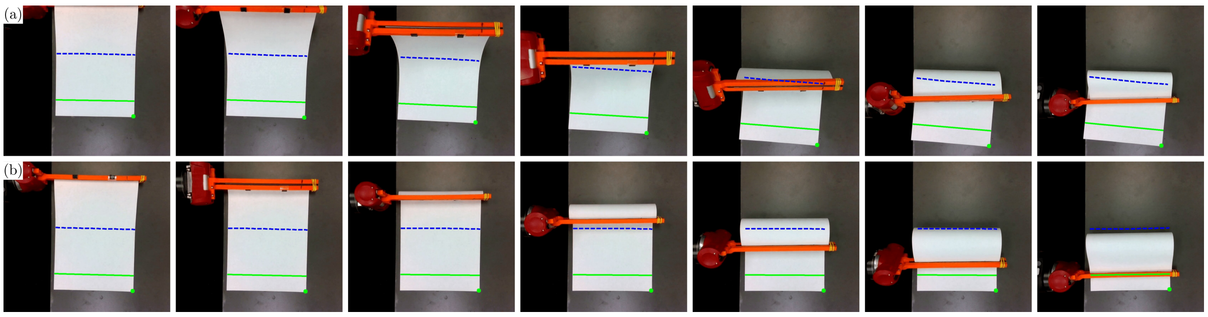


Figure 7.8: Example of our perception system with a top down view of the folding procedure. (a) Shows the intuitive baseline results while (b) shows our open-loop algorithm for $L_{gb} = 0.048\,\text{m}$ and $C = 0.25\,\text{m}$. As in Figure 7.2, the solid green line indicates the desired end effector position while the dashed blue line indicates the crease location. For this case, we observe that the intuitive baseline suffers from considerable sliding while our open-loop algorithm has near-perfect performance.

7.6.2 Perception System

For perception, we take an eye-in-hand approach by attaching an Intel Realsense D435 camera to the roller arm. We do not use the range output of the camera as it points down along the world z-axis and the distance from the camera to the table is known. To determine the pose of the paper, we use simple color detection to segment the paper and then use Shi-Tomasi corner detection (Shi and Tomasi, 1994) to obtain the position of the bottom edge. Figure 7.8 shows an example of the top-down view as well as the poses detected by the vision system.

7.7 Model-Predictive Control via Visual Feedback

Although we minimize λ with our proposed framework, sliding could still occur due to a substrate's low friction surface and/or jittering of the robot's end-effector. Notice that the optimal trajectory τ^* generated as described in Section 7.5.4 assumes that the origin $\mathbf{o}$ of our coordinate system, shown in Figure 7.4a, is fixed. We can define the origin as $\mathbf{o} = \mathbf{q}_0 - l\hat{\mathbf{x}}$, where l is the total length of the paper. Any amount of sliding indicates that $\mathbf{q}_0$ is moving along the x-axis and, therefore, the origin $\mathbf{o}$ also moves an identical amount. When this occurs, our position within the manifold during traversal deviates from the optimal trajectory. Furthermore, without adaptive replanning, the amount of sliding Δx will directly result in Δx amount of error when creasing. To circumvent this, we introduce a model-predictive control approach that mitigates the effects of sliding through trajectory corrections via visual feedback.

We acquire visual feedback at N evenly spaced intervals along the trajectory τ^*, as shown in Figure 7.9. To do so, we first partition τ^* into N partial trajectories. Aside from the first partial trajectory τ_0^*, we extract the start and goal states of the other $1 \leq i \leq N$ partial trajectories resulting in a sequence of N evenly spaced out states $\mathcal{S} = \{(x_1, z_1, \alpha_1), \ldots, (x_N, z_N, \alpha_N)\}$ when accounting for overlaps. After carrying out τ_0^*, we detect the amount of sliding Δx and incorporate this error by updating the start state and non-dimensionalizing as

$$\bar{x}_i^c = \frac{x_i - \Delta x}{L_{gb}}. \tag{7.14}$$

We then replan a partial trajectory τ_i^* from the updated start state (x_i^c, z_i) to the next state (x_{i+1}, z_{i+1}) in the sequence and carry out this updated trajectory. This is repeated until reaching the goal state. By properly accounting for sliding, we ensure that the

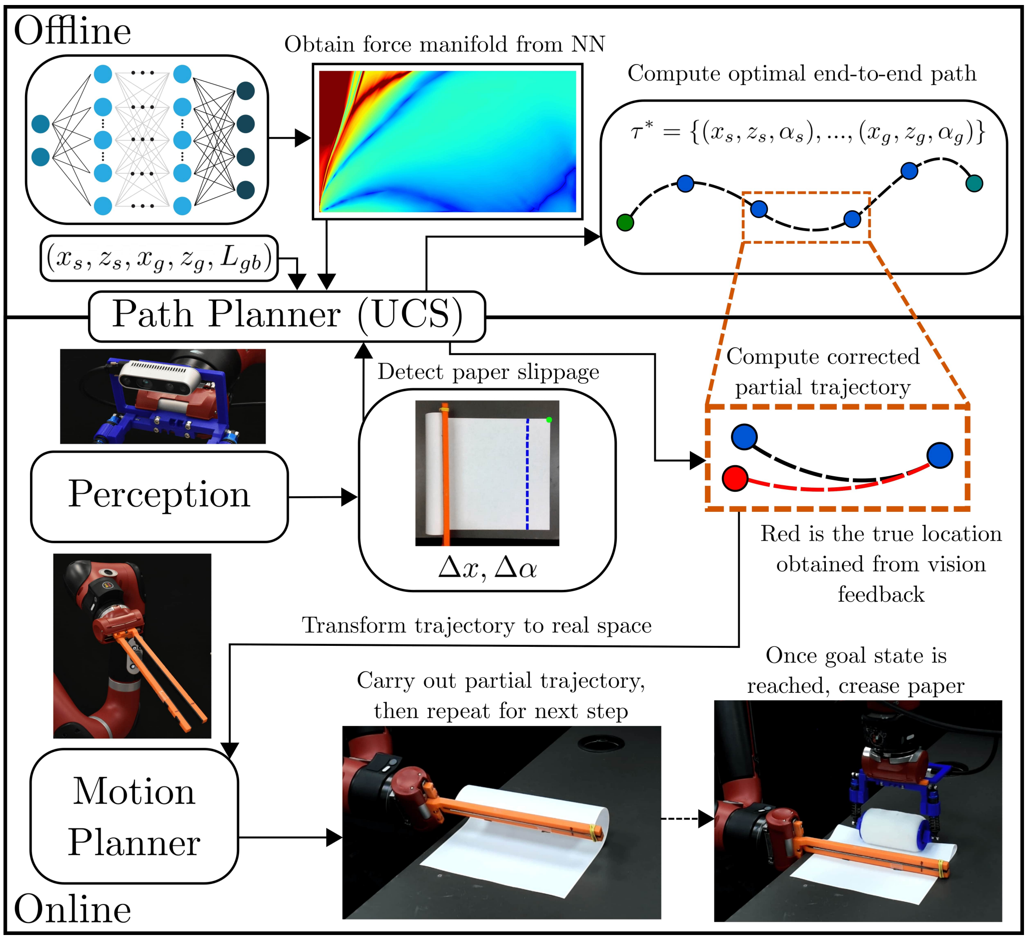

Offline
Obtain force manifold from NN
Compute optimal end-to-end path
$\tau^* = \{(x_s, z_s, \alpha_s), ..., (x_g, z_g, \alpha_g)\}$
$(x_s, z_s, x_g, z_g, L_{gb})$
Path Planner (UCS)
Detect paper slippage
Compute corrected partial trajectory
Perception
$\Delta x, \Delta\alpha$
Red is the true location obtained from vision feedback
Transform trajectory to real space
Once goal state is reached, crease paper
Carry out partial trajectory, then repeat for next step
Motion Planner
Online

Figure 7.9: Overview of our robotic paper-folding pipeline. The top row shows offline components while the bottom row shows online ones. On the offline side, we use our trained neural network to generate the necessary force manifold for planning. Then, given an input tuple $(x_s, z_s, x_g, z_g, L_{gb})$, we generate an end-to-end trajectory using uniform cost search. This end-to-end trajectory is then split up into partial trajectories that are carried out by the robot. At the conclusion of each partial trajectory, we measure paper sliding and replan the next partial trajectory to rectify the error.

traversal through the NFM is as accurate as possible. Note that this scheme allows us to obtain corrected partial trajectories in near real time once N becomes sufficiently large, as each partial trajectory's goal state approaches its start state, allowing for uniform cost search to conclude rapidly. We refer the reader to our supplementary videos* which showcase the speed of the feedback loop.

The sliding Δx is not the only error we must rectify. Recall that we assume an optimal

*See https://github.com/StructuresComp/deep-robotic-paper-folding.

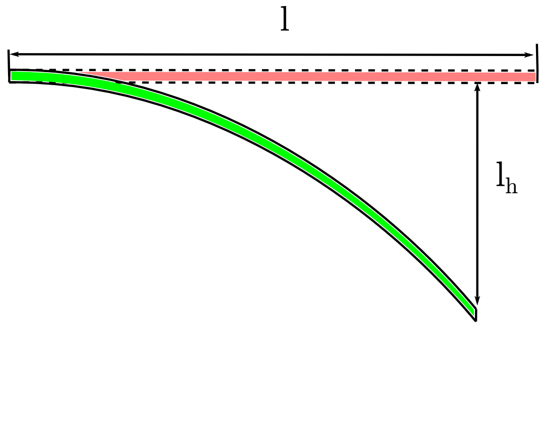
Algorithm 8: Closed-loop Control Pseudocode

```
   Input: $(x_s, z_s), (x_g, z_g), L_{gb}, \delta, N, \mathcal{F}_{\text{NN}}$
1  $\mathcal{M} \leftarrow$ DiscretizeManifold $(\mathcal{F}_{\text{NN}}, \delta)$
2  $\bar{x}_s, \bar{z}_s, \bar{x}_g, \bar{z}_g \leftarrow$ non-dimensionalize with $L_{gb}$
3  $\bar{\tau}^* \leftarrow$ UCS $(\bar{x}_s, \bar{z}_s, \bar{x}_g, \bar{z}_g, \mathcal{M})$                    // Alg. 7
4  update $\bar{\tau}^*$ with $\alpha_s$ using $\mathcal{F}_{\text{NN}}$
5  $\tau^* \leftarrow$ convert $\bar{\tau}^*$ to real space with $L_{gb}$
6  $\tau_0^*, ..., \tau_{N-1}^* \leftarrow$ SplitTrajectory $(\tau^*, N)$
7  $\mathcal{S} \leftarrow$ extract start and goal states
8  carry out $\tau_0^*$ on robot
9  for $(x_i, z_i, \alpha_i)$ and $(x_{i+1}, z_{i+1}, \alpha_{i+1}) \in \mathcal{S}$ do
10     $\Delta x \leftarrow$ detect sliding of paper
11     $x_i^c \leftarrow x_i - \Delta x$
12     $\bar{x}_i^c, \bar{z}_i, \bar{x}_{i+1}, \bar{z}_{i+1} \leftarrow$ non-dimensionalize with $L_{gb}$
13     $\alpha_i^c \leftarrow \mathcal{F}_{\text{NN}}(\bar{x}_i^c, \bar{z}_i)$
14     $\Delta\alpha \leftarrow \alpha_i - \alpha_i^c$
15     $\bar{\tau}_i^* \leftarrow$ UCS $(\bar{x}_i^c, \bar{z}_i, \bar{x}_{i+1}, \bar{z}_{i+1}, \mathcal{M})$
16     $L \leftarrow \text{len}(\bar{\tau}_i^*)$
17     $\boldsymbol{\alpha}_i \leftarrow$ obtain $\alpha$s of $\bar{\tau}_i^*$ using $\mathcal{F}_{\text{NN}}$
18     $\boldsymbol{\alpha}_i^c \leftarrow \boldsymbol{\alpha}_i + \Delta\alpha[1, (L-1)/L, ..., 1/L, 0]^T$
19     append $\bar{\tau}_i^*$ with $\boldsymbol{\alpha}_i^c$
20     $\tau_i^* \leftarrow$ convert $\bar{\tau}^*$ to real space with $L_{gb}$
21     carry out $\tau_i^*$ on robot
22 end
23 crease paper with roller
```

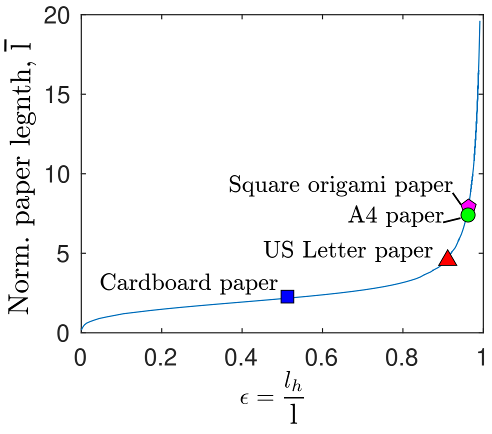
grasp orientation α for each position within the manifold. When the origin of our NFM moves, the true position does not match the intended position, resulting also in an angular error

$$\begin{aligned} \alpha_i^c &= \mathcal{F}_{\text{NN}}(\bar{x}_i^c, \bar{z}_i), \\ \Delta\alpha &= \alpha_i - \alpha_i^c. \end{aligned} \tag{7.15}$$

Simply applying a $-\Delta\alpha$ update to the first point in a partial trajectory results in a large rotational jump that only exacerbates the sliding issue. Furthermore, so long as sliding is not extremely large, the incorrect α at the current position within the manifold is still



(a) Schematic of a hanging plate



(b) Relationship between ϵ vs. $\bar{l}$

Figure 7.10: (a) Schematic of a hanging plate deforming under gravity with fixed left edge. (b) Relationship between the ratio $\epsilon = l_h/l$ and normalized total length of the paper $\bar{l} = l/L_{gb}$ obtained via simulation. The papers used in the experiments can be seen plotted.

fairly optimal. Therefore, the $\Delta\alpha$ error is incorporated into the trajectory gradually:

$$\begin{aligned}
&\tau_i^* = \text{UCS}(\bar{x}_i^c, \bar{z}_i, \bar{x}_{i+1}, \bar{z}_{i+1}, \mathcal{M}), \\
&\boldsymbol{\alpha}_i = \mathcal{F}_{\text{NN}}(\tau_i^*), \\
&\boldsymbol{\alpha}_i^c = \boldsymbol{\alpha}_i + \Delta\alpha[1, (L-1)/L, ..., 1/L, 0]^T,
\end{aligned} \tag{7.16}$$

where UCS denotes uniform cost search and L is the length of trajectory τ_i^*. This gradual correction ensures that we minimize sliding while maintaining smoothness of the trajectory. Alg. 8 provides the pseudocode for our full closed-loop method.

7.8 Experiments and Analysis

7.8.1 Measurement of Material Properties

To use our framework, we must develop a way to accurately measure the parameter L_{gb}. Recall that L_{gb} is composed of the bending stiffness $k_b = Eh^3/12$ and density ρ. Therefore, we need only measure this single quantity to describe the paper's material properties. We

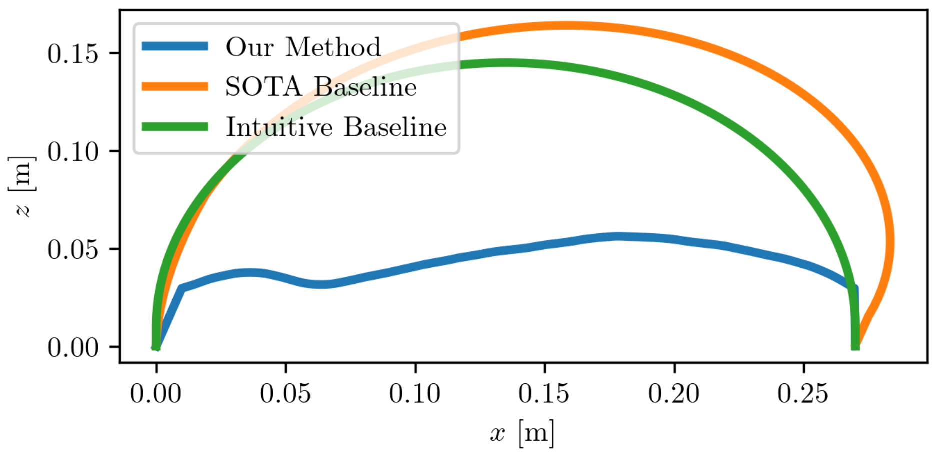


Figure 7.11: Comparison of trajectories computed by the folding algorithms for US letter paper with $\mathcal{C} = 0.27\,\mathrm{m}$. Note the similarity between the SOTA baseline (Petrík et al., 2016) and the intuitive baseline.

next propose a simple way to measure the parameter.

As shown in Figure 7.10a, when one end of the paper is fixed, it will deform due to the coupling of bending and gravitational energy. As L_{gb} encapsulates the influence of bending and gravity on the paper, we have the following mapping:

$$\mathcal{L}(\epsilon) = \bar{l} = \frac{l}{L_{gb}}, \quad \epsilon = \frac{l_h}{l}, \tag{7.17}$$

where l_h is the vertical distance from the free end to the fixed end and l is the total length of the paper. We can obtain the mapping $\mathcal{L}(\epsilon)$ using numerical simulations, which is shown in Figure 7.10b. With this mapping known, simple algebra can be performed to obtain L_{gb}. First, we measure the ratio $\epsilon = l_h/l$ for a particular paper to obtain its corresponding normalized total length $\bar{l}$. Then, the value of L_{gb} can be calculated simply by $L_{gb} = l/\bar{l}$. Once we obtain L_{gb}, we can now use the non-dimensionlized mapping (7.9) to find the optimal path for manipulating the paper.

Table 7.1: Offline trajectory computation time comparison with Petrík et al. (2016, 2020) for various papers and crease types.

Paper Type	Crease Type	Generation Time [s]	
		Petrík et al. (2016, 2020)	Our Method
A4	$\mathcal{C} = 0.25\,\text{m}$	59.46	3.28
A4	$\mathcal{C} = \text{Half}$	51.15	4.13
US Letter	$\mathcal{C} = 0.20\,\text{m}$	68.14	1.80
US Letter	$\mathcal{C} = \text{Half}$	47.30	2.28
Cardboard	$\mathcal{C} = 0.20\,\text{m}$	80.07	1.27
Cardboard	$\mathcal{C} = \text{Half}$	77.28	4.19
Origami	$\mathcal{C} = 0.30\,\text{m}$	43.20	11.73

7.8.2 Baseline Algorithms

To demonstrate the benefits of our folding algorithm, we compared it to both an intuitive and a state-of-the-art baseline. We can think of an intuitive baseline algorithm as one that would work if the opposite end of the paper were fixed to the substrate. Naturally, such a trajectory would be one that grabs the edge of the paper and traces the half perimeter of a circle with radius $R = C/2$:

$$\begin{aligned} \mathrm{d}\theta &= \pi/M, \\ \tau_B &= \{(R\cos(i\mathrm{d}\theta), R\sin(i\mathrm{d}\theta), i\mathrm{d}\theta)\ \forall\ i \in [0, M]\}, \end{aligned} \tag{7.18}$$

where M is an arbitrary number of points used to sample the trajectory. We chose $M = 250$ for all our experiments.

Additionally, we conducted comparisons against the state-of-the-art mechanics-based folding algorithm presented by Petrík et al. (2016, 2020), which we refer to as the "SOTA baseline", that uses a beam model to compute folding trajectories for fabric minimizing sliding. However, this baseline considers only the current status of the deformed material when computing subsequent optimal grasp and, consequently, is unable to handle the challenging task of paper folding. Examples of the computed trajectories are shown in Figure 7.11.

7.8.3 Experimental Setup

We tested folding on 4 different types of paper:

1. A4 paper, $L_{gb} = 0.048\,\text{m}$,
2. US Letter paper, $L_{gb} = 0.060\,\text{m}$,
3. cardboard paper (US Letter dimensions), $L_{gb} = 0.132\,\text{m}$,
4. square origami paper, $L_{gb} = 0.043\,\text{m}$.

For the rectangular papers (1–3), we performed two sets of experiments. The first involved folding the papers to an arbitrary crease location ($C = 0.25\,\text{m}$ for A4 and $C = 0.20\,\text{m}$ for US Letter and cardboard), while the second involves folding the papers in half. For the square origami paper, we chose an arbitrary crease location of $C = 0.30\,\text{m}$. This resulted in a total of 7 folding scenarios. For each of the scenarios, we conducted experiments using 4 different algorithms—the intuitive baseline, the SOTA baseline, our open-loop approach, and our closed-loop approach. We completed 10 trials for each of these algorithms, resulting in a total of 280 experiments.

We also validated our model's non-dependence on the paper's width w (Section 7.3) through additional experiments for narrow strip folding. We created strips of width $w = 2.5\,\text{cm}$ for both A4 and cardboard and performed 10 trials for each algorithm, resulting in 80 additional experiments for a total of 360 experiments in our extensive case study.

7.8.4 Metrics

The metrics used for the experiments were the average fold length and the spin error. The average fold length was calculated by simply taking the average of the left and right side lengths up until the crease. The spin error was calculated as the angle θ_{err} that results in the difference between the left and right side lengths. For square papers, the fold length was defined as the perpendicular length from the tip to the crease and the spin error was the angular deviation from this line to the true diagonal.

7.8.5 Parameters

The neural force manifold $\mathcal{M}$ was discretized using $\bar{\delta} = 0.0548$ as we found this discretization to be a good compromise between accuracy and computational speed. All rectangular papers used a penalty region $\mathcal{L}_s$ defined by $\bar{l}_s < 0.958$ while the square paper used one defined by $\bar{l}_s < 1.137$. This discrepancy is due to the fact that the diagonal paper has a smaller yield strength compared to the rectangular paper; i.e., to prevent extremely high curvatures, a larger suspended length $\bar{l}_s$ range must be avoided.

For closed-loop control, we chose to split all trajectories into $N = 5$ intervals regardless of trajectory length. Furthermore, we used a slick (i.e., low friction) table to showcase the robustness of our method. Note that smaller friction coefficients result in a significantly harder manipulation problem due to the lower threshold for sliding. This is made evident by the excessive sliding of the baseline algorithms shown later. Using an empirical method, we conducted measurements to determine the static coefficient of friction between the papers and substrate, yielding an approximate value of $\mu_s = 0.12$. For comparison, the static coefficient of friction for lubricated steel on steel is typically $\mu_s = 0.15$.

7.8.6 Results and Analysis

In Table 7.1 we report the offline trajectory computation times for all experiments using a single Intel i9-9900KF CPU, demonstrating on average a 15$\times$ speed improvement over the SOTA baseline. In Figure 7.12, all experimental results are reported as box plots where we show the achieved fold lengths and spin errors. From the achieved fold lengths, we see significant improvement over the two baselines for all folding scenarios. As expected, the SOTA baseline demonstrates better performance compared to the intuitive method with the exception of cardboard paper where the former fails to fold it at all. Due to the large gap in performance, broken axes are used to properly display the variance of the recorded data. Note that not only do our algorithms achieve significantly better performance on average, the variance of our approach is also much lower as shown by the decreased y-axis resolution after the axis break. We attribute the high variances of the

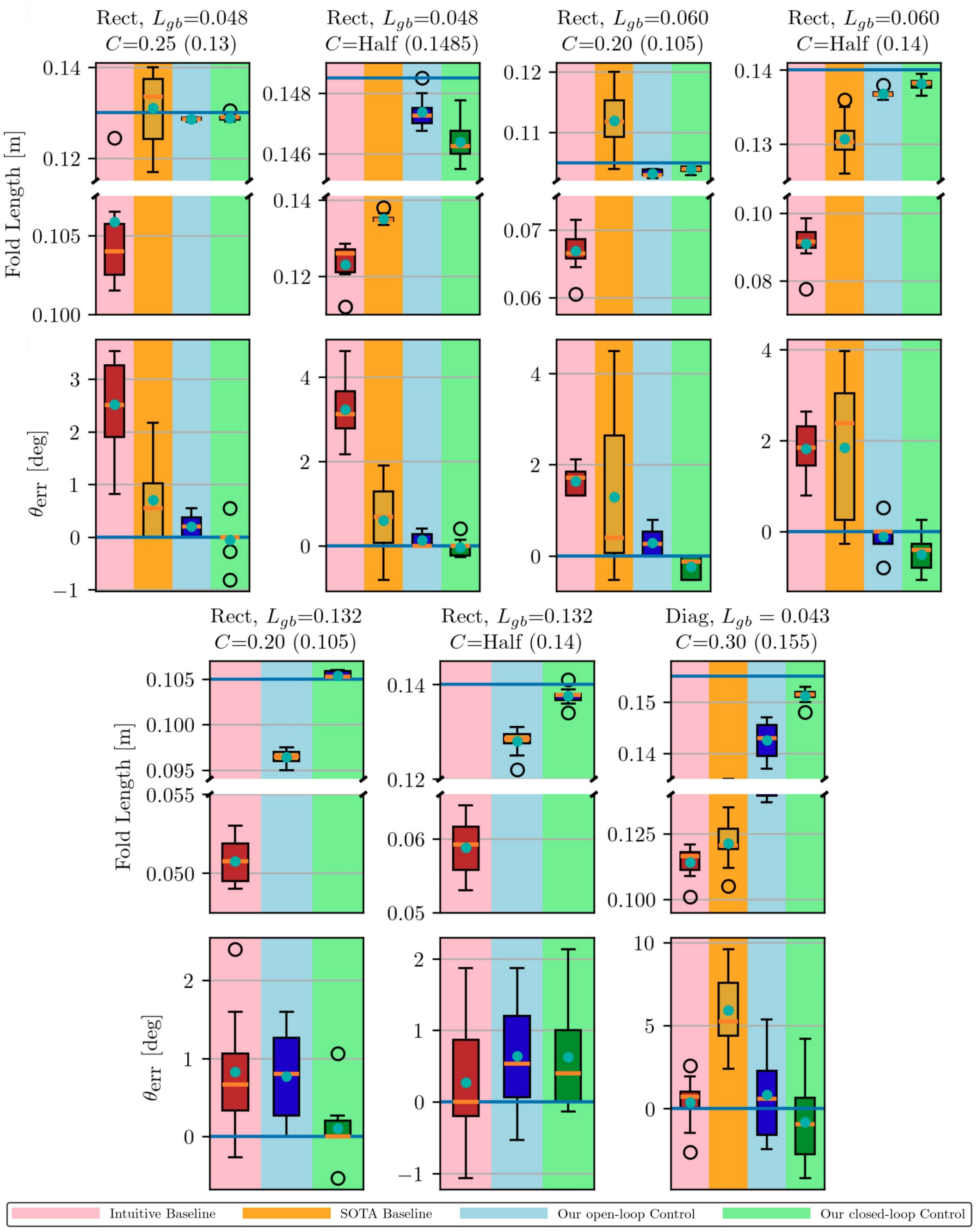


Figure 7.12: Experimental results for all folding scenarios. Each column indicates a folding scenario with the top and bottom plots showing the fold length and spin error results, respectively. Boxplot results are shown color coded for the intuitive baseline, the SOTA baseline (Petrík et al., 2016), open-loop control, and closed-loop control algorithms. Medians are shown as orange lines, means as turquoise circles, and the desired target value as a light blue horizontal line. Both our open-loop and closed-loop algorithms yield significant improvements over the intuitive baseline and the SOTA baseline, as shown by the broken axis for fold length boxplots. Our algorithms also exhibit significantly less variance.

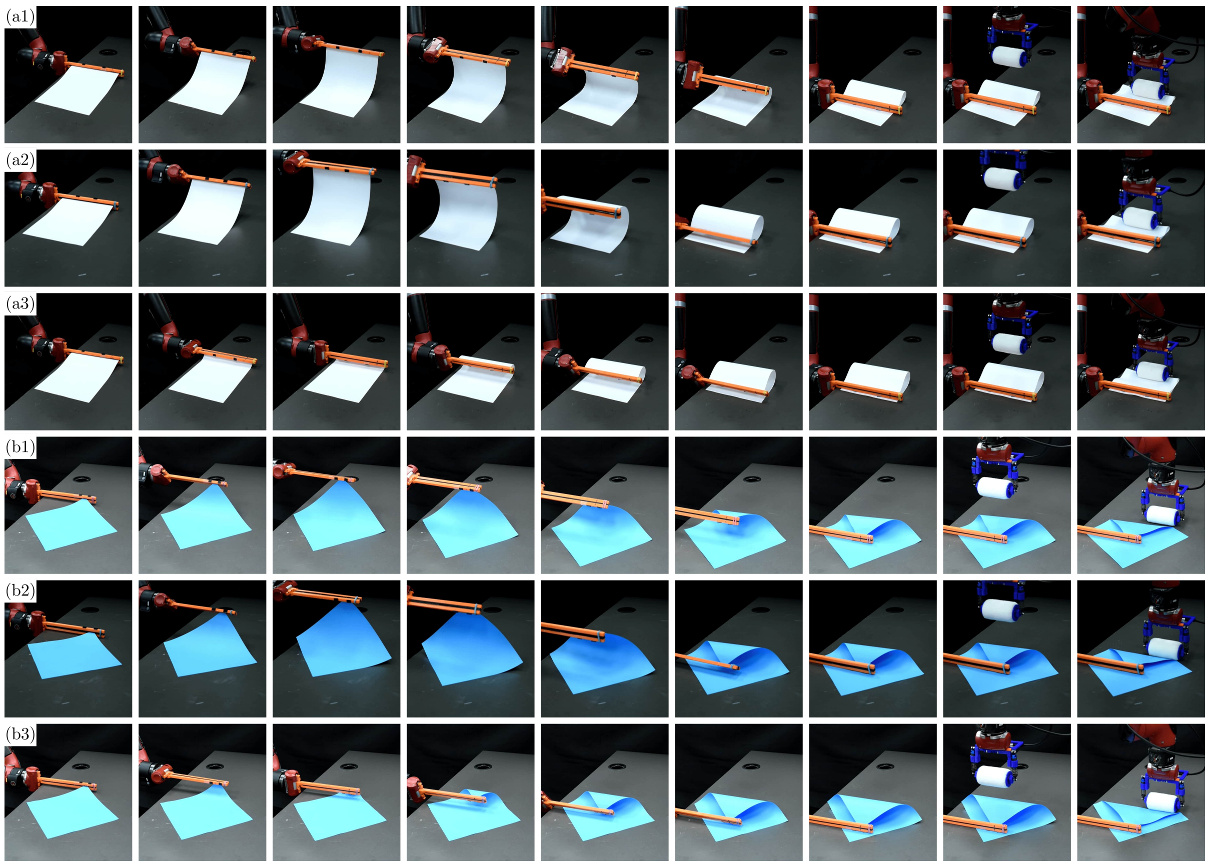


Figure 7.13: Isometric views of different folding scenarios. (a) C = Half folding for $L_{gb} = 0.048\,\text{m}$ paper with the intuitive baseline (a1), the SOTA baseline (a2), and our open-loop algorithm (a3). (b) $C = 0.30\,\text{m}$ diagonal folding for $L_{gb} = 0.043\,\text{m}$ with the intuitive baseline (b1), the SOTA baseline (b2), and our closed-loop algorithm (b3).

baseline algorithms to the increased influence of friction, which can often cause chaotic, unpredictable results. In other words, truly deterministic folding can only be achieved when sliding is nonexistent.

For the vast majority of cases, incorporating visual feedback yields a clear improvement over the open-loop algorithm. Intuitively, we observe a trend where the performance gap between our open-loop and closed-loop algorithms grows as the material stiffness increases for rectangular folding. For softer materials ($L_{gb} = 0.048\,\text{m}$), the open-loop algorithm has near perfect performance as shown when folding a paper in half in Figure 7.13(a3). By comparison, Figure 7.13(a1)–(a2) shows the intuitive and SOTA baselines failing with significant sliding.

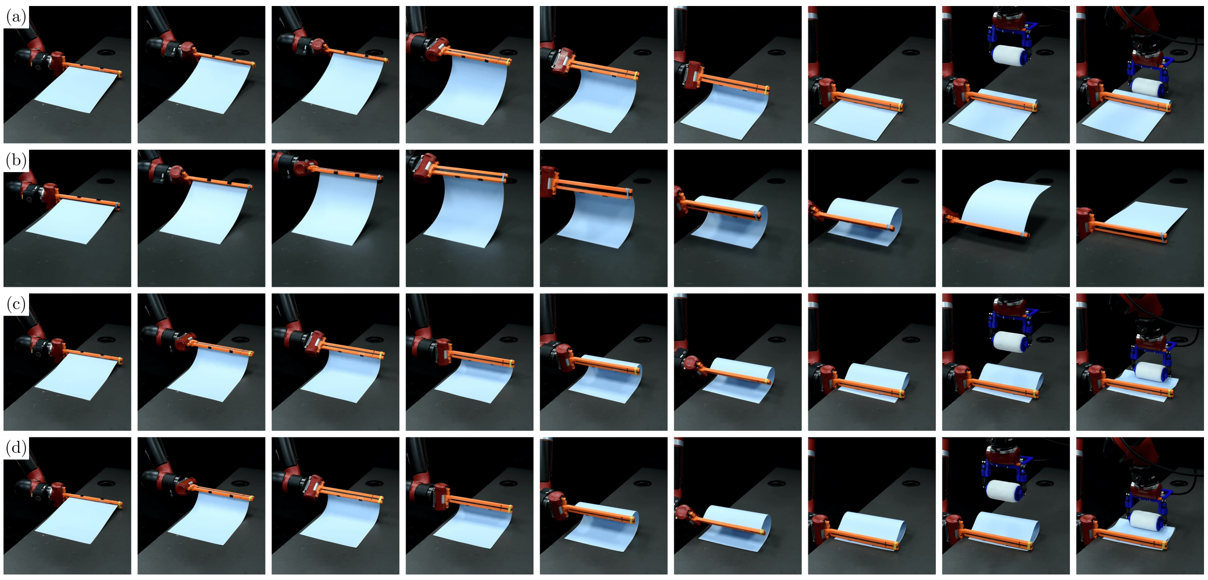


Figure 7.14: Isometric views for folding $C =$ Half with the stiffest paper ($L_{gb} = 0.132\,\mathrm{m}$): (a) shows the intuitive baseline, which fails drastically as the stiffness of the paper causes excessive sliding during the folding process, (b) shows the SOTA baseline, which is unable to fold cardboard at all and experiences a high energy snap caused by the large induced deformations, (c) shows our open-loop algorithm, demonstrating significant improvements over both baselines with minimal sliding, and (d) shows our closed-loop algorithm, which improves upon our open-loop results and achieves near perfect folding.

The sliding problem is only exacerbated by increasing the stiffness of the material ($L_{gb} = 0.132\,\mathrm{m}$). Figure 7.14(a) shows the intuitive baseline algorithm failing to fold the cardboard paper in half by a margin almost as long as the paper itself, while Figure 7.14(b) shows how the SOTA baseline method experiences complete failure due to a high energy snapping caused by excessive deformation. By comparison, our open-loop algorithm is capable of folding the cardboard with significantly better results albeit with some sliding, as shown in Figure 7.14(c). As the material stiffness increases, the benefits of visual feedback are more clearly seen as we are able to achieve near perfect folding for cardboard, as shown in Figure 7.14(d). All of our findings for rectangular folding also match the results of our diagonal folding experiment shown in Figure 7.13(b1)–(b3), where the closed-loop approach once again achieves minimal sliding when compared to the baselines. Overall, the matching findings across all our experiments demonstrate the robustness of our formulation against material and geometric factors.

We observed one oddity for the folding scenario of $L_{gb} = 0.048\,\mathrm{m}$ and $C =$ Half, in

Table 7.2: Fold length results for narrow and wide papers for A4 and cardboard.

Paper ($C/2$)	w [cm]	Fold length mean $\pm\ \sigma$ [cm]			
		INT-base	SOTA-base	Open-loop	Closed-loop
A4 (14.85)	2.5	10.89 ± 0.69	14.66 ± 0.46	14.88 ± 0.07	14.83 ± 0.04
	21	12.31 ± 0.63	13.51 ± 0.13	14.74 ± 0.05	14.64 ± 0.07
CB (14.0)	2.5	7.00 ± 0.45	N/A	13.31 ± 0.14	13.81 ± 0.08
	21	5.88 ± 0.38	N/A	12.8 ± 0.27	13.75 ± 0.19

Table 7.3: Spin error results for narrow and wide papers for A4 and cardboard.

Paper ($C/2$)	w [cm]	Spin error θ_{err} mean $\pm\ \sigma$ [deg]			
		INT-base	SOTA-base	Open-loop	Closed-loop
A4 (14.85)	2.5	0.36 ± 1.65	0.02 ± 0.65	0.48 ± 0.47	-0.23 ± 0.48
	21	6.56 ± 1.29	1.68 ± 0.97	0.27 ± 0.44	0.35 ± 0.38
CB (14.0)	2.5	0.07 ± 0.75	N/A	-0.70 ± 0.31	-0.35 ± 0.82
	21	1.25 ± 1.43	N/A	1.38 ± 1.06	1.45 ± 1.48

which the open-loop algorithm outperformed our closed-loop variant, but the decrease in performance is on average only 1 mm, which is attributable to repetitive discretization error caused by $N = 5$ replanning. In fact, as we use a discretization of $\delta = 2$ mm for the manifold, compounding rounding errors can easily cause 1–2 mm errors. With this in mind, our closed-loop method achieves an average fold length performance within a 1-2 mm tolerance across all experiments.

In terms of spin error, we found that softer materials had the greatest error. As the surface of the table is not perfectly flat, any amount of sliding will directly result in uneven spin, as shown in Figure 7.13(a). As the material stiffness increases, the spin errors became more uniform across the methods as the influence of friction is not enough to deform the paper. Nevertheless, we can see that our open and closed-loop algorithms resulted in less sliding than the baseline on average.

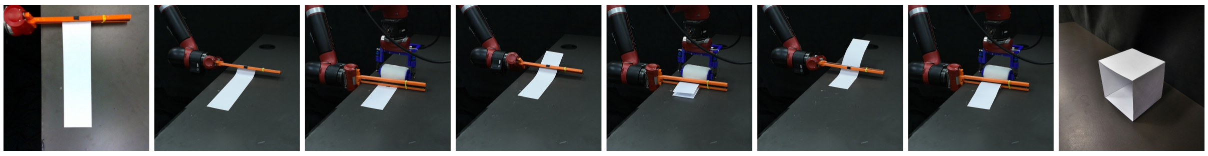


Figure 7.15: Folding a simple origami cube with open-loop control. The left image shows the starting strip of paper (A4), the middle images show three folding steps to create each corner, and the right image shows the resulting cube. Note that regrasps were performed manually.

7.8.7 Effects of Paper Width on Folding Performance

Previously, we mathematically deduced that the paper's width w should have no influence on our folding scheme (Section 7.3). We now validate this claim through experiments. Tables 7.2 and 7.3 report comparisons of fold length and spin error between narrow strips and wide sheets of paper error for half folding A4 paper and cardboard. As expected, both our open-loop and closed-loop methods have near identical performance regardless of paper width. Aside from a slight exception for open-loop narrow cardboard folding that actually benefits from a 5 mm improvement, differences are almost imperceptible to the human eye. Interestingly, both baseline methods experience significant changes in fold length ($> \pm 1$ cm) when width is changed. The prevention of such nondeterministic behavior is another benefit of our method.

7.9 Additional Discussion

7.9.1 Performing Multiple Folds on the Same Paper

Our optimal folding strategy can also be used to fold a single piece of paper multiple times so long as our assumptions regarding material homogeneity and symmetrical centerline hold. Given our method's exceptional accuracy, a precise equilateral origami cube can be created using solely open-loop control, and excellent results are shown in Figure 7.15. Future work pertaining to performing arbitrary folds is discussed in Section 7.10.

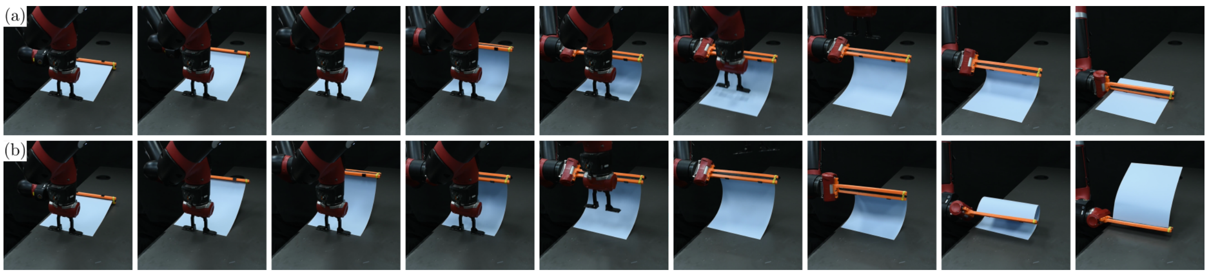


Figure 7.16: Isometric views for folding $C =$ Half with the stiffest paper ($L_{gb} = 0.132\,\mathrm{m}$) using an auxiliary manipulator: (a) shows the intuitive baseline while (b) shows the SOTA baseline. Note the similarity in results to Figure 7.14 despite the use of an auxiliary manipulator to prevent sliding.

7.9.2 Importance of Single Manipulator Folding

To emphasize the significance of single-manipulator folding, we next present results for half-folding cardboard using both baselines with the incorporation of an auxiliary manipulator used to prevent the opposing edge from sliding (Figure 7.16). The auxiliary manipulator "holds down" the edge during the folding process and then moves out of the way at the last possible point in the folding trajectory in order to avoid collision. Despite the additional hardware, we can see that the results for both baselines are near identical to the original experiments shown in Figure 7.14.

For the intuitive baseline, significant sliding occurs as soon as the auxiliary manipulator lifts away due to the paper's high energy state, ultimately yielding an end result that is nowhere near the desired half-fold. This sliding is less noticeable for the SOTA baseline, but the sheet still suffers from buckling. With this demonstration, we conclude that for stiff materials, naive folding strategies can fail drastically despite the use of auxiliary manipulators to explicitly prevent sliding. In fact, the use of auxiliary manipulators is a rather expensive approach, both in terms of requiring additional hardware as well as introducing more overall motions into the pipeline, which our folding strategy completely avoids with its inherent sliding prevention.

7.10 Conclusion

In this chapter, we have introduced a novel sim2real robot control strategy capable of robustly folding sheets of paper of varying materials and geometries along symmetrical centerlines with only a single manipulator. Our framework incorporates a combination of techniques spanning several disciplines, including physical simulation, machine learning, scaling analysis, and path planning. The effectiveness of the framework was demonstrated through extensive real world experiments against both natural and state-of-the-art paper folding strategies. Furthermore, an efficient, near real-time visual feedback algorithm was implemented that further minimizes folding error. Our closed-loop model-predictive control algorithm successfully accomplished challenging scenarios such as folding stiff cardboard with highly consistent accuracy.

In future work, we hope to tackle the difficult problem of creating arbitrary creases along sheets of paper with non-symmetric centerlines. Such non-symmetric paper sheets can no longer be represented as a reduced-order 2D elastic rod model, hence requiring a more sophisticated shell-based formulation. Additionally, precisely folding paper with preexisting creases and folds will be a crucial step to accomplishing elaborate folding tasks, such as robotic origami. We believe that our optimal symmetrical folding trajectories can serve as a valuable "seed" or initial guess when optimizing for asymmetrical folds. Moving forward, we anticipate exploring solutions that take advantage of generalized problem formulations with data-driven control schemes, such as reinforcement learning.

CHAPTER 8

mBEST: Physics-Inspired Realtime Perception for DLOs

Note: This perception algorithm was developed by "accident" when I was originally trying to find a working DLO perception module for robotic knot tying experiments (Chapter 9). It was to my surprise that most state-of-the-art DLO perception algorithms performed poorly and were often needlessly complicated (e.g., using complex blackbox neural networks to perform instance segmentations at intersections and tangles). Given my extensive experience with the DER algorithm, it dawned on me that minimizing bending energy may be a cheap and robust way of performing DLO instance segmentation. And thus, mBEST was born. In this chapter, algorithm names will be italicized for clarity.

8.1 Introduction

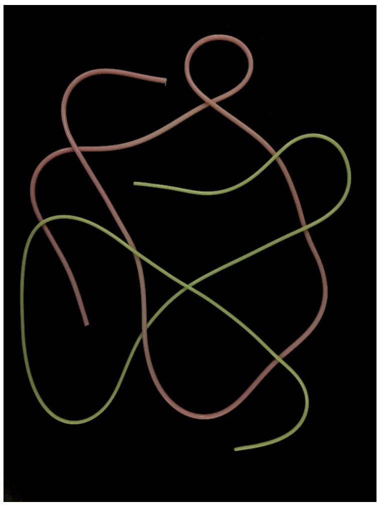
Among various deformable objects, deformable linear objects (DLOs) — typically referred to as "rods" by the mechanics community — are a special group, including everyday objects such as cables, ropes, tubes, and threads. Due to their distinctive geometric characteristic (width $\approx$ height $\ll$ length), DLOs are widely used in various domestic and industrial applications, including surgical suturing (Schulman et al., 2013), knot fastening (Mayer et al., 2008; Choi et al., 2021), cable manipulation (Zhu et al., 2018; Yu et al., 2022b), food manipulation (Iqbal et al., 2017), mechanics analysis (Tong et al., 2021), and more. Because of their flexibility, DLOs are often prone to complex tangling, which complicates manipulation. Additionally, the complicated structures made by DLOs usually have unique topology-induced mechanical properties (Audoly et al., 2007; Jawed et al., 2015a; Patil et al., 2020; Johanns et al., 2021; Sano et al., 2022) and are, therefore, used to tie knots for sailing, fishing, climbing, and various other engineering applications. Given all the aforementioned, a robust, efficient, and accurate perception algorithm for DLOs is crucial to both deformable material manipulation and soft robotics.

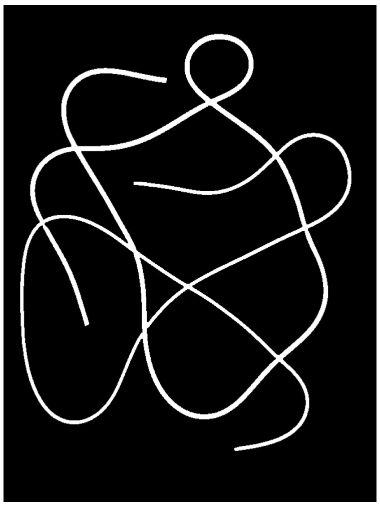
In this chapter, we present an algorithm for robust, accurate, and fast instance segmentation of DLOs, named *mBEST* (Minimal Bending Energy Skeleton pixel Traversals). Without any prior knowledge regarding the geometries, colors, and total number of DLOs, *mBEST* takes a raw RGB image as input and outputs a series of ordered pixels defining the centerline of each individual DLO in the image, thus allowing for the configurations of different DLOs to be easily incorporated into motion planning and manipulation schemes.

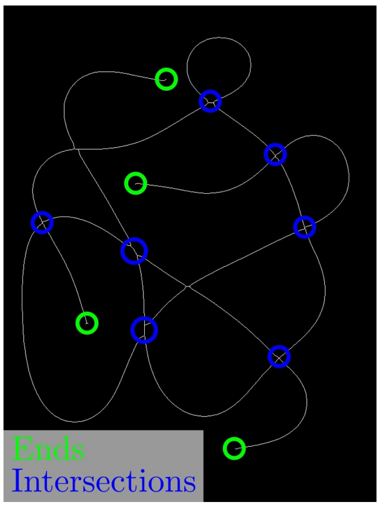
To achieve instance segmentation of DLOs in images, we implement the following sequence of processing steps: Like previous work (Caporali et al., 2022a), we first perform semantic segmentation to produce a binary mask of the DLOs against the background using either simple color filtering methods or a Deep Convolutional Neural Network (DCNN). After a binary mask is obtained, we apply a thinning algorithm to the mask to produce a single-pixel-wide skeleton representation of the DLOs, which preserves the connectivity and centerlines of the binary mask. Thus, key points such as ends and intersections are easily detected. After a series of refinement steps to ensure topological correctness, the skeleton is then traversed, one end at a time, in a manner that minimizes the cumulative bending energy of the DLOs, until another end is encountered. Each traversal yields a single DLO's centerline pixel coordinates, which optionally can then be used to produce segmentation masks. Figure 8.1 overviews the *mBEST* processing pipeline.

Overall, our main contributions in this chapter are that we

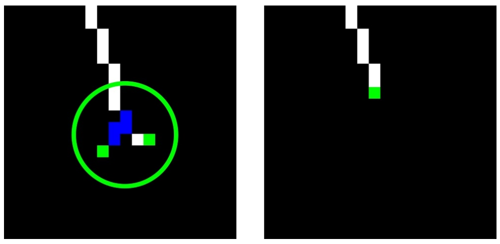
1. develop a robust pipeline for obtaining ordered centerline coordinates and segmentation masks of DLOs from semantic binary masks;
2. demonstrate that the relatively simple and physically meaningful optimization objective of minimizing cumulative bending energy outperforms several state of the

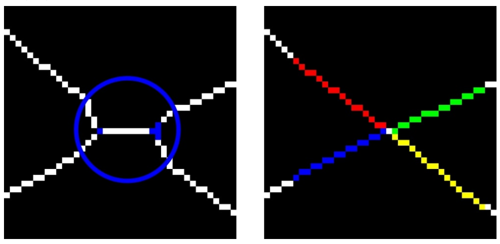

(a) Original Image

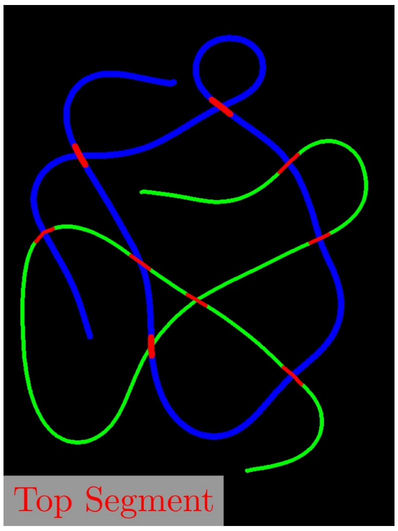

(b) Binary Mask



(c) Skeleton Pixels & Keypoint Detection


(d) Skeleton Refinement: Split Ends


(e) Skeleton Refinement: Intersections



(f) Path Generation

Figure 8.1: Overview of the *mBEST* processing pipeline. An input image (a) is converted to a binary mask (b) using a segmentation method. The binary mask is then converted to a skeleton pixel representation (c), where the connectivity and centerlines of the DLOs are preserved as a single-pixel wide structure and keypoints, such as intersections and ends, are detected. This is followed by a series of refinement steps to maintain the topological correctness of the skeleton: split ends (d) are pruned and pixels representing a single topological intersection (e) are clustered, matched, and replaced with a more intuitive intersection. Finally, the DLOs are delineated (f) by traversing skeleton pixels and choosing minimal cumulative bending energy paths.

art (SOTA) algorithms;

3. showcase the effectiveness of our topology-correcting skeleton refinement steps by outperforming the SOTA algorithms with a hybrid *mBEST* formulation that uses the intersection handling scheme of SOTA algorithms;
4. achieve faster, real-time performance compared to the SOTA algorithms.

Moreover, we have released all our source code, datasets (with ground truth), and a supplementary video.*

8.2 Methodology

The *mBEST* algorithm consists of the following steps:

1. DLO Segmentation
2. Skeletonization
3. Keypoint Detection
4. Split End Pruning
5. Intersection Clustering, Matching, and Replacement
6. Minimal Bending Energy Path Generation
7. Crossing Order Determination

The following sections describe each step in detail.

8.2.1 DLO Segmentation

The first step in detecting the DLOs is to obtain a binary mask $\mathbf{M}_{\text{dlo}}$ of the image that distinguishes all DLO-related pixels from the background. The initial image segmentation method is not a key contribution of *mBEST*. Rather, it is a modular component of our pipeline, allowing for different methods to be plugged in depending on the use case. As stated previously, we employ two semantic segmentation methods: a DCNN segmentation model and color filtering. In particular, we use *FASTDLO*'s pretrained DCNN model

* See `https://github.com/StructuresComp/mBEST`.

(Caporali et al., 2022a) in our experiments.

8.2.2 Skeletonization

As shown in Figures 8.1b–8.1c, the next step of our algorithm is to convert $\mathbf{M}_{\text{dlo}}$ to a skeleton mask $\mathbf{M}_{\text{sk}}$, which is useful as both the connectivity and general topology of the DLOs are maintained. Furthermore, as segments are only 1 pixel wide, traversals along segments are not susceptible to path ambiguity. To achieve skeletonization, we use an efficient thinning algorithm designed specifically for 2D images, and refer the reader to Zhang and Suen (1984) for the details.

8.2.3 Keypoint Detection

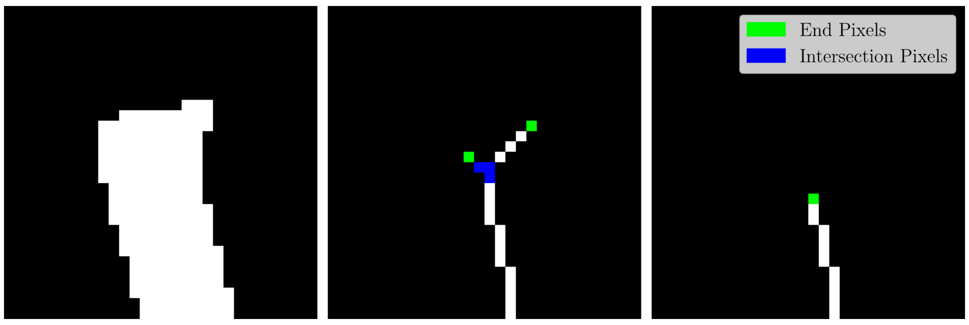
After obtaining the skeleton pixel representation, we can then detect two types of key points: ends and intersections. Locating ends is crucial since they serve as the start and finish points for skeleton pixel traversals. Locating intersections is crucial as they represent the only points at which a pixel traversal will have multiple possible routes. Therefore, care must be taken in choosing the correct path when passing through an intersection.

To detect ends and intersections, a skeleton pixel classification kernel,

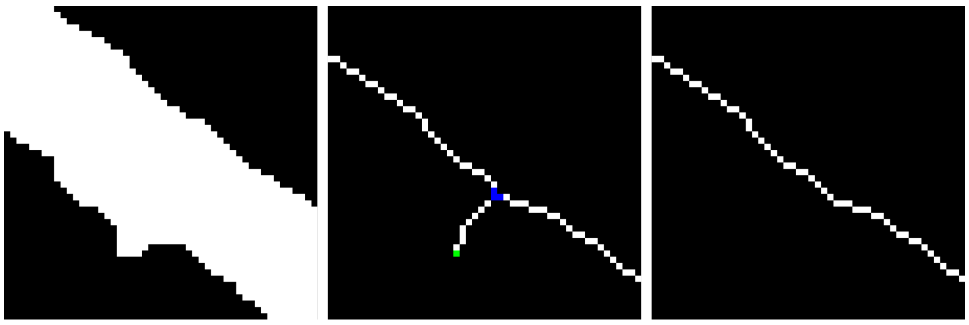
$$\mathbf{K} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

is convolved with the skeleton mask; i.e., $\mathbf{M}_{\text{sk}} \circledast \mathbf{K}$. We then identify all end pixels $\mathbf{E}$ as those with a value of 11 (1 neighbor) and all intersection pixels $\mathbf{I}$ as those with a value greater than 12 (3 or more neighbors).

After obtaining both $\mathbf{E}$ and $\mathbf{I}$, additional work must be done to obtain the correct representative sets. For example, end pixels that are unindicative of a topological end may be produced from a noisy binary mask. These "split ends" will then falsely produce



(a) Split End at DLO End



(b) Split End along DLO Segment

Figure 8.2: Examples of split ends that may occur during the skeletonization process. Row (a) shows split ends that may occur at an actual topological end, while Row (b) shows a split end along a segment produced by a jagged mask. For both examples, the first column shows the binary mask; the second shows the split end after skeletonization, and the third shows the topologically correct structure after pruning.

intersection pixels themselves, as shown in Figure 8.2. Additionally, a single topological intersection will result in either two Y-shaped divides or a single X-shaped divide, as shown in Figure 8.4. Such pixels must be clustered accordingly, with a single point of intersection determined. In the case of a skeleton possessing two Y-shaped divides in the context of a single intersection, the intersection must also be replaced with an X-shaped divide that more accurately represents the centerlines of the DLOs.

8.2.4 Split End Pruning

When the boundary of the binary mask $\mathbf{M}_{\text{dlo}}$ is jagged, the skeleton mask $\mathbf{M}_{\text{sk}}$ may contain several types of split ends, as shown in Figure 8.2. Such split ends must be identified and pruned as they do not accurately represent the topology of the DLO(s) and will result in incorrect start points as well as cause path ambiguity during pixel traversals.

Note that the length of a split end can be at most the radius of the DLO from which it is sprouting. Therefore, the length of all split ends should be within a threshold δ much less than the length of the DLO. As such, for every end in $\mathbf{E}$, we traverse along its segment until one of the following three conditions occurs before traversing δ pixels:

1. an intersection is encountered,
2. an end is encountered,
3. or neither was encountered.

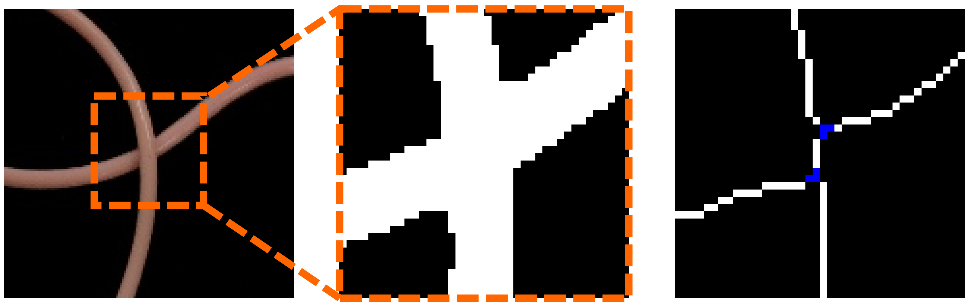
For Conditions 1 and 2, we remove the segment that was just traversed from $\mathbf{M}_{\text{sk}}$ as well as the corresponding end from $\mathbf{E}$. For Condition 1, we must also remove from $\mathbf{I}$ all intersection pixels that were produced from the pruned split end. For any endpoint that satisfies Condition 3, we do nothing.

To encompass all possible split ends, we can set δ to be the diameter of the widest DLO in the image. Radii of the DLOs can be obtained by computing an L2 distance transform on $\mathbf{M}_{\text{dlo}}$, which results in a matrix $\mathbf{D}$ containing for each pixel location the closest Euclidean distance to a 0-value pixel. With this, we can then simply set $\delta = 2\max(\mathbf{D})$. As the distance matrix $\mathbf{D}$ tells us the radii information for all centerline points, we can reuse it to generate segmentation masks once each DLO's path is ascertained (Caporali et al., 2022a).

8.2.5 Intersection Clustering, Matching, and Replacement

As mentioned in Section 8.2.3, a single topological intersection can result in either a 2Y or X-shaped branching (Figure 8.3). Furthermore, each of these branches may have several intersection pixels; i.e., pixels with 3 or more neighbors. Our goal then is to group each pixel in $\mathbf{I}$ to a single branch and then group each branch to its true topological intersection. With all the intersection pixels properly grouped, we then define a single intersection pixel that represents the true center of a crossing, for all crossings.

First, to cluster all adjacent intersection pixels, we use Density-Based Spatial Clustering


(a) 2Y Crossing

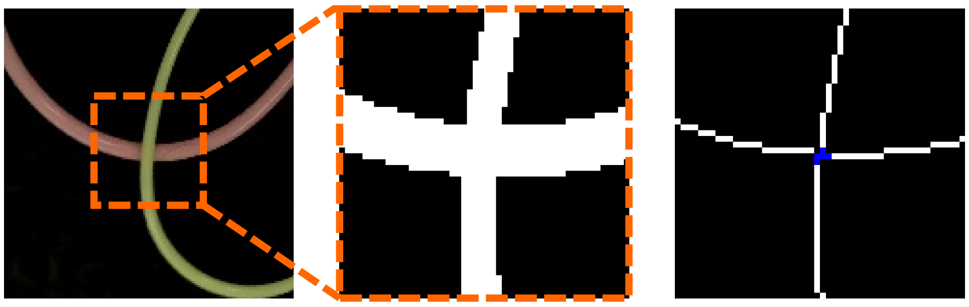
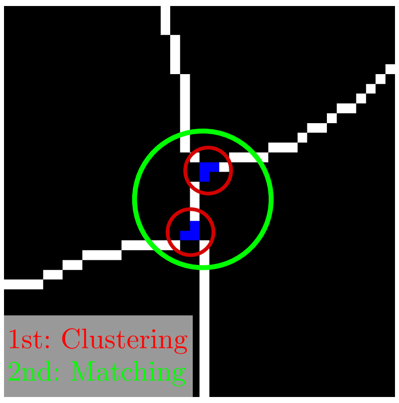

(b) X Crossing

Figure 8.3: Skeleton pixel intersection examples showcasing the two possible scenarios of (a) 2Y branching and (b) X branching. The first column is the original image; the second column is the zoomed binary mask; and the third column is the skeleton pixel representation with intersection pixels highlighted in blue.

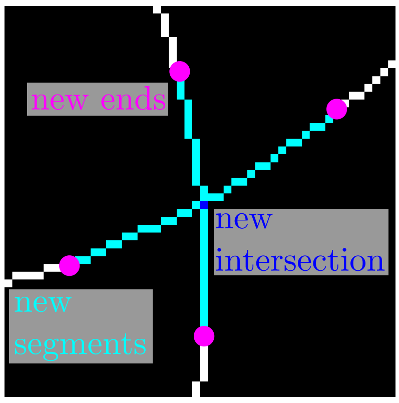
of Applications with Noise (DBSCAN) (Ester et al., 1996), an algorithm that clusters data points within a distance threshold of each other. In our case, the Euclidean pixel distance and is simply set to 2. Once all adjacent pixels in **I** are clustered, each cluster is averaged to create a new **I**.

The next step is to group all branches in **I** by their respective topological intersection. To do so, we first classify all branches in **I** as either Y or X-branches. Intersections that are X-branches are already topologically correct so they are left unmodified in **I**. The remaining Y-branches are removed from **I**, after which we obtain a list of all possible Y-branch pair combinations and sort them by their pair distance. The closest branch pairs are then iteratively popped from the list and matched so long as neither branch has already been matched. A new intersection pixel is then computed from the average of the matched branch locations and then added back to **I**.

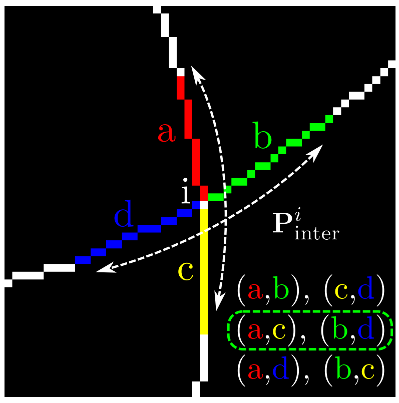
Using the new intersection pixel location, all matched Y-branches can then be replaced



(a) Intersection Grouping



(b) Intersection Replacement



(c) Optimal Path Generation

Figure 8.4: Intersection handling for the 2Y crossing shown in Figure 8.3a. As 2Y-shaped crossings are topologically incorrect, we correct them by replacing the intersection pixels (a) in two stages: the first involves clustering adjacent pixels and the second involves pair matching nearby clusters. Using the centroid location of the matched clusters, we then replace the intersection (b) by creating new ends and having new segments sprout and connect to the centroid. Finally, (c) the new generated ends and segments are used to discover the combination of paths that minimizes the cumulative bending energy of the DLO.

with an X-shaped branch, as shown in Figure 8.4b. Note that there may be cases where an intersection is topologically a Y-branch (i.e., an end perfectly overlaps with a segment) and thus has no corresponding match. To account for these cases, we stop matching Y-branches once the pair distance exceeds a limit $\epsilon = 10\max(\mathbf{D})$ or if every Y-branch has already been matched. Any remaining non-matched Y-branches are added back to $\mathbf{I}$. As shown in Figure 8.4b, we record new "ends" for all topologically correct intersections. This is done so that we know that an intersection is imminent during a pixel traversal and, hence, take the correct precomputed path, as discussed next.

8.2.6 Minimal Bending Energy Path Generation

For rods that have nonuniform curvatures, the bending energy must be computed in a discretized fashion. Recall from (3.5), that if we discretize a rod into N nodes and $N-1$

edges, then the total bending energy (in 2D) is

$$E_b = \frac{1}{2} EI \sum_{k=1}^{N-2} (\kappa_k - \bar{\kappa}_k)^2 \frac{1}{V_k},$$

where EI is the bending stiffness, κ_k and $\bar{\kappa}_k$ are the deformed and undeformed discrete dimensionless curvatures, respectively, at node $k \in [1, N-2]$, and V_k is the Voronoi length. For our DLOs, we assume that the undeformed curvature is a straight configuration ($\bar{\kappa} = 0$). Then, minimizing the bending energy of an elastic rod amounts to minimizing the discrete curvatures.

The norm of the discrete dimensionless curvature for a node k is easily computed using the unit tangent vectors of the adjacent edges from (3.3):

$$\|\kappa_k\| = \left\| \frac{2\mathbf{t}^{k-1} \times \mathbf{t}^k}{1 + \mathbf{t}^{k-1} \cdot \mathbf{t}^k} \right\|, \tag{8.1}$$

where $\mathbf{t}^{k-1}$ and $\mathbf{t}^k$ are the unit tangent vectors of edges $k-1$ and k, respectively.

Note that the only time we must choose between multiple paths is at an intersection, whereas traversals through segments are unambiguous. Using the new ends shown in Figure 8.4b, we can compute the combination of paths that minimizes the cumulative bending energy of the DLOs by simply computing the pairs of segments that minimize cumulative norm curvature. In other words, if an intersection at $\mathbf{i}$ has four end points $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and $\mathbf{d}$, then we must find the pairs of end points $(\mathbf{p}_1^1, \mathbf{p}_1^2)$ and $(\mathbf{p}_2^1, \mathbf{p}_2^2)$ that minimizes $\|\kappa_1\| + \|\kappa_2\|$, where

$$\begin{aligned} \kappa_1 &= \frac{2\mathbf{t}_1^1 \times \mathbf{t}_1^2}{1 + \mathbf{t}_1^1 \cdot \mathbf{t}_1^2}, \quad & \kappa_2 &= \frac{2\mathbf{t}_2^1 \times \mathbf{t}_2^2}{1 + \mathbf{t}_2^1 \cdot \mathbf{t}_2^2}, \\ \mathbf{t}_1^1 &= \frac{\mathbf{i} - \mathbf{p}_1^1}{\|\mathbf{i} - \mathbf{p}_1^1\|}, \quad & \mathbf{t}_1^2 &= \frac{\mathbf{p}_1^2 - \mathbf{i}}{\|\mathbf{p}_1^2 - \mathbf{i}\|}, \\ \mathbf{t}_2^1 &= \frac{\mathbf{i} - \mathbf{p}_2^1}{\|\mathbf{i} - \mathbf{p}_2^1\|}, \quad & \mathbf{t}_2^2 &= \frac{\mathbf{p}_2^2 - \mathbf{i}}{\|\mathbf{p}_2^2 - \mathbf{i}\|}. \end{aligned} \tag{8.2}$$

Figure 8.4c shows an example of this optimization, where out of the 3 possible combinations of paths the one that minimizes total curvature is selected. With the

Algorithm 9: *mBEST* Pipeline Pseudocode

```
Input: $\mathbf{M}_{\text{dlo}}$
Output: $\mathbf{P}$
1  Func mBEST($\mathbf{M}_{\text{dlo}}$):
2      $\mathbf{P} \leftarrow [\,]$
3      $\mathbf{M}_{\text{sk}} \leftarrow$ Skeletonize($\mathbf{M}_{\text{dlo}}$)
4      $\mathbf{D} \leftarrow$ DistTransform($\mathbf{M}_{\text{dlo}}$)
5      $\delta, \epsilon \leftarrow$ ComputeParams($\mathbf{D}$)
6      $\mathbf{E}, \mathbf{I} \leftarrow$ DetectKeyPoints($\mathbf{M}_{\text{sk}} \circledast \mathbf{K}$)
7      $\mathbf{E}, \mathbf{I} \leftarrow$ PruneSplitEnds($\mathbf{E}, \mathbf{I}, \mathbf{M}_{\text{sk}}, \delta$)
8      $\mathbf{I} \leftarrow$ ReplaceIntersections($\mathbf{I}, \epsilon$)
9      $\mathbf{P}_{\text{inter}} \leftarrow$ GenIntersectionPaths($\mathbf{I}, \mathbf{M}_{\text{sk}}$)
10     while $\mathbf{E}$ is not empty do
11         $\mathbf{x} \leftarrow \mathbf{E}$.pop()
12         while True do
13             $\boldsymbol{\tau} \leftarrow$ traverse along $\mathbf{M}_{\text{sk}}$ from $\mathbf{x}$ until reaching an end $\mathbf{e}$
14             if $\mathbf{e} \in \mathbf{P}_{\text{inter}}$ then
15                 $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} + \mathbf{P}^{i}_{\text{inter}}$
16                 $\mathbf{x} \leftarrow$ last pixel of $\mathbf{P}^{i}_{\text{inter}}$
17             end
18             else
19                 $\mathbf{E}$.remove($\mathbf{e}$)
20                 break
21             end
22         end
23         $\mathbf{P}$.append($\boldsymbol{\tau}$)
24     end
25     return $\mathbf{P}$
```

paths through intersections properly precomputed, the skeleton pixel traversals to obtain each DLO's centerline can now take place. Alg. 9 shows the pseudocode of the *mBEST* pipeline.

8.2.7 Crossing Order Determination

The final step of the pipeline involves ascertaining which DLO is resting on top at intersections. To solve this problem, we use a modified version of *FASTDLO*'s (Caporali et al., 2022a) solution. To compute crossing order at intersections, we use the precomputed optimal paths shown in Figure 8.4c. Crossing order is then determined by computing the sum of the standard deviations of the RGB channels of the pixels along each path. Finally, the path that contains the lower sum is assumed to be the one on top. Although this solution from *FASTDLO* works fairly well, we discovered that failures can occur due to glare along the centerline, which may even cause failures for intersections with two completely different colored DLOs. To eliminate the influence of glare, we compute the standard deviations of the intersection path pixels not on the original input image but on its blurred version.

8.3 Experimental Results

8.3.1 Datasets

We used two different datasets to evaluate the effectiveness of *mBEST*. The first consists of relatively simple configurations of DLOs against complex backgrounds, whereas the second consists of complex configurations (i.e., highly varying curvatures and numerous self-loops) of DLOs against a simple black background. We focused mostly on images with a simple black background since the initial binary mask segmentation is not a key aspect of our algorithm; however, *mBEST* also works well for complex backgrounds, as shown in Figures 8.5 and 8.6.

The complex background dataset was provided by Caporali et al. (2023a), and comprises a total of 132 images of size 640 × 360. It is split into tiers C1, C2, and C3, each containing 44 images, where the tier numbers reflect the increasing complexity of the background. Given the complexity of the background, DCNN segmentation was used to obtain the initial binary mask. We removed two images each from C2 and C3 as they included intersections involving > 2 DLOs, scenarios which are currently outside the scope of *mBEST*.

The simple background dataset consists of a total of 300 images of size 896 × 672 and is split into tiers S1, S2, and S3, each containing 100 images, where the tier numbers reflect the number of DLOs in the image, resulting in both an increase in complexity and

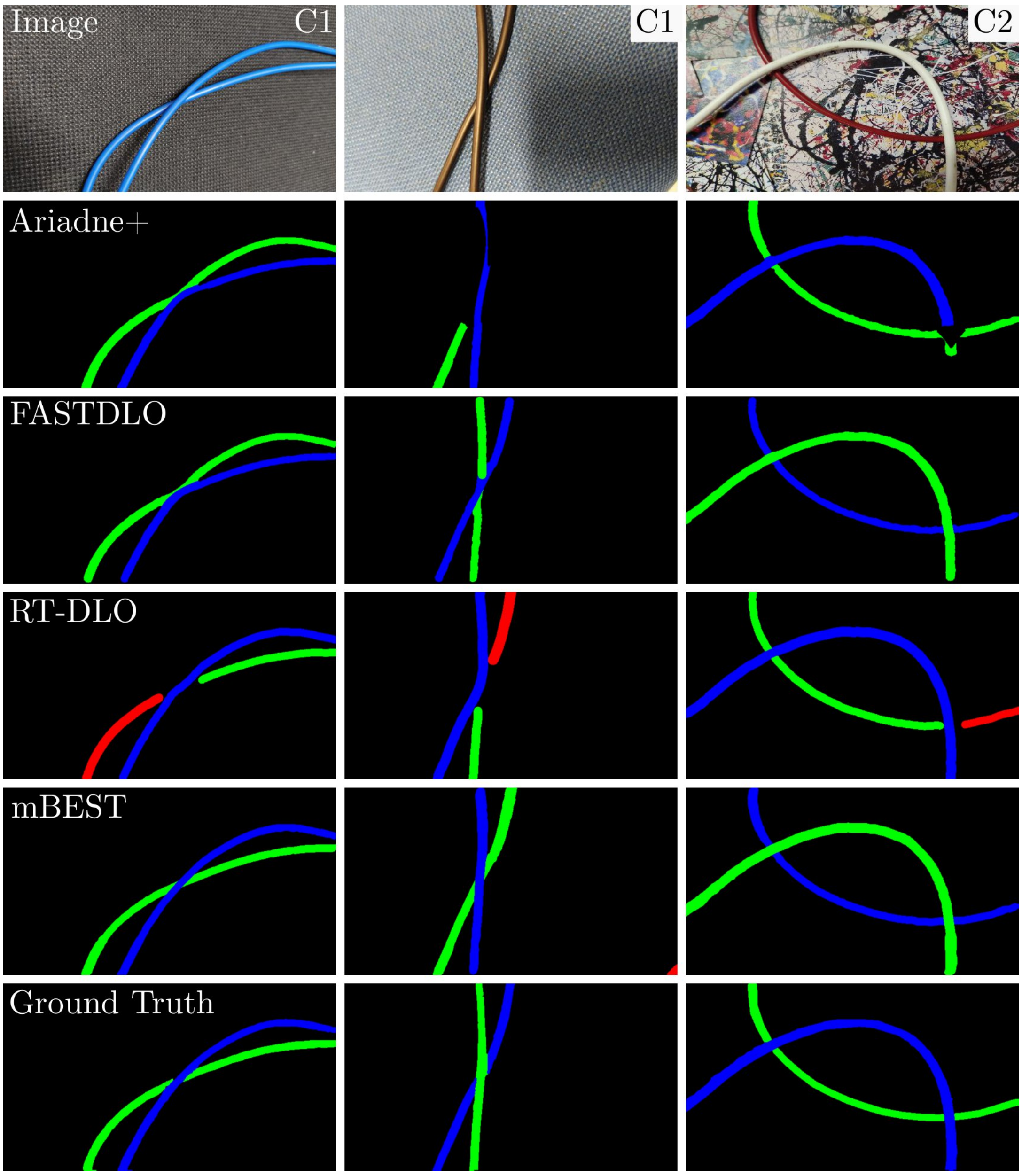


Figure 8.5: Sample segmentations for images in datasets C1 and C2. Each column shows segmentation results for a different image with the top row indicating the dataset to which the image belongs. Rows 2–5 show *Ariadne+*, *FASTDLO*, *RT-DLO*, and *mBEST* results, respectively. The bottom row shows the ground truth. Note the failure to properly handle intersections for all baseline algorithms, especially when strands are nearly parallel.

computational demand as the numbers increase. Given the high contrast background, color filtering sufficed to obtain the initial binary mask.

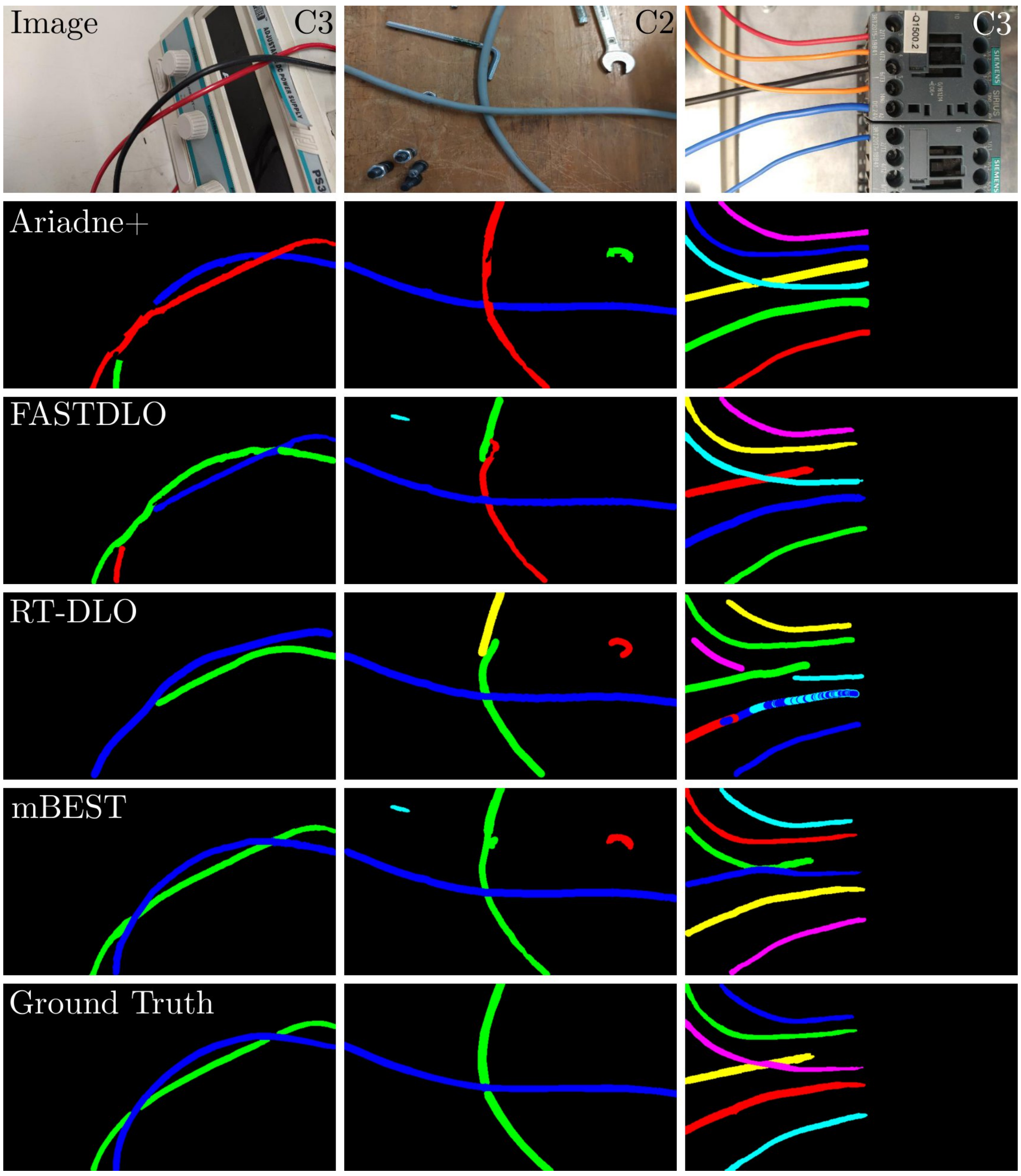


Figure 8.6: Sample segmentations for images in datasets C2 and C3. Each column shows segmentation results for a different image with the top row indicating the dataset to which the image belongs. Rows 2–5 show *Ariadne+*, *FASTDLO*, *RT-DLO*, and *mBEST* results, respectively. The bottom row shows the ground truth. Note the failure to properly handle intersections for all baseline algorithms, especially when strands are nearly parallel. In fact, *RT-DLO* can be seen to produce an unintuitive output for the last example where certain wires are labeled multiple times.

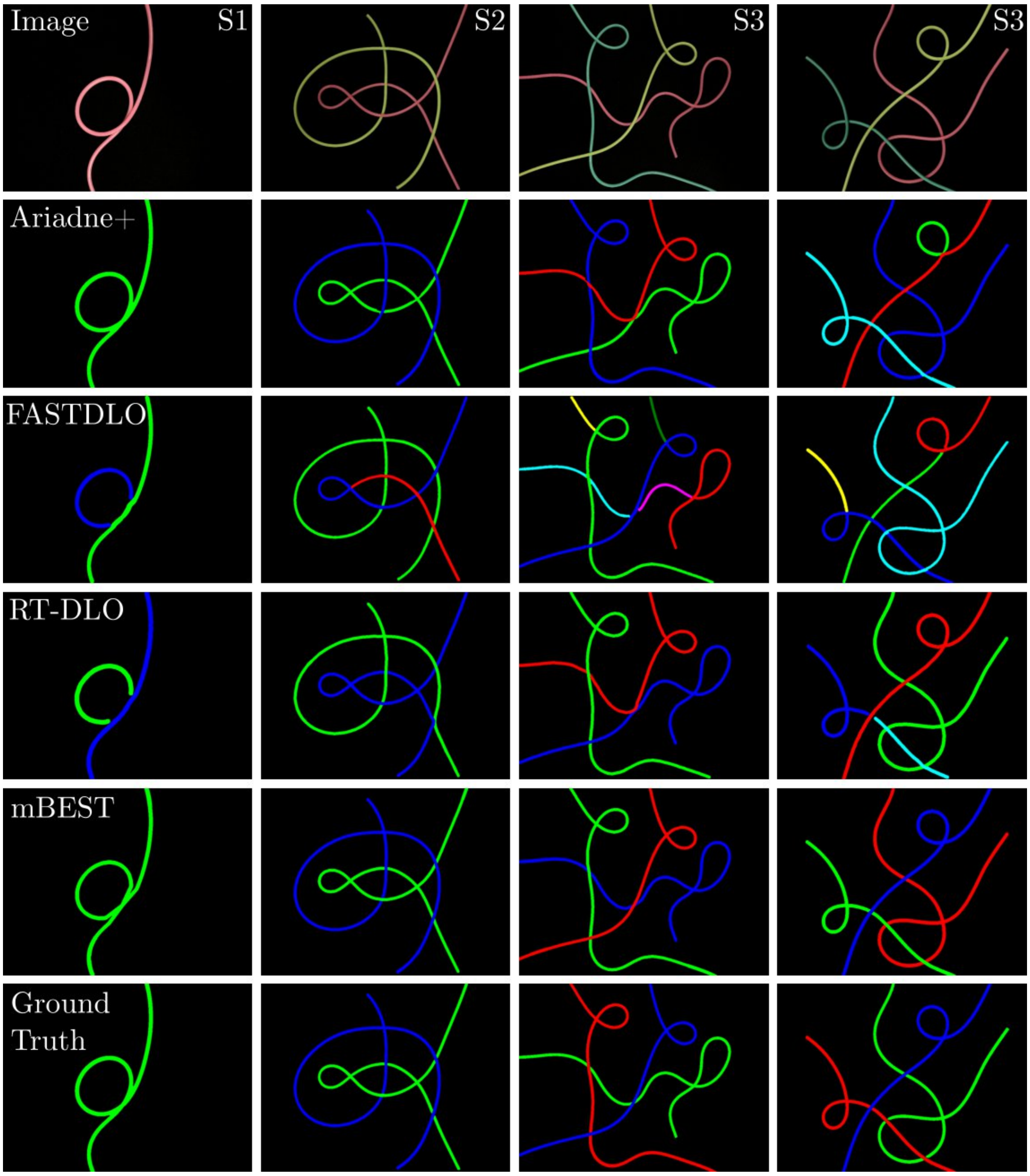


Figure 8.7: Sample segmentations for images in datasets S1, S2, and S3. Each column shows segmentation results for a different image with the top row indicating the dataset to which the image belongs. Rows 2–5 show results for *Ariadne+*, *FASTDLO*, *RT-DLO*, and *mBEST*, respectively. The bottom row shows the ground truth. Several cases of incorrect intersection handling can be observed for all the baseline algorithms, whereas *mBEST* robustly handles intersections using its simple bending energy optimization.

8.3.2 Baselines and Parameters

We tested *mBEST* against three state-of-the-art baselines: *Ariadne+* (Caporali et al., 2022b), *FASTDLO* (Caporali et al., 2022a), and *RT-DLO* (Caporali et al., 2023a). In terms of hyperparameters, the number of superpixels for *Ariadne+* was set to 75 for complex background images and to 200 for simple background images. Both these values were chosen as optimal after performing a parameter sweep on each dataset. For *RT-DLO*, the K-nearest neighbors matching parameter was set to 8, the edge similarity threshold was set to 0.1, and the vertex sampling ratio was set to 0.15. These hyperparameters were provided by default and shown to have good performance in Caporali et al. (2023a). For all experiments involving use of the DCNN model, a pixel segmentation threshold of 77 (0–255) was used. Furthermore, although *Ariadne+* has its own neural network for the initial segmentation of the DLOs, we replaced it with *FASTDLO*'s DCNN model for consistency and better performance.

Additionally, we demonstrated the effectiveness of *mBEST*'s skeleton refinement steps by conducting experiments on an aggregated dataset consisting of S1, S2, and S3 with a hybrid formulation that uses *FASTDLO*'s intersection handling neural network in *mBEST*'s framework.

All experiments were run on a workstation with an Intel i9-9900KF CPU and an NVIDIA RTX 2080 Ti GPU.

8.3.3 Results and Analysis

We report two key metrics. First, we look at segmentation accuracy using the popular DICE metric. We also report the average run times for each algorithm in frames per second (FPS). Tables 8.1 and 8.2 report both metrics for all our experiments, respectively.

For the complex background datasets, we see that *mBEST* outperforms all baseline algorithms in terms of mean DICE score. In particular, we see that the baseline algorithms often struggle to handle intersections that are nearly parallel, as shown in Figures 8.5 and 8.6.

Table 8.1: mBEST vs. SOTA DICE results.

Dataset	DICE [%]			
	Ariadne+	*FASTDLO*	*RT-DLO*	*mBEST*
C1	88.30 ± 0.102	89.82 ± 0.091	90.31 ± 0.085	$\mathbf{91.08 \pm 0.083}$
C2	91.03 ± 0.044	91.45 ± 0.039	91.10 ± 0.058	$\mathbf{92.17 \pm 0.050}$
C3	86.13 ± 0.123	86.55 ± 0.110	87.27 ± 0.128	$\mathbf{89.69 \pm 0.089}$
S1	97.24 ± 0.065	87.91 ± 0.062	96.72 ± 0.014	$\mathbf{98.21 \pm 0.006}$
S2	96.81 ± 0.074	88.92 ± 0.061	94.91 ± 0.019	$\mathbf{97.10 \pm 0.010}$
S3	96.28 ± 0.067	90.24 ± 0.042	94.12 ± 0.043	$\mathbf{96.98 \pm 0.009}$

Table 8.2: mBEST vs. SOTA runtime results.

Dataset	Runtime [FPS]			
	Ariadne+	*FASTDLO*	*RT-DLO*	*mBEST*
C1	2.69	20.81	30.58	**31.86**
C2	2.63	20.90	**32.50**	32.03
C3	2.72	20.51	**32.44**	32.17
S1	0.92	21.88	39.60	**52.79**
S2	0.78	17.34	25.73	**41.04**
S3	0.73	15.33	22.06	**37.11**

With regard to runtime, *mBEST* is roughly on par with *RT-DLO* and is a clear improvement over *Ariadne+* ($\approx 11\times$) and *FASTDLO* ($\approx 1.5\times$). Note three important caveats for these results: 1) the initial DCNN segmentation comprises a large portion of the computation time; 2) the images are relatively small and the number N of DLOs is random, giving little insight as to how the algorithms scale with N, and 3) *RT-DLO*'s ability to keep up with *mBEST* in speed is solely due its low vertex sampling rate (0.15). We observe that increasing the sampling rate increases the compute time significantly given the computational expense of graph construction.

To address the above concerns, consider the results for the simple background datasets. As these datasets do not require the use of a DCNN and are labeled by the number of DLOs they contain, we can accurately determine how each algorithm scales and performs with respect to N. As reported Table 8.2, *mBEST* offers clear speed improvements

Table 8.3: Skeleton refinement analysis on datasets S1+S2+S3.

Algorithm	DICE [%]	Runtime [FPS]
FASTDLO	89.02 ± 0.056	16.13
HYmBEST	97.39 ± 0.013	29.94
mBEST	97.43 ± 0.010	42.86

over the *Ariadne+* ($\approx 54\times$), *FASTDLO* ($\approx 2.4\times$), and *RT-DLO* ($\approx 1.5\times$) baselines. Additionally, we see that *mBEST* scales better with respect to N compared to *RT-DLO* despite the latter's sparse sampling rate, with *mBEST* experiencing runtime decreases of about 22.3% and 9.6% when moving up each tier compared to *RT-DLO*'s 35% and 14.3%. Though a low sampling rate works well for the relatively straight configurations of DLOs in C1, C2, and C3, we notice that performance degrades significantly once a coarse sampling rate is unable to capture the highly variable curvatures of complex assemblies of rods (i.e., those in S1, S2, and S3). Examples of this can be observed in our supplementary video.[*]

In addition to the significant improvement in runtime, *mBEST* also outperforms all the baseline algorithms in terms of mean DICE score as well. Several examples of intersection failures experienced by the baseline algorithms are shown in Figure 8.7. Such failures typically occur in extreme cases (i.e., either nearly-parallel or extremely curved self-loops). Interestingly, *Ariadne+*'s mean DICE score is very close to *mBEST*'s, but had up to 10× the standard deviation, meaning that *Ariadne+* suffered a higher number of outright failures. In fact, *mBEST* has a lower standard deviation compared to all the baseline algorithms across all the datasets with the exception of C2, indicating a higher level of consistency for a wide range of data.

Finally, we analyze the effectiveness of our skeleton refinement steps by formulating a hybrid algorithm, *HYmBEST*, which uses *FASTDLO*'s intersection handling neural network (IHNN) plugged into *mBEST*'s framework. As reported in Table 8.3, *HYmBEST* achieves a mean DICE score almost identical to *mBEST*, with both significantly out-

[*]See `https://github.com/StructuresComp/mBEST`.

performing *FASTDLO*. This is noteworthy as it shows that *FASTDLO*'s IHNN works reasonably well, but that the improperly handled skeleton structure yields poor results, thus highlighting the importance of the topology-correcting refinement steps. Note also that although *FASTDLO*'s IHNN can perform well in a hybrid formulation, *mBEST*'s remarkably cheap bending energy formulation still results in an $\approx 43\%$ runtime improvement.

8.4 Conclusion

In this chapter, we have introduced *mBEST*, an end-to-end pipeline for the segmentation of deformable linear objects (DLOs) in images that improves upon the state of the art both in terms of accuracy and computational speed. Through a variety of experiments, we have shown that *mBEST* can robustly handle complex scenes with highly tangled DLOs by generating paths on topologically correct skeletons that minimize the cumulative bending energy of the scene.

In future work, we will explore solutions that take into consideration occlusions, multiple DLOs at an intersection, poor quality binary masks, and dense knots; i.e., strands touching in parallel. We note that though we do not cover it in this manuscript, the bending energy formulation of *mBEST* can easily be expanded to deal with multiple DLOs at an intersection by simply accounting for additional path combinations. Furthermore, methods like *RT-DLO* (Caporali et al., 2023a) already take into consideration the possibility of poor binary masks and may be better suited for such situations. Finally, another promising research direction is 3D detection of DLOs, thus enabling robots to go beyond simple planar manipulation. Solutions for this may involve using *mBEST* to generate segmentations from multiple viewing angles for the purposes of 3D reconstruction.

CHAPTER 9

Sim2Real Neural Controllers for DLO Deployment

Note: This work on DLO deployment is very similar to the work of Chapter 7 in that we use scaling analysis with Buckingham π theorem to produce a generalizable formulation. The key difference in this chapter is that rather than a sim2real model (which is then indirectly used to compute optimal trajectories), we learn a sim2real optimal control policy directly.

9.1 Introduction

Prior works on manipulating DLOs can be divided into two categories. The first involves robots attempting to manipulate DLOs to satisfy some high-level conditions without controlling the exact shapes of DLOs. This includes knot tangling/untangling (Wakamatsu et al., 2006; Saha and Isto, 2007), obstacle avoidance (McConachie et al., 2020; Mitrano et al., 2021), following guidance and insertion (She et al., 2021; Zhu et al., 2019), etc. The second category involves robots attempting to precisely control the exact shape of the DLOs. For this task, a key challenge is formulating a mapping between the robot's motions and the shape of the manipulated DLO (Nair et al., 2017; Takizawa et al., 2015; Lv et al., 2022).

In this chapter, we look into how to design a manipulation scheme for controlling the shape of elastic rods through deployment, which involves manipulating one end of DLO in a way that gradually lays the DLO on a substrate in a desired pattern with superhuman accuracy, sufficient efficiency, and strong robustness. The full end-to-end pipeline of our physics-based deployment scheme is shown in Figure 9.1. In addition to achieving precise

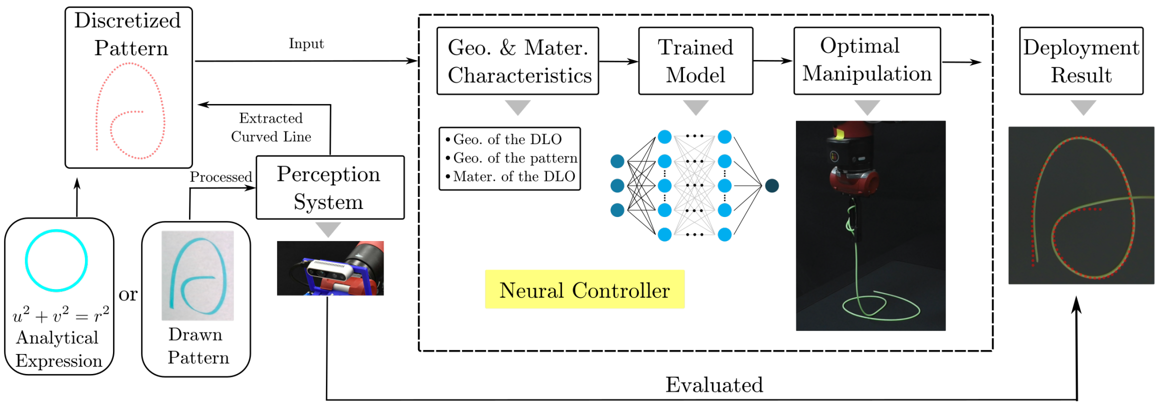


Figure 9.1: A full end-to-end pipeline for deploying a DLO with a sim2real physics-based deployment scheme. The pipeline begins by discretizing the DLO pattern, which can be obtained through user input via an analytical expression or a hand-drawn pattern scanned by a perception system (Choi et al., 2023b). A neural controller trained entirely from simulation then generates an optimal manipulation path for deploying the pattern, taking into account the shape of the pattern as well as the geometrical and material properties of the DLO. Finally, the deployment result is evaluated using an Intel RealSense camera positioned to provide a top-down view of the pattern to assess the accuracy of the deployment.

shape control, we show our control method can be used to solve high-level tasks such as reproducing human writing with a deployed DLO, cable placement, and knot tying.

Deploying DLOs is instrumental in the practical world, e.g., drawing or writing on cakes with icing (Sun et al., 2015), deploying marine cables (Whitcomb, 2000), depositing carbon nanotubes (Geblinger et al., 2008), and melting electrospinning for advanced manufacturing (Teo and Ramakrishna, 2006). Therefore, a concrete and applicable deployment scheme is a perfect solution to the shape control problem of DLOs.

Now a natural question arises: how to deploy a DLO along a prescribed pattern accurately on a substrate? Intuitively, we can assume that during the deployment process, the manipulated end $\mathbf{q}_M$ is directly above the contact point $\mathbf{q}_C$ and that the gripper's decreasing distance along the negative z-axis is equal to the added deployed length on the substrate. However, this deployment strategy does not take into consideration the nonlinear geometric deformations of the manipulated DLO and therefore, results in a poor quality deployment as illustrated by later experimental results. A schematic of the intuitive deployment method inspired by Takizawa et al. (2015) can be observed in Figure 9.2(a).

In this chapter, we propose a framework that combines physically accurate simulation, scaling analysis, and machine learning to generate an optimized control scheme capable of deploying solid rod-like structures, which we refer to as DLOs, along any feasible pattern. Our control scheme does not currently incorporate energy dissipation from manipulations with DLOs such as viscous threads, as our physical-based simulation is based on the rod model. However, the controlling scheme can be adapted by adjusting the physical-based simulation in our combined framework to include these factors. We validate the scheme with various DLOs (e.g., elastic rods, rope, and cable) in robotic experiments. The usage of physically accurate numerical simulations not only allows us to incorporate physics into our manipulation scheme but also results in full sim2real realization. Scaling analysis allows us to formulate the problem with generality using non-dimensional parameters, resulting in a control scheme robust against the material properties of the manipulated rods. Finally, machine learning allows us to train a neural network to model the controlling rules of deployment in a data-driven fashion. The high inference speed of our neural controller makes real-time operation feasible.

Our main contributions in this chapter are as follows:

1. we formulate a solution to the DLO shape control problem through deployment with a physically robust scheme that leverages scaling analysis, resulting in generality against material, geometric, and environmental factors (friction);
2. we train a neural network (NN) with non-dimensional simulation data to serve as a fast and accurate neural controller for optimal manipulations of deployment tasks. The trained mechanics-based NN solver has remarkable efficiency and sufficient accuracy when compared to a numerical solver; and
3. we demonstrate full sim2real realization through an extensive robotic case study demonstrating our control method's success for various practical deployment patterns with various DLOs on different substrates. In addition, we showcase the utility of our control scheme for complex high-level applications such as mimicking human

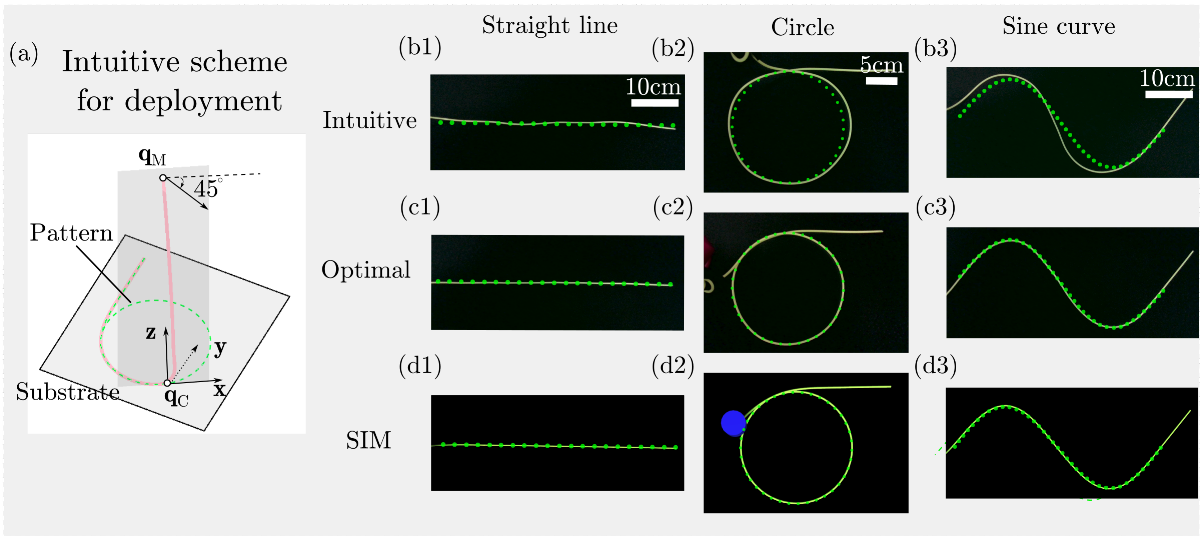


Figure 9.2: (a) Schematic of the intuitive control method from Takizawa et al. (2015). A DLO is being deployed along a circular pattern shown in dashed green. During the deployment process, the manipulated position $\mathbf{q}_M$ deploys along the tangent of the pattern $\mathbf{x}$ in a downward 45-degree angle with respect to the y-axis. The x-z-plane is shown in opaque gray. In addition, a comparison of experimental results between the (b) intuitive control method, (c) our designed optimal control method, (d) and simulation results using the optimal control method for the patterns of straight line, circle, and sine curve are shown. Note the effects of forgoing the influence of nonlinear geometric deformations in the intuitive deployment scheme's failure to follow simple patterns.

handwriting, managing cables, and tying different knots.

Moreover, we have released our source code and supplementary videos.[*]

9.2 Generalized Solution and Scaling Analysis

When manipulating DLOs, we should consider their geometrically nonlinear deformations. Moving forward, $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ refer to the unit directors of the coordinate system defined by the connective node $\mathbf{q}_C$ as shown in Figure 9.3. When a DLO is being deployed along a prescribed pattern on a rigid substrate, it can be divided into two parts: a deployed part on the substrate and a suspended part that does not contact the substrate (Figure 9.3). Here, we presume the pattern on the substrate is fixed since the DLO should ideally be deployed along the prescribed pattern. Therefore, the unknown deformations only exist

[*]See `https://github.com/StructuresComp/rod-deployment`.

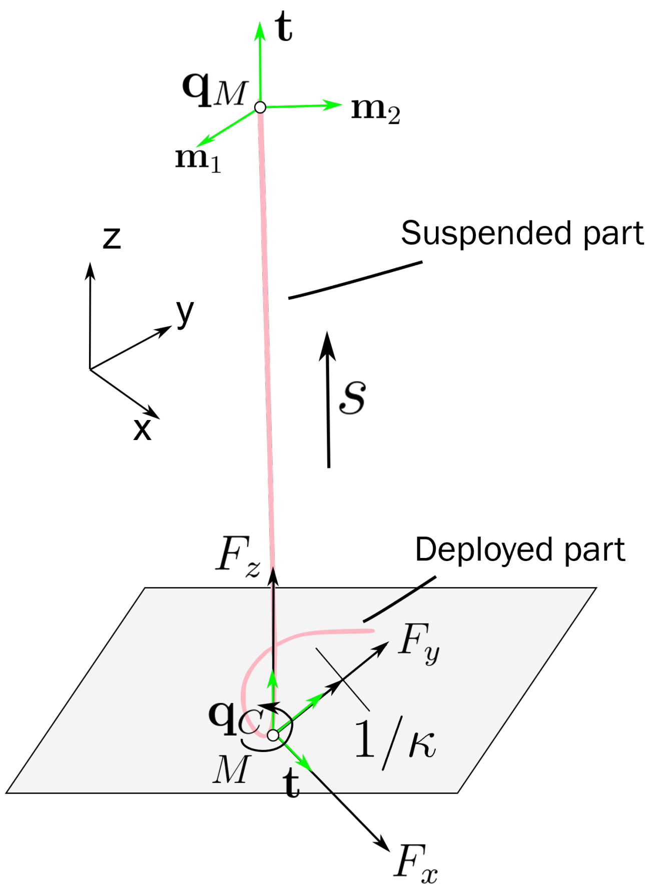


Figure 9.3: Schematic of deploying a DLO along a prescribed pattern.

in the suspended part. Moving forward, all simulations of the DLO will be performed using DER (Chapter 3).

9.2.1 Solving the Suspended Part

To capture the deformations of the suspended part, we introduce some quantities to assist our analysis. First, we define $\mathbf{q}(s)$ to describe the position of the manipulated DLO's centerline, where s is the arc length along the DLO's centerline. Then, a material frame $\mathbf{m}(s) = [\mathbf{m}_1, \mathbf{m}_2, \mathbf{t}] \in SO(3)$ is attached along the DLO to capture the DLO's rotation, where $\mathbf{t} = \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}s}$ is the tangent of the DLO. With the help of $\mathbf{q}(s)$ and $\mathbf{m}(s)$, we can fully describe the deformed configuration of the suspended part.

To solve the configuration of the suspended part, we can treat the suspended part as an independent DLO starting from the connective node $\mathbf{q}_C$ to the manipulated node $\mathbf{q}_M$.

Here, $\mathbf{q}_C = \mathbf{q}(0)$ is the connective node connecting the deployed part and the suspended part. Given the continuity of the manipulated DLO, the curvature vector $\boldsymbol{\kappa}$ at $\mathbf{q}_C$ can be obtained from the prescribed pattern, where the magnitude of $\boldsymbol{\kappa}$ is denoted as κ. The manipulated end grasped by the robot is then $\mathbf{q}_M = \mathbf{q}(l_s)$, where l_s is the total curve length of the suspended part. Deployment of the pattern is then carried out purely by controlling $\mathbf{q}_M$. Since (3.9) implies that the DLO's configuration $\mathbf{q}(s)$ and $\mathbf{m}(s)$ can be solved when boundary conditions are determined, we can write down the governing equations for the suspended part as

$$\begin{aligned} &\mathcal{R}(\mathbf{q}) = 0, \\ \text{s.t.} \quad &\mathbf{q}(l_s) = \mathbf{q}_M, \quad \mathbf{R} = \mathbf{m}(l_s)\mathbf{m}(0)^T, \\ &\mathbf{q}(0) = \mathbf{q}_C, \quad \frac{\mathrm{d}\mathbf{q}(0)}{\mathrm{d}s} = \mathbf{t}(0), \quad \frac{\mathrm{d}\mathbf{t}(0)}{\mathrm{d}s} = \kappa\hat{\mathbf{y}}, \end{aligned} \tag{9.1}$$

where $\mathcal{R}$ is the DER equations of motion (3.9), and $\mathbf{q}_M$ is the position and $\mathbf{R}$ is the orientation of the manipulated end with respect to the connective node $\mathbf{q}_C$. Note that the position of the connective node $\mathbf{q}_C$, tangent $\mathbf{t}(0)$, and curvature vector $\kappa\hat{\mathbf{y}}$ can be determined from the deployed pattern, where $\hat{\mathbf{y}}$ is the unit vector illustrated in Figure 9.3. By solving (9.1), we can obtain the configuration of the suspended part for any predefined pattern and manipulated end pose.

9.2.2 Influence of Force and Friction

Once the deformed configuration is known, we can now calculate the forces applied on the suspended part, which is key to hyper-accurate control of the DLO. We denote the external forces $\mathbf{F}^{\text{ext}} = F_x\hat{\mathbf{x}} + F_y\hat{\mathbf{y}} + F_z\hat{\mathbf{z}}$ and twisting moment $M(0)$ applied on the suspended part from the connective node $\mathbf{q}_C$. Here, the moment M is a function of arc length s; for example, $M(s)$ is the twisting moment applied on the manipulated end. The quantities $\mathbf{F}^{\text{ext}}$ and $M(0)$ are relevant with the friction coefficient μ between the substrate and the rod, and μ is an unknown and uncontrollable environment factor. In addition, the quantities $\mathbf{F}^{\text{ext}}$ and $M(0)$ also influence the tangent $\mathbf{t}(0)$ at the connective node $\mathbf{q}_C$

because of Newton's third law. Therefore, we must minimize quantities $\mathbf{F}^{\mathrm{ext}}$ and $M(0)$ to achieve an optimal controlling rule.

Despite the optimal controlling rule minimizing the influence of friction, it is still worth clarifying the significance of friction within this context. Though we make the strong assumption that the deployed pattern remains fixed during deployment, this is only upheld if the following relation is satisfied for the deployed segment:

$$k_b \kappa'' \leq \mu_s \rho A g, \tag{9.2}$$

where k_b is the bending stiffness of the rod, κ'' is the second derivative of κ with respect to the arc length s ($\kappa'' = \frac{\mathrm{d}^2\kappa}{\mathrm{d}s^2}$), μ_s is the static friction coefficient, ρ is the volumetric density of the rod, A is the cross-sectional area, and g is the gravitational acceleration. Here, (9.2) is derived by analyzing an arbitrary finite element of the deployed pattern with a clamped-end Euler-Bernoulli beam model. Clearly, friction plays a crucial role in the deployment process.

As a result, our designed optimal deployment strategy maintains a reliance on adequate friction for effective execution while the scheme mitigates external tangential forces apart from the essential friction on the substrate. Consequently, the scheme necessitates only a modest static friction coefficient between the substrate and the manipulated DLO.

9.2.3 Computing Optimal Grasp

In addition to the minimization of the external forces $\mathbf{F}^{\mathrm{ext}}$ and twisting moment $M(0)$ applied on the suspended part, we set up a rule for the manipulated end: the robot end-effector should induce minimal deformations on the manipulated node $\mathbf{q}_M$ so that the curvature (bending deformations) at the manipulated end should be 0. This results in the following optimization problem to compute the optimal grasp:

$$\begin{aligned}(\mathbf{q}_M^*, \mathbf{R}^*) &= \arg\min \left(\|\mathbf{F}^{\text{ext}}\|^2 + \left(\frac{\|\mathbf{M}(0)\|}{h} \right)^2 \right) \\ \text{s.t.} \quad \frac{\mathrm{d}\mathbf{q}(0)}{\mathrm{d}s} &= \mathbf{t}(0), \quad \frac{\mathrm{d}\mathbf{t}(0)}{\mathrm{d}s} = \kappa \hat{\mathbf{y}}, \\ \frac{\mathrm{d}^2\mathbf{q}(l_s)}{\mathrm{d}s^2} &= 0, \qquad \mathcal{R}(\mathbf{q}) = 0.\end{aligned} \tag{9.3}$$

By solving (9.3), optimal grasp ($\mathbf{q}_M^*$ and $\mathbf{R}^*$) can be obtained. Physical analysis tells us that a direct mapping relationship exists between the contributing factors and the optimal grasp. Recall from (3.2), (3.5), and (3.7), that stretching stiffness $k_s \equiv EA$, bending stiffness $k_b \equiv EI$, twisting stiffness $k_t \equiv GJ$, density ρ, and rod radius h are the primary material and geometric properties of a rod. By adding in additional geometric properties such as suspended length l_s and curvature κ, the mapping relationship

$$(\mathbf{q}_M^*, \mathbf{R}^*) = f(l_s, \kappa, k_s, k_b, k_t, h, \rho) \tag{9.4}$$

can be constructed where $f(\cdot)$ is a highly nonlinear (and unknown) function that describes the controlling rule.

Note however the high input dimensionality of (9.4). In other words, to accurately learn the mapping $f(\cdot)$, we would have to exhaustively perform large parameter sweeps for various ranges of material and geometric parameters within simulations. This process of collecting data quickly gets out of hand due to the curse of dimensionality. To circumvent this, we can perform scaling analysis to obtain an equivalent reduced-order mapping.

9.2.4 Nondimensionalization via Buckingham π Theorem

Like in Section 7.4.2, we use Buckingham π theorem to reduce the dimensions of the mapping $f(\cdot)$. Using the theorem, we obtain the following reduced-order non-dimensionalized mapping $\mathcal{F}(\cdot)$ from the original function f:

$$
\begin{aligned}
(\bar{\mathbf{q}}_M^*, \mathbf{R}^*) &= \mathcal{F}(\bar{l}_s, \bar{\kappa}, \bar{k}_s), \\
L_{gb} &= \left(\frac{k_b}{2\pi h^2 \rho g} \right)^{1/3}, \\
\bar{k}_s &= \frac{k_s L_{gb}^2}{k_b}, \\
\bar{\mathbf{q}}_M^* &= \frac{\mathbf{q}_M^* - \mathbf{q}_C}{L_{gb}}, \\
\bar{l}_s &= \frac{l_s}{L_{gb}}, \\
\bar{\kappa} &= \kappa L_{gb}.
\end{aligned}
\tag{9.5}
$$

Hereafter, all quantities with $\bar{()}$ indicate normalized quantities. In (9.5), all quantities are unitless so that the mapping relationship $\mathcal{F}(\cdot)$ maps from the unitless groups encapsulated the geometric and material properties to the unitless optimal robotic grasp. The benefit of doing such is that we reduce the dimensions of the mapping function $\mathcal{F}(\cdot)$ in (9.4) and eliminate the dependence of $\mathcal{F}(\cdot)$ on the units. Note that in (9.5), we omit the twisting stiffness k_t as we found that twisting energies are minimal compared to bending and stretching. In the following section, we will show how to establish the nonlinear mapping function in (9.5).

9.3 Deep Learning and Optimization

In this section, we further analyze the optimization of the system to obtain the nonlinear mapping function in (9.5). Given the high nonlinearity of the system, we first solve (9.5) with a numeric optimization solver in a data-driven way. While doing so, we analyze the elastic instability of the system to choose the optimal robotic grasp for the deployment task. Afterward, we reconstruct (9.5) using a neural network to take advantage of its high inference speed. This neural controller is then used by our robotic system as the controlling law to complete various deployment tasks in Section 9.5.

9.3.1 Deployment in 2D Workspace

9.3.1.1 Elastic Instability in Straight Line Deployment

We first take a look at an intriguing physical phenomenon: elastic instability. Elastic instability occurs when changes in the boundary conditions cause a deformed structure to become unstable. When observed visually, a small geometric perturbation of the system will lead to a substantial change in configuration (Timoshenko and Gere, 2009). An example of this can be observed when a robot employs the intuitive control method to deploy a DLO along a straight line as the rod unexpectedly adopts a curved shape on the substrate. This observation defies our intuition as the intuitive method only manipulates the DLO in the 2D plane (x-z plane) as illustrated in Figure 9.2(a). Consequently, the suspended part should ideally experience only 2D deformations within that plane, thereby avoiding significant deformations along the y-axis. On the contrary, this observed phenomenon results from the unaccounted elastic instability of the manipulated DLO.

Given this, it is crucial to take elastic instability into consideration when designing an optimal deployment scheme so that the robot's grasp and possible jittering of the manipulator does not introduce large undesired deformations of the DLO. To achieve this, we thoroughly analyze all potential robot grasps for manipulating a DLO in the x-z plane to achieve a straight-line deployment. Our objective is to identify an optimal grasp that satisfies (9.3) while effectively preventing the manipulated DLO from buckling due to elastic instability.

9.3.1.2 Discovering the Potential Grasp Region

Given the suspended part's geometric properties and boundary conditions, we can write down the constraints $\mathcal{C}$ which should be satisfied:

$$\begin{aligned} &\bar{\mathbf{q}}(\bar{s}) \cdot \hat{\mathbf{z}} \geq 0 \;\forall\; \bar{s} \in [0, \bar{l}_s], \\ &\bar{\mathbf{F}}^{\text{ext}} \cdot \hat{\mathbf{z}} \geq 0. \end{aligned} \tag{9.6}$$

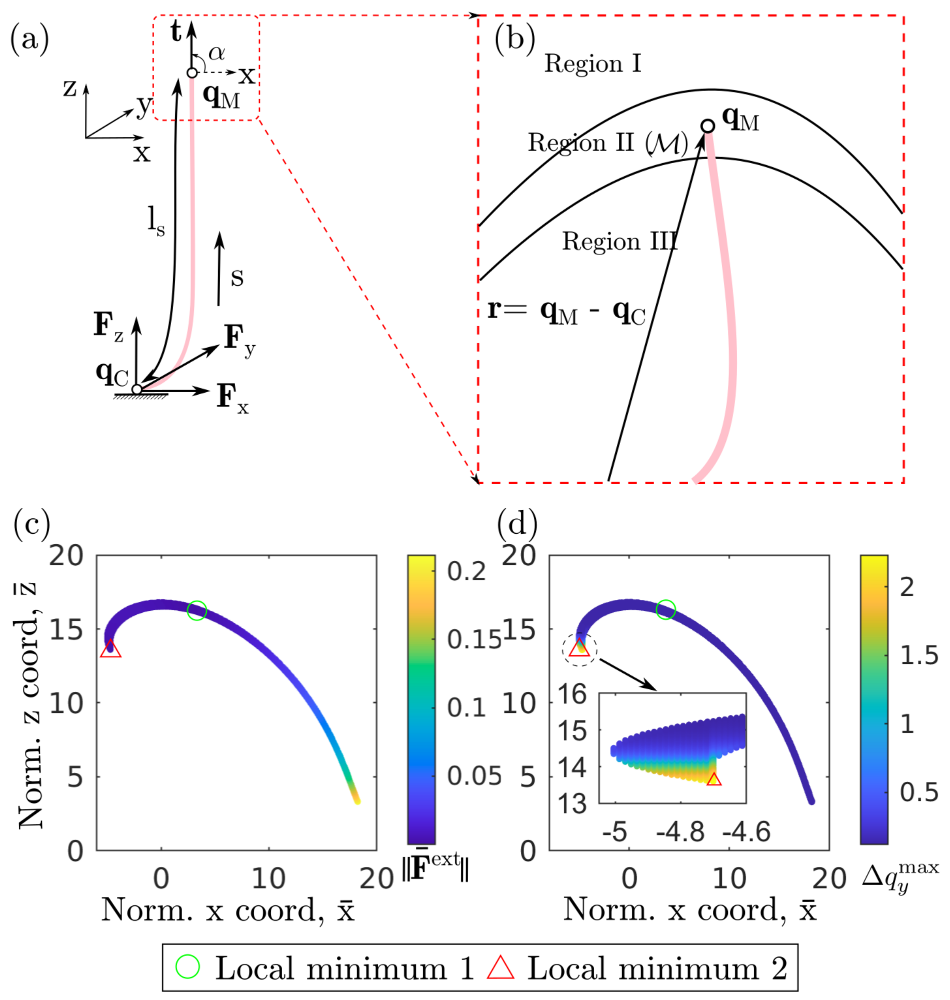


Figure 9.4: Schematic of a DLO manipulated in a 2D workspace (a) and its corresponding available region denoted by $\mathcal{M}$ (b). Visualization of a specific case with $\bar{l}_s = 17.68$. The force distribution is shown in (c), and (d) displays the maximum geometric deformation of the suspended part under a disturbance of $\Delta\bar{y} = 0.12$ along the y-axis.

These constraints enforce that (i) the suspended part should be above the substrate and (ii) external contact responses along the z-axis should always be larger than or equal to 0.

By solving (9.1) with constraints $\mathcal{C}$, we obtain all potential robot grasps of the manipulated end, forming a closed manifold $\mathcal{M}$ for a fixed normalized suspended length $\bar{l}_s$. The boundary condition at the connective node $\bar{\mathbf{q}}_C$ is defined as $\mathbf{t}(0) = (1, 0, 0)$ and $\bar{\kappa} = 0$. Each point in the manifold $\mathcal{M}$ corresponds to a position $\bar{\mathbf{q}}_M$ and rotation $\mathbf{R}$ of the manipulated end. Given that the deformed configuration is located within the 2D x-z plane, we can use a 2×1 vector $\bar{\mathbf{q}}_M = (\bar{x}_{\text{Top}}, \bar{z}_{\text{Top}})$ to express the position of $\bar{\mathbf{q}}_M$ and a scalar value α to denote the rotation information. For example, tangent $\mathbf{t}(\bar{l}_s) = (\cos(\alpha), \sin(\alpha))$ is shown in Figure 9.4(a). Since the manifold $\mathcal{M}$ is a closed set, we only need to obtain the boundary of the manifold $\partial\mathcal{M}$.

To discover the boundary $\partial\mathcal{M}$, we explore along a ray $\mathbf{r}$ from the connective node $\bar{\mathbf{q}}_C$ to the manipulated node $\bar{\mathbf{q}}_M$. The robot grasp along the ray can be divided into three regions as shown in Figure 9.4(b). When the robot grasp exists in regions I and III, constraints $\mathcal{C}$ are not satisfied. In region I, the external force $\bar{F}_z = F_z h^2/k_b$ is smaller than 0, violating the constraints as stretching occurs, and in region III, the manipulated end is too low, leading to contact between the suspended part and the substrate. Thus, region II, existing between regions I and III, represents the manifold $\mathcal{M}$ area that satisfies the constraints $\mathcal{C}$. To obtain the boundary $\partial\mathcal{M}$ of region II, a bisection method is used (further details in Tong et al. (2023c)).

A specific case for $\bar{l}_s = 17.68$ is visualized in Figure 9.4(c). Since deformations only occur in the x-z plane, the twisting moment $\bar{\mathbf{M}}(0) = \mathbf{M}(0)h/k_b$ applied on the connective node $\bar{\mathbf{q}}_C$ is always 0. To achieve the optimal pose of the manipulated end for $\bar{l}_s = 17.68$, we need to find the poses in $\mathcal{M}$ that minimizes $\|\bar{\mathbf{F}}^{\text{ext}}\|$. Two local minima are found in the case shown in Figure 9.4(c), corresponding to two solutions of (9.3). As stated before, we must select the local minima corresponding to the stable deformed suspended part.

9.3.1.3 Checking Elastic Instability via Perturbations

To test the elastic stability of these local minima, we apply a disturbance $\Delta\bar{y} = y/L_{gb}$ along the y-axis while the manipulated end $\bar{\mathbf{q}}_M$ is at each local minimum. Figure 9.5 illustrates the changes in $\bar{F}_y = F_y h^2/k_b$ and the configurations resulting from these perturbations for each local optimum.

For local minimum 2, we can see a sudden snapping process, where an immediate change can be observed, while the disturbance on local minimum 1 results in a continuous and steady change. Therefore, we can conclude that the optimum for deploying the DLO is at a local minimum 1 since this minimum corresponds to a configuration with more

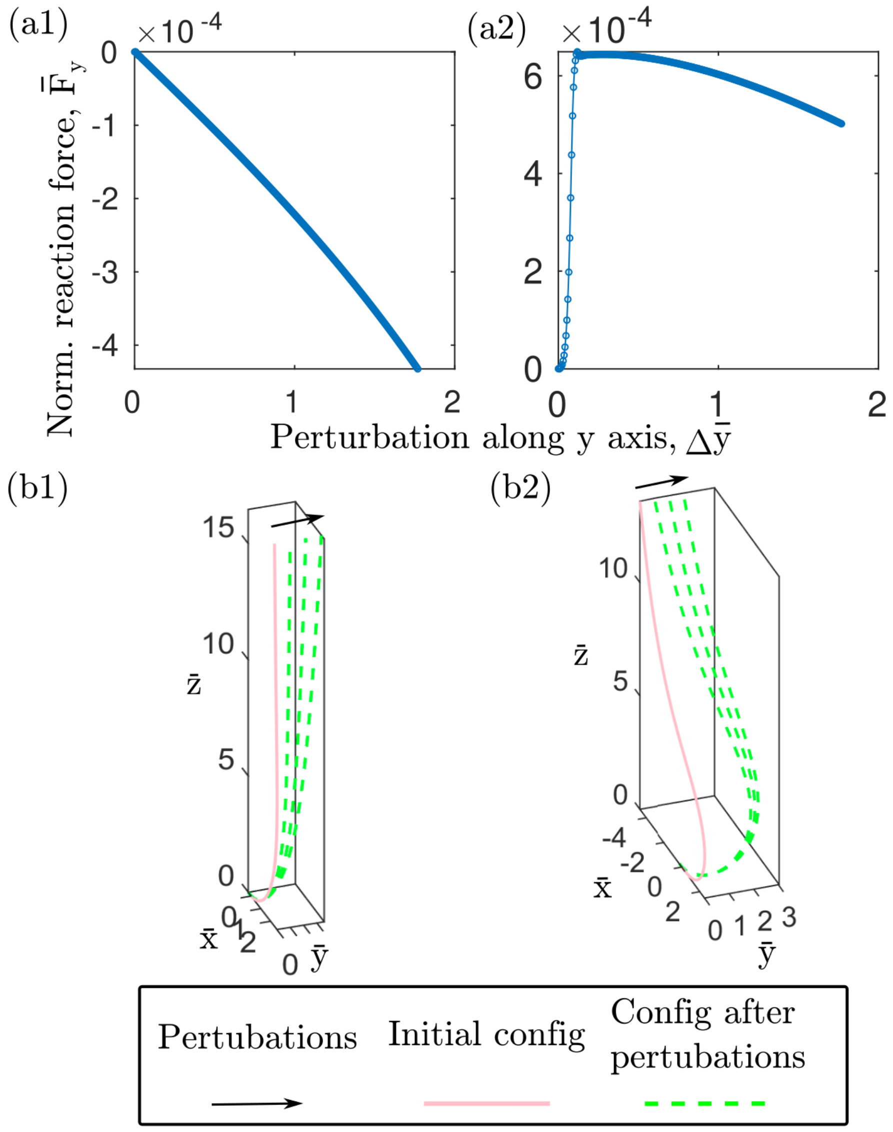


Figure 9.5: Change of the magnitude of normalized force $\bar{F}_y$ when adding a perturbation along the y-axis at local minimum 1 (a1) and local minimum 2 (a2), and change of the configurations of the rod when adding the perturbations at local minimum 1 (b1) and local minimum 2 (b2) for $\bar{l}_s = 17.68$.

gentle bending deformations of the suspended part.

Here, we also illustrate that the neighboring region around the elastic instability points has a higher tendency for significant deformations when the jittering of the manipulator occurs. In simulation, we introduce a small disturbance of $\Delta\bar{y} = 0.12$ along the y-axis for all potential robot grasps on the manifold $\mathcal{M}$. Figure 9.4(d) illustrates the maximum displacement of the suspended part along the y-axis $\Delta q_y^{\max} = \max_{0 \le s \le l_s}(\mathbf{q}(s) \cdot \hat{\mathbf{y}})$ caused by this small disturbance. It is evident from the results that the neighboring region around local minimum 2 exhibits a higher tendency for significant deformations along the y-axis. Consequently, robot grasps within this region are more likely to induce instability in the manipulated DLO.

We can now output the optimal deployment rule for a straight line using the method introduced in this section. In the next section, we focus on optimal 3D manipulation, i.e., deploying patterns with nonzero curvature. The following section discusses how to use a first-order optimization algorithm to solve (9.3) for deploying any arbitrary prescribed pattern, where the optima for straight-line deployment is used as seeds when searching for the optima of more complex patterns.

9.3.2 Deployment in 3D Workspace

As mentioned in Section 9.2.4, the mapping relationship $\mathcal{F}(\cdot)$ in (9.5) must be constructed to achieve optimal deployment in the 3D workspace. For the connective node of any prescribed pattern, since the deformations of the pattern are only in the x-y plane, we can ensure that the twisting moment $\mathbf{M}(0)$ can always be 0. Therefore, the optimal pose of the manipulated end can be obtained by minimizing $\|\bar{\mathbf{F}}^{\text{ext}}\|$ by solving

$$\nabla_{\bar{\mathbf{q}}_M} \|\bar{\mathbf{F}}^{\text{ext}}\| = \frac{\partial \bar{\mathbf{F}}^{\text{ext}}}{\partial \bar{\mathbf{q}}_M} \bar{\mathbf{F}}^{\text{ext}} = 0. \tag{9.7}$$

As the deploying rod is a continuous system, $\bar{\mathbf{F}}^{\text{ext}}$ must change when $\mathbf{q}_M$ changes. Therefore, we can convert (9.7) to be a root finding problem

$$\bar{\mathbf{F}}^{\text{ext}} = 0. \tag{9.8}$$

As discussed before, solving the configurations of the deploying DLO is a boundary value problem. Since the pattern's shape determines the boundary conditions on the connective end, the external forces $\bar{\mathbf{F}}^{\text{ext}}$ are influenced solely by the manipulated end pose

Algorithm 10: Gradient Descent for Optimal Grasp

```
   Input: $\bar{l}_s, \bar{\kappa}\hat{\mathbf{y}}, \bar{k}_s, \nu$
   Output: $\bar{\mathbf{q}}_M^*$
1  Func OptimalGrasp($\bar{l}_s, \bar{\kappa}, \bar{k}_s$):
2      $k \leftarrow 0$
3      $\delta \leftarrow$ a small value as tolerance
4      $\bar{\mathbf{q}}_M^{(k)} \leftarrow$ initialize a random pose of end-effector
5      $\mathcal{R}(\cdot) \leftarrow$ initialize the rod solver with $\bar{l}_s, \bar{\kappa}, \bar{k}_s$
6      do
7          $\bar{\mathbf{F}}^{\text{ext}} \leftarrow \mathcal{R}(\bar{\mathbf{q}}_M^{(k)})$
8          $\mathbf{J}^{\text{ext}} \leftarrow$ (9.9)
9          $\Delta\bar{\mathbf{q}} \leftarrow (\mathbf{J}^{\text{ext}})^{-1}\bar{\mathbf{F}}^{\text{ext}}$
10         $\alpha \leftarrow$ LineSearch($\bar{\mathbf{q}}_M^{(k)}, \Delta\mathbf{q}, \|\bar{\mathbf{F}}^{\text{ext}}\|, \mathcal{R}$)
11         $\bar{\mathbf{q}}_M^{(k+1)} \leftarrow \bar{\mathbf{q}}_M^{(k)} - \alpha\Delta\bar{\mathbf{q}}$
12         $k \leftarrow k + 1$
13     while $\|\bar{\mathbf{F}}^{\text{ext}}\| \geq \delta$
14     $\bar{\mathbf{q}}_M^* \leftarrow \bar{\mathbf{q}}_M^{(k)}$
15     return $\bar{\mathbf{q}}_M^*$
```

$\mathbf{q}_M$, with a unique corresponding $\mathbf{R}$ for describing the rotation of the manipulated end.

Given the high nonlinearity of the DLO, it is nontrivial to solve the root-finding problem in (9.8) analytically. Therefore, we employ a finite difference approach to calculate the numerical Jacobian of $\mathbf{F}^{\text{ext}}$. We perturb the manipulated end along x, y, and z-axes with a small distance ϵ and use the finite difference to compute the numerical Jacobian

$$\mathbf{J}^{\text{ext}} = \frac{1}{\epsilon}\begin{bmatrix} \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M + \epsilon\hat{\mathbf{x}}) - \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M), \\ \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M + \epsilon\hat{\mathbf{y}}) - \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M), \\ \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M + \epsilon\hat{\mathbf{z}}) - \bar{\mathbf{F}}^{\text{ext}}(\bar{\mathbf{q}}_M) \end{bmatrix}^T, \tag{9.9}$$

where T is the transpose operator and $\epsilon\hat{\mathbf{x}}, \epsilon\hat{\mathbf{y}}$, and $\epsilon\hat{\mathbf{z}}$ are small perturbations along x, y, and z-axes, respectively; i.e., $\epsilon\hat{\mathbf{x}} = [\epsilon, 0, 0]^T$. Here, $\mathbf{J}^{\text{ext}}$ is a 3×3 matrix and can be used to calculate the Newton search step so that (9.8) can be solved with a gradient descent method. Further details of this solving process are stated in Alg. 10. Additionally, we also implement a simple line search algorithm to help determine the appropriate step size for the Newton search step $\Delta\bar{\mathbf{q}}$ as shown in Alg. 11.

Algorithm 11: Line Search Algorithm

```
Input: $\bar{\mathbf{q}}_M, \Delta\mathbf{q}, f_0, \mathcal{R}$
Output: $\alpha$
1  Func LineSearch($\bar{\mathbf{q}}_M, \Delta\mathbf{q}, f_0, \mathcal{R}, \alpha_0 = 1, m = 0.5$):
2      $\alpha \leftarrow \alpha_0$
3      $k \leftarrow 0$
4      success $\leftarrow$ False
5      do
6          $\bar{\mathbf{q}}_M^{(k)} \leftarrow \bar{\mathbf{q}}_M - \alpha\Delta\mathbf{q}$
7          $\bar{\mathbf{F}}^{\text{ext}} \leftarrow \mathcal{R}(\bar{\mathbf{q}}_M^{(k)})$
8          $f^{(k)} \leftarrow \|\bar{\mathbf{F}}^{\text{ext}}\|$
9          if $f^{(k)} \geq f_0$ then
10             $\alpha \leftarrow m\alpha$
11             $k \leftarrow k + 1$
12         else
13             success $\leftarrow$ True
14         end
15     while not success
16     return $\alpha$
```

Both position $\bar{\mathbf{q}}_M$ and rotation $\mathbf{e}$ of the manipulated end are represented as 3×1 vectors: $\bar{\mathbf{q}}_M = (\bar{x}^{\text{Top}}, \bar{y}^{\text{Top}}, \bar{z}^{\text{Top}})$ and $\mathbf{e} = (e_x, e_y, e_z)$. The rotation vector $\mathbf{e}$ can be translated to a rotation matrix through an axis-angle representation $(\hat{\mathbf{e}}, \|\mathbf{e}\|)$, where $\|\mathbf{e}\|$ is the rotation angle along the rotation axis $\hat{\mathbf{e}} = \mathbf{e}/\|\mathbf{e}\|$. For an input tuple $(\bar{l}_s, \bar{\kappa}, \bar{k}_s)$, we can now solve for the optimal pose of the manipulated end $(\mathbf{q}_M^*, \mathbf{e}^*)$. Visualizations of the discretely solved optimal poses obtained from simulation are shown as red hollow circles in Figure 9.6.

We now know how to obtain the optimal manipulation pose given the input $(\bar{l}_s, \bar{\kappa}, \bar{k}_s)$ with simulations. A numeric solver based on simulations for generating the optimal trajectory for various prescribed patterns is released.[*] However, solving for the optimal poses with the numeric solver makes real-time manipulation infeasible as trajectory generation can take several hours. Instead, the following section introduces using a neural network to learn the optimal controlling rule for fast real-time inference.

[*]See https://github.com/StructuresComp/rod-deployment.

9.3.3 Training the Neural Controller

Rather than obtaining the optimal grasp through the numerical solver detailed in the previous section, we train a neural network to learn an analytical approximation of $\mathcal{F}(\cdot)$ similar to the approach in Choi et al. (2023c). We use a simple fully-connected feed-forward nonlinear regression network consisting of 4 hidden layers, each with 392 nodes, as the network architecture. Aside from the output, each layer is followed by a rectified linear unit (ReLU) activation.

We frame the neural controller to have an input $\mathbf{i} \in \mathbb{R}^3$ and an output $\mathbf{o} \in \mathbb{R}^6$, where the input consists of the three non-dimensional values $\bar{l}_s$, $\bar{\kappa}$, and $\bar{k}_s$ and the output consists of two concatenated 3×1 vectors: the optimal position $\bar{\mathbf{q}}_M^*$ and rotation $\mathbf{e}^*$ of the manipulated end. Using our simulation framework, we construct a dataset $\mathcal{D}$ consisting of 6358 training samples.

When training the neural controller, we first preprocess all inputs $\mathbf{i}$ through the standardization

$$\mathbf{i}' = \frac{\mathbf{i} - \bar{\mathbf{i}}_{\mathcal{D}}}{\boldsymbol{\sigma}_{\mathcal{D}}},$$

where $\bar{\mathbf{i}}_{\mathcal{D}}$ and $\boldsymbol{\sigma}_{\mathcal{D}}$ are the mean and standard deviation of the input portion of the dataset $\mathcal{D}$. Afterward, we use an initial 80-20 train-val split on the dataset $\mathcal{D}$ with a batch size of 128. We use mean absolute error (MAE) as our loss and use a training strategy that alternates between stochastic gradient descent (SGD) and Adam whenever training stalls. In addition, the batch size is gradually increased up to a max size of 2048, and the entire dataset is used to train the controller once MAE reaches < 0.003. With this scheme, we achieve a final MAE of < 0.0015. The neural network's approximation of $\mathcal{F}(\cdot)$ can be seen visualized in Figure 9.6.

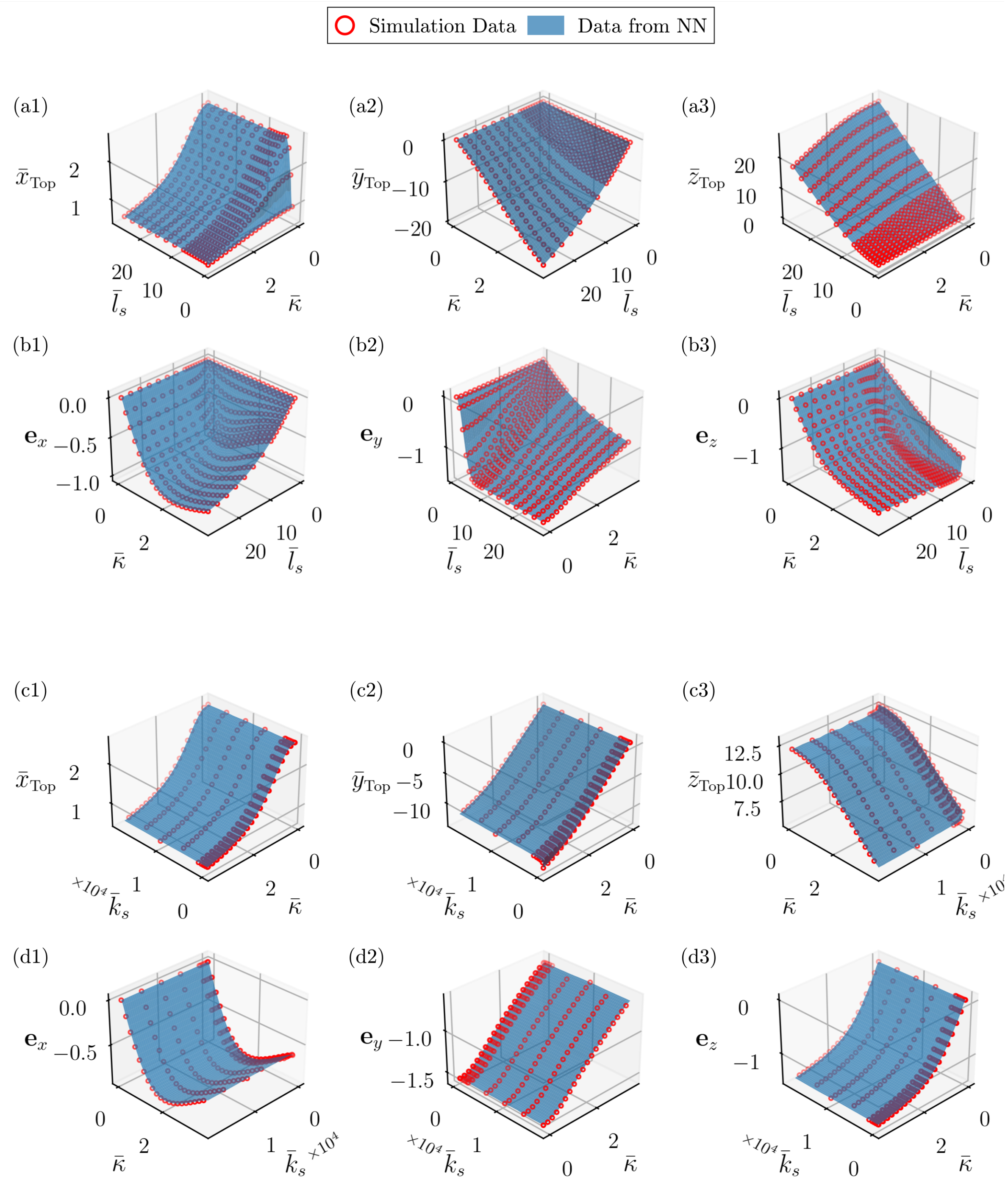


Figure 9.6: Visualization of the influence from curvature $\bar{\kappa}$ and suspended length $\bar{l}_s$ on the (a1-a3) manipulated end position and (b1-b3) manipulated end orientation for fixed values of $\bar{k}_s = 2087$; visualization of the influence from stretching stiffness $\bar{k}_s$ and curvature $\bar{\kappa}$ on the (c1-c3) manipulated end position and (d1-d3) manipulated end orientation with fixed values of $\bar{l}_s = 13.68$.

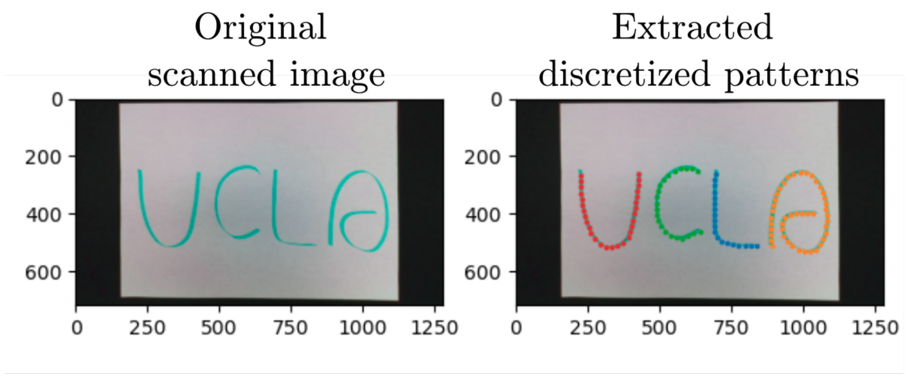


Figure 9.7: Handwritten letters and the corresponding extracted discretized patterns using mBEST (Choi et al., 2023b).

9.4 Robotic System

For our experiments, we used two Rethink Robotics' Sawyer manipulators, as shown in Figure 9.8. One arm is attached with a gripper for manipulating the rod while the other is outfitted with an Intel RealSense D435 camera. Furthermore, a workstation with an AMD Ryzen 7 3700X CPU and an NVIDIA RTX 2070 SUPER GPU was used for all experiments.

9.4.1 Perception System

To obtain the Cartesian centerline coordinates of the deployed DLO, we use the DLO perception algorithm mBEST (Choi et al., 2023b) from Chapter 8. In addition to detecting DLOs, mBEST is also used for general pattern extraction. One case of extracting discretized patterns from a handwritten pattern is shown in Figure 9.7. RGB images of the deployed DLO are obtained through an Intel RealSense D435 camera, as shown in Figure 9.8.

9.4.2 Motion Planning with the Neural Controller

In Figure 9.1, we showcase the full end-to-end pipeline of our proposed deployment scheme. Here, we give a full description of how to integrate the trained neural controller into a

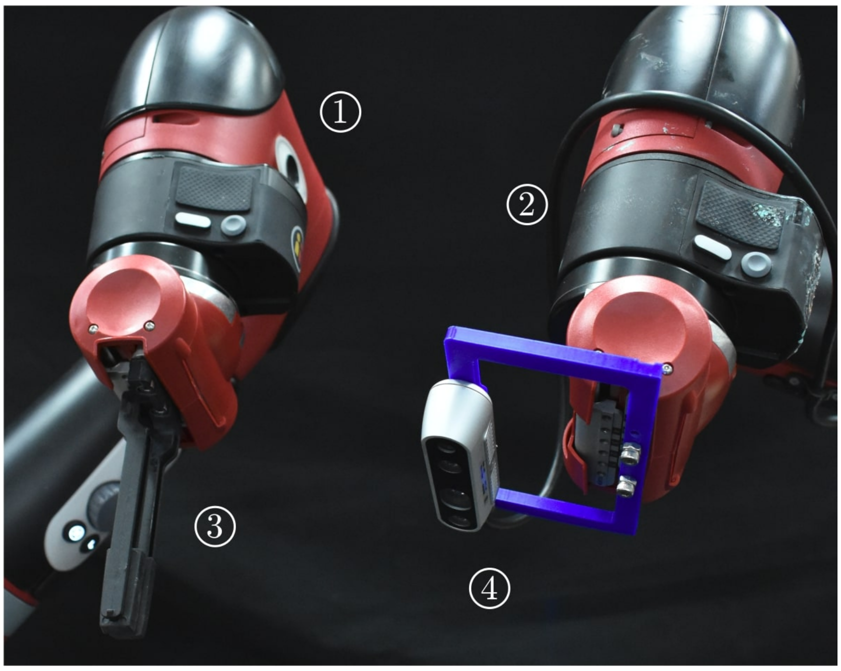


Figure 9.8: Experimental apparatus: Two robot manipulators, one for manipulation of the deploying rod (1) and the other for holding the camera for perception (2). A gripper (3) is used for grabbing the manipulated end of the rod. A camera (4) is used for extracting patterns from the drawn patterns and evaluating the deployment results.

robot motion planner.

The first step of the deployment process is to specify the desired pattern. This pattern can be defined by either an analytical function or detected as a drawn curve (Figure 9.1). Note that the pattern $\mathbf{P}(s)$ is treated as a function of the curve length s. Based on the configuration of the pattern, we can compute the required inputs for the neural controller when the connective node $\mathbf{q}_C$ achieves each point in the pattern $\mathbf{P}(s)$. The details of generating an optimal trajectory based on the pattern $\mathbf{P}(s)$ and the properties of the manipulated rod are given in Alg. 12.

In Alg. 12, κ and $\mathbf{T}$ are all functions of the arc length s of the pattern, where $\mathbf{T}$ is the tangent along the pattern. With Alg. 12, we obtain the optimal grasp trajectory τ and then use OMPL (Sucan et al., 2012) to generate a valid motion planning sequence on a real robot system.

One highlight of our overall robotic system is its real-time capability. The real-time

Algorithm 12: Optimal Deployment Trajectory

```
Input: $\mathbf{P}, L, L_{gb}, \bar{k}_s$
Note that material parameters $L_{gb}, \bar{k}_s$ must be measured in advance (Figure 9.9
and (9.10)).
Output: $\tau$
1  Func OPT($L, \mathbf{P}, L_{gb}, \bar{k}_s$):
2      $S, \kappa, \mathbf{T} \leftarrow$ ProcessPattern ($\mathbf{P}$)
3      $\Delta s \leftarrow$ step size of deployment
4      $\tau \leftarrow$ initialize an empty list
5      $\hat{\mathbf{z}} \leftarrow$ director along vertical direction
6      $s \leftarrow 0$
7      while $s \leq S$ do
8          $\mathbf{q}_C \leftarrow \mathbf{P}(s)$
9          $\hat{\mathbf{x}} \leftarrow \mathbf{T}(s)$
10         $\bar{\kappa} \leftarrow \kappa(s) L_{gb}$
11         $\bar{l}_s \leftarrow (L - s)/L_{gb}$
12         $(\bar{\mathbf{q}}_M^*, \mathbf{e}^*) \leftarrow \mathcal{F}(\bar{l}_s, \bar{\kappa}, \bar{k}_s)$
13         $\mathbf{R} \leftarrow$ AxangtoRot ($\hat{\mathbf{e}}^*, \|\mathbf{e}^*\|$)
14         $\mathbf{R}_t \leftarrow (\hat{\mathbf{x}}, \hat{\mathbf{z}} \times \hat{\mathbf{x}}, \hat{\mathbf{z}})$
15         $\mathbf{q}_M^* \leftarrow \mathbf{q}_C + \mathbf{R}_t \bar{\mathbf{q}}_M^* L_{gb}$
16         $\mathbf{R}^* \leftarrow \mathbf{R}_t \mathbf{R}$
17         Append $(\mathbf{q}_M^*, \mathbf{R}^*)$ to $\tau$
18         $s \leftarrow s + \Delta s$
19     end
20     return $\tau$
```

efficiency of the perception algorithm has been validated by Choi et al. (2023b) while the average end-to-end time to generate a full optimal deployment motion plan is less than 1 second. Therefore, our approach is also efficient enough for sensorimotor closed-loop control. However, as offline control has achieved excellent deployment accuracy in our experiments, online control is not carried out in this work.

9.5 Experiments and Analysis

9.5.1 Measurement of Material Parameters

To carry out deployment with our proposed scheme, we must validate its efficacy with comprehensive experiments. In this chapter, we deploy various DLOs on different sub-

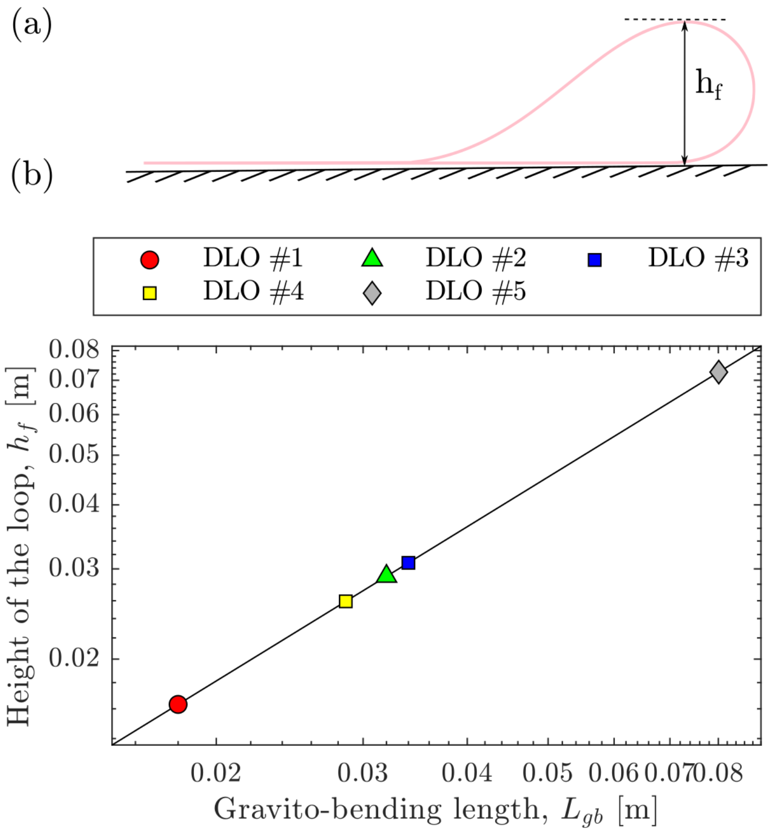


Figure 9.9: (a) Deformed configurations of a DLO under gravity in 2D plane; (b) Relationship between the height of the loop h_f and the gravito-bending length L_{gb}.

strates for multiple tasks so that we can look into the robustness of the proposed scheme against the material difference and friction.

First, we need to find the geometric and material properties of the manipulated DLO. The geometry of the manipulated rod, e.g., total length L and rod radius h, is trivial to measure. Measuring the material properties of the DLO is less clear. Overall, we need to develop a way to find the following material properties: gravito-bending length L_{gb} and normalized stretching stiffness $\bar{k}_s$.

Here, we presume the material is linearly elastic and incompressible. The incompressible material means the volume of the rod will not change when deformations happen. Therefore, Poisson's ratio is set as $\nu = 0.5$. In addition, bending stiffness is $k_b = \frac{E\pi h^4}{4}$ where E is Young's modulus, and the expression for gravito-bending length L_{gb} and

Table 9.1: Material and geometric properties of the DLOs used in the experiments. The amount of friction for different substrates is also shown.

DLO	Material & Geometric Parameters							
	Material	L_{gb} [cm]	h [mm]	L [m]	ν	μ_{fabric}	μ_{steel}	μ_{foam}
#1	Pink VPS	1.8	1.6	0.875	0.5	Low	Medium	High
#2	Green VPS	3.2	1.6	0.885	0.5	Low	Medium	High
#3	Rope	3.4	2.0	0.89	0.5	Medium	Low	High
#4	Pink VPS	2.86	3.2	0.84	0.5	Low	Medium	High
#5	Cable	8.01	1.8	0.87	0.5	Medium	Low	High

normalized stretching stiffness $\bar{k}_s$ is

$$\begin{aligned} L_{gb} &= \left(\frac{Eh^2}{8\rho g}\right)^{1/3}, \\ \bar{k}_s &= \frac{k_s L_{gb}^2}{k_b} = \frac{4L_{gb}^2}{h^2}. \end{aligned} \tag{9.10}$$

When observing (9.10), we find that the only parameter we must obtain is L_{gb}. It is still unclear how to compute this as L_{gb} is relevant to Young's Modulus E and the density ρ of the rod, which is usually hard to measure. Here, we propose a simple method that is able to measure L_{gb} by observing the geometry of the rod. When we form a loop in a rod naturally using gravity in a 2D plane, we can observe the geometry of the rod becomes what is shown in Figure 9.9(a). Indeed, the height h_f of the loop has a linear relationship with L_{gb}. Therefore, we can obtain L_{gb} for different rods by simply measuring h_f. According to prior work (Pan et al., 2020) and our validation shown in Figure 9.9(b), $h_f = 0.9066L_{gb}$.

9.5.2 Experiment Setup

9.5.2.1 Materials

We conducted experiments involving five distinct types of DLOs. Among these, three are silicone-based rubber fabricated by vinyl polysiloxane (VPS); the fourth is a commercially available rope; and the fifth is a stiff USB cable. Note that we also validate the robustness of the deployment scheme against different substrates. The friction between the DLOs and substrates is also qualitatively measured. Comprehensive details regarding the parameters for each of these DLOs can be found in Table 9.1.

9.5.2.2 Experiment Tasks

We implement our proposed deployment scheme across four distinct tasks. First, we deploy a rod along some canonical cases obtainable through analytical expressions such as a line, circle, and sine curve. The rod is deployed using the robotic arm with the gripper. Once the deployment is finished, the other arm with the camera moves to scan the deployment result.

The second task involves deploying patterns drawn on paper. Users draw patterns, subsequently scanned by the camera to obtain ordered discretized pattern coordinates. The robot then manipulates the rod to replicate the drawn pattern. This chapter showcases deployment results for the letters "U", "C", "L", and "A" with the precise shapes detailed in Figure 9.10(a). The third task is geared towards validating the deployment scheme's application in cable placement, a vital aspect of cable management. The scheme's efficacy is demonstrated by placing cables along constrained paths with the help of pre-installed fixtures on the substrate. Lastly, the deployment scheme's application for tying knots is verified. Both robotic arms are equipped with grippers for this task.

For the first two tasks, patterns are evaluated using both intuitive and optimal control methods. Additionally, three different rods (DLOs #1, #2, and #3) are deployed on substrates of various materials (fabric, steel, foam) to assess the method's robustness against material disparities and friction. In the third task, DLO #5 (USB cable) is employed for cable placement using both algorithms. Finally, DLOs #2 and #4 are used to tie distinct knots for the fourth task. Each experimental case is subjected to ten trials for each control method, culminating in a total of 1340 experimental trials.

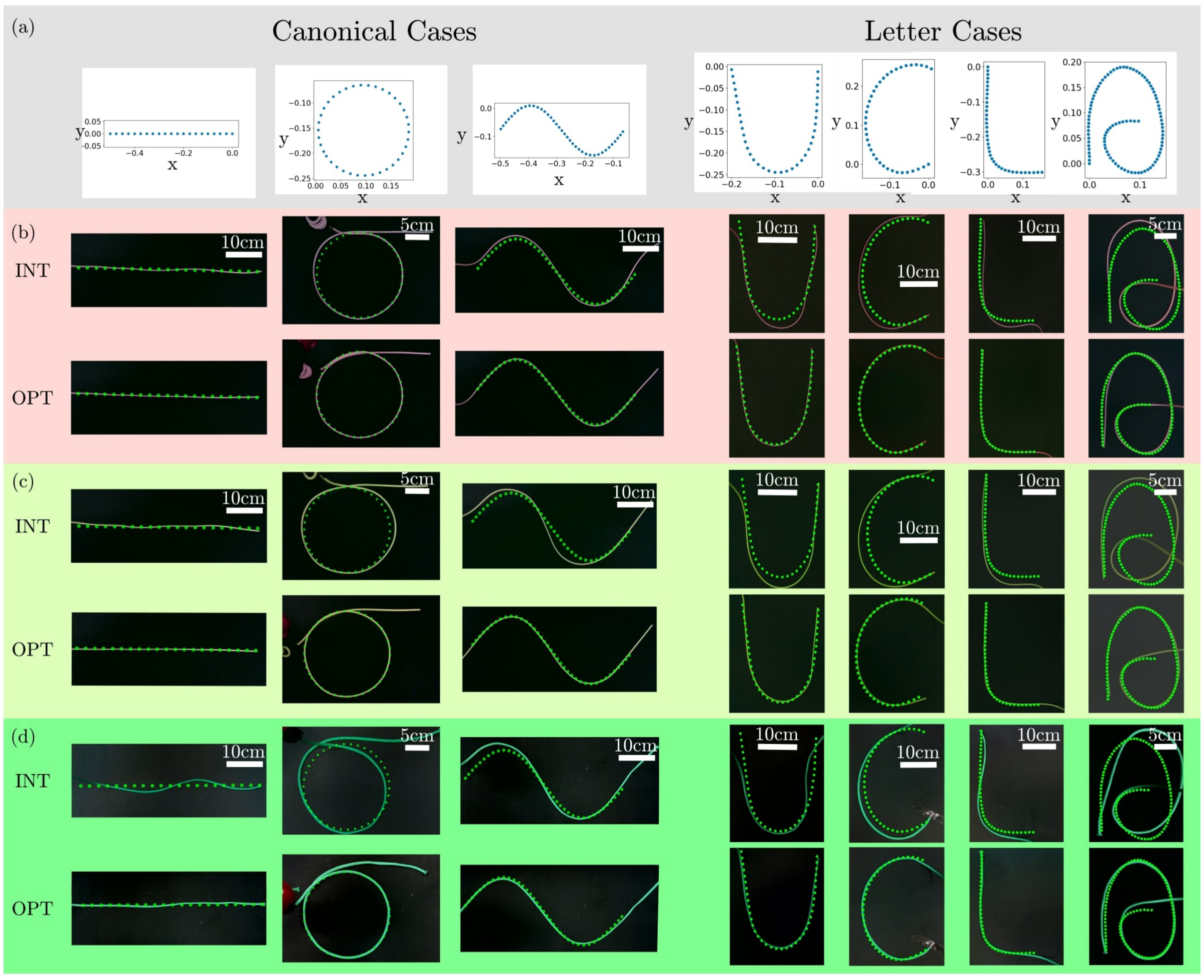


Figure 9.10: Experiment results of deployment along various patterns. (a) All used prescribed patterns are discretized and plotted. Deployment results for (b) DLO #1 (pink VPS), (c) DLO #2 (green VPS), and (3) DLO #3 (rope) are shown for each prescribed pattern. Results for the intuitive control method and optimal control method are shown for each rod.

9.5.3 Metrics

We now formulate the metrics used to evaluate the performance of the deployment scheme. When deploying a pattern $\mathbf{P}$, we need to assess the accuracy of the deployment result. We first discretize the pattern $\mathbf{P}$ into N points and denote the i-th point of the prescribed pattern as $\mathbf{P}^i$. The actual deployment pattern obtained from perception is denoted as $\mathbf{P}_{\text{exp}}$. With this discretization scheme, we compute the average error e_{mean} and standard

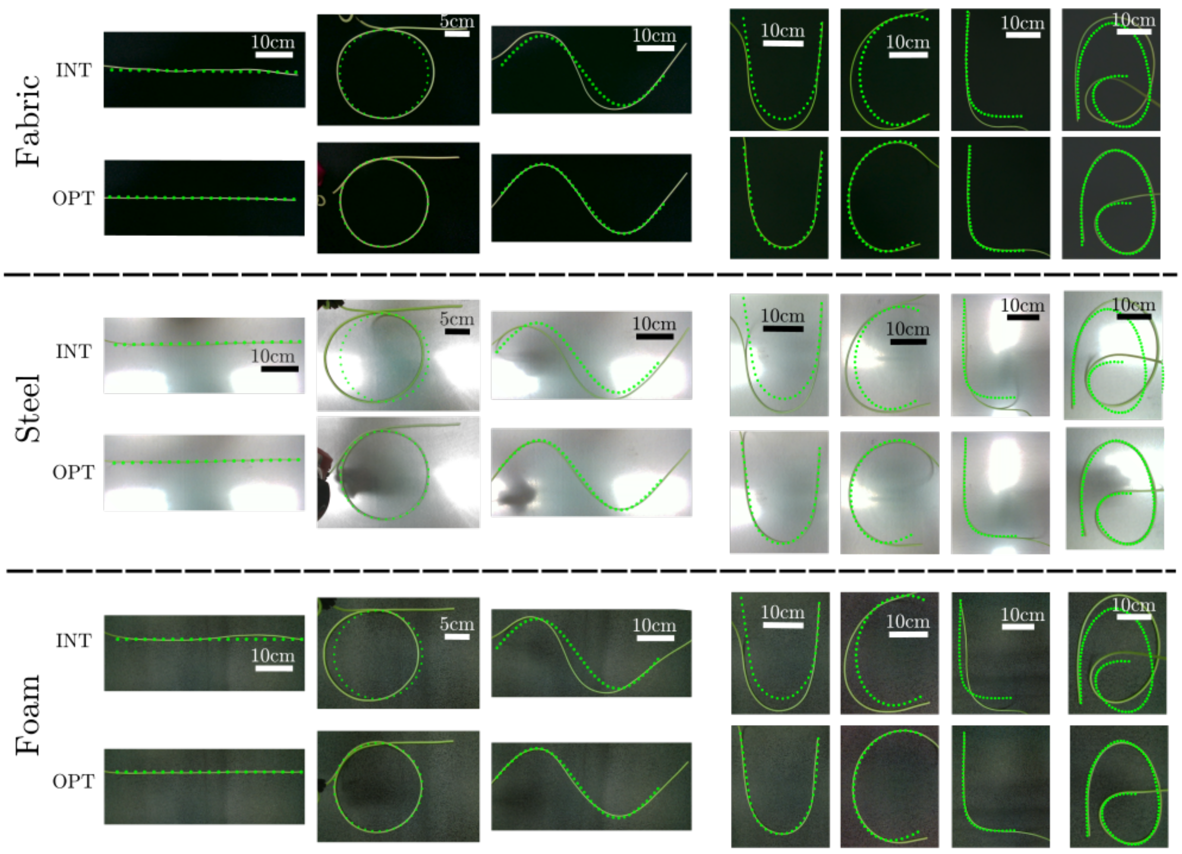


Figure 9.11: Experiment results of deployment with DLO #2 (green VPS) along various patterns on different substrates.

deviation σ as

$$\begin{aligned} e_{\text{mean}} &= \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{P}^i_{\text{exp}} - \mathbf{P}^i\|, \\ \sigma &= \sqrt{\frac{\sum_{i=1}^{N}(\|\mathbf{P}^i_{\text{exp}} - \mathbf{P}^i\| - e_{\text{mean}})}{N}}, \end{aligned} \tag{9.11}$$

for both the intuitive and optimal control results.

The accuracy evaluation is not applicable for the two application tasks: cable placement and knot tying as they are high-level tasks. Therefore, we simply use the success rate of those application tasks to evaluate the performance of the deployment scheme. In addition to accuracy, we also report a detailed comparison of runtimes and errors between the numerical and NN-based solvers. Details of the relevant results and analysis are discussed in the next section.

Table 9.2: Evaluation of deployment accuracy for canonical patterns using various DLOs and substrates.

DLO	SUB	Control Scheme	Pattern Type and Accuracy $e_{\text{mean}} \pm \sigma$ [cm] (9.11)		
			Line	Circle	Sine curve
#1	Fabric	INT	0.40 ± 0.22	0.61 ± 0.36	1.66 ± 0.74
		OPT	0.14 ± 0.09	0.15 ± 0.07	0.27 ± 0.10
	Steel	INT	1.42 ± 0.66	2.34 ± 1.24	2.69 ± 1.69
		OPT	0.22 ± 0.12	0.22 ± 0.08	0.27 ± 0.10
	Foam	INT	1.03 ± 0.21	1.23 ± 0.45	2.84 ± 1.52
		OPT	0.25 ± 0.15	0.18 ± 0.06	0.29 ± 0.16
#2	Fabric	INT	0.52 ± 0.13	1.64 ± 0.95	1.60 ± 0.83
		OPT	0.13 ± 0.07	0.16 ± 0.07	0.20 ± 0.09
	Steel	INT	1.72 ± 0.63	2.52 ± 1.02	3.30 ± 2.08
		OPT	0.17 ± 0.08	0.22 ± 0.09	0.54 ± 0.20
	Foam	INT	1.38 ± 0.60	2.24 ± 0.97	4.17 ± 2.57
		OPT	0.27 ± 0.13	0.20 ± 0.09	0.37 ± 0.14
#3	Fabric	INT	1.56 ± 0.81	1.13 ± 0.53	5.09 ± 1.35
		OPT	0.49 ± 0.28	0.29 ± 0.15	0.47 ± 0.23
	Steel	INT	4.53 ± 2.80	1.85 ± 0.45	4.43 ± 2.82
		OPT	0.47 ± 0.20	0.29 ± 0.13	0.46 ± 0.20
	Foam	INT	2.00 ± 0.88	1.94 ± 0.84	3.80 ± 1.96
		OPT	0.78 ± 0.34	0.27 ± 0.15	0.46 ± 0.20

9.5.4 Results and Analysis

9.5.4.1 Accuracy

All experimental results for canonical and handwritten patterns can be seen in Tables 9.2 and 9.3, respectively. To compute the error metrics in (9.11), we used a discretization of $N = 50$. From all results, we can observe a noticeable improvement in our optimal control method over the intuitive method for various geometrical, material, and environmental parameters.

To better visualize our method's generality, we visually depict deployment outcomes across different DLOs on the fabric surface in Figure 9.10. In addition, a comparative

Table 9.3: Evaluation of deployment accuracy for handwritten patterns using various DLOs and substrates.

DLO	SUB	Control Scheme	Pattern Type and Accuracy $e_{\text{mean}} \pm \sigma$ [cm] (9.11)			
			Letter "U"	Letter "C"	Letter "L"	Letter "A"
#1	Fabric	INT	1.39 ± 0.63	2.21 ± 0.92	1.00 ± 0.59	4.81 ± 2.27
		OPT	0.22 ± 0.07	0.22 ± 0.10	0.35 ± 0.18	0.47 ± 0.23
	Steel	INT	3.59 ± 2.39	3.67 ± 1.93	0.87 ± 0.55	3.64 ± 2.09
		OPT	0.24 ± 0.13	0.27 ± 0.09	0.42 ± 0.16	0.58 ± 0.37
	Foam	INT	3.33 ± 1.93	3.89 ± 1.29	1.13 ± 0.74	4.09 ± 2.19
		OPT	0.24 ± 0.15	0.41 ± 0.20	0.35 ± 0.12	0.54 ± 0.24
#2	Fabric	INT	3.74 ± 2.89	4.58 ± 1.15	1.74 ± 1.11	4.95 ± 2.63
		OPT	0.17 ± 0.11	0.19 ± 0.22	0.29 ± 0.11	0.32 ± 0.18
	Steel	INT	4.78 ± 4.15	6.66 ± 2.53	2.14 ± 1.26	5.23 ± 3.38
		OPT	0.21 ± 0.09	0.74 ± 0.31	0.66 ± 0.24	0.36 ± 0.17
	Foam	INT	5.42 ± 4.47	6.14 ± 3.08	1.70 ± 1.32	5.09 ± 3.39
		OPT	0.17 ± 0.08	0.39 ± 0.18	0.37 ± 0.15	0.43 ± 0.19
#3	Fabric	INT	4.22 ± 3.10	3.36 ± 1.58	2.37 ± 1.56	4.59 ± 2.54
		OPT	0.36 ± 0.18	0.35 ± 0.19	0.50 ± 0.24	0.56 ± 0.29
	Steel	INT	4.53 ± 2.80	3.35 ± 1.55	2.57 ± 1.62	4.30 ± 1.73
		OPT	0.47 ± 0.20	0.56 ± 0.20	0.51 ± 0.24	0.81 ± 0.30
	Foam	INT	3.67 ± 2.46	6.03 ± 3.11	3.32 ± 1.80	4.47 ± 2.50
		OPT	0.32 ± 0.16	0.56 ± 0.26	0.33 ± 0.14	0.52 ± 0.20

visual representation of deployment results for a single DLO (#2) on varying substrates is shown in Figure 9.11. Readers seeking comprehensive visual comparisons of all deployment outcomes can refer to the supplementary video for detailed insights.[*]

Among the seven deployed patterns, the first three (straight line, circle, and sine curve) are canonical cases; i.e., their shapes have explicit analytical expressions. Note that when deploying the circle and sine curve patterns, a small "remainder" section is first deployed. This is necessary as the circle and sine curve patterns have a non-zero curvature at the start of their pattern. We compensate for this by deploying a remainder part whose curvature gradually evolves from a straight line with 0 curvature to the prescribed

[*]See `https://github.com/StructuresComp/rod-deployment`.

curvature of the pattern's first point. The remainder can improve the deployment task's accuracy as the deployed pattern will require slight friction based on (9.2).

We have omitted the designed remainder for the four remaining patterns denoted by the letters "U", "C", "L", and "A" for better visualization. Among these, patterns "U", "L", and "A" exhibit a relatively low κ'' value during the beginning stage of the deployment, resulting in the deployment accuracy being minimally affected by surface friction.

Conversely, the "C" pattern demonstrates a comparatively higher κ'' value initially, leading to a possible noticeable mismatch between the deployed DLO and the intended pattern in the beginning. The impact of friction becomes more pronounced during the rope deployment corresponding to DLO #3 since the rope has higher bending stiffness k_b and experiences lower friction with the substrate. Fixing the free end is essential to precisely replicate the "C" pattern with the rope as shown in Figure 9.10(d). Despite this limitation, our optimized deployment strategy consistently outperforms the intuitive approach.

9.5.4.2 Computational Efficiency

Next, we also evaluated the computational efficiency of our neural controller. Tables 9.4 and 9.5 compare time costs between the neural network solver (NN-solver) and the numeric solver for canonical and handwritten patterns, respectively. When calculating a single optimal robot grasp for a given parameter tuple ($\bar{l}_s$, $\bar{\kappa}$, $\bar{k}_s$), the numeric solver takes approximately 10 to 20 seconds, while our NN-solver takes roughly 0.4 seconds.

The difference of time costs becomes more significant when generating a series of optimal robot grasps for a discretized pattern. Note that a discretized pattern typically consists of 100 to 200 nodes and that the numeric solver needs to compute the robot trajectory in sequence as the optimal grasp for the previous step is needed as the seed for computing the next optimal grasp. Therefore, the time costs quickly accumulate for the numeric solver, which substantially elongates the overall computation time. In contrast,

Table 9.4: Evaluation of computation times for canonical patterns for the numerical and NN-solvers. Error metrics for position and orientation between the neural controller and numeric solvers are also given.

DLO	Solver Times [s] & MAEs	Patterns with Number of Nodes		
		Line 101 nodes	Circle 156 nodes	Sine curve 138 nodes
#1	Numeric-Solver	1572.68	2036.11	2897.17
	NN-Solver	0.402	0.393	0.395
	Position Error [m]	0.0008	0.0007	0.0009
	Orientation Error	0.0012	0.0010	0.0032
#2	Numeric-Solver	776.56	1213.14	1769.66
	NN-Solver	0.397	0.391	0.396
	Position Error [m]	0.0016	0.0016	0.0019
	Orientation Error	0.0012	0.0078	0.0050
#3	Numeric-Solver	666.01	1041.71	1561.12
	NN-Solver	0.400	0.407	0.395
	Position Error [m]	0.0016	0.0017	0.0020
	Orientation Error	0.0010	0.0087	0.0052

the NN-solver leverages vectorization to solve multiple robot grasps simultaneously, resulting in a speed advantage of several orders of magnitude compared to the numeric solver when generating optimal deployment trajectories.

9.5.4.3 Precision of the Neural Controller

Finally, Tables 9.4 and 9.5 also presents the precision of the NN-solver. The solutions from the numeric solver serve as the ground truth. Mean Absolute Error (MAE) is employed to evaluate the optimal trajectories the NN-solver generates against the ground truth. Remarkably, the MAE consistently remains below 0.003 m for position error and 0.009 for differences in rotation quaternions. Importantly, it's noteworthy that none of the solved trajectories in this analysis were part of the training dataset. Thus, we can confidently assert that our NN-solver exhibits robustness, efficiency, and accuracy, rendering it well-suited for real-time control applications.

Table 9.5: Evaluation of computation times for handwritten patterns for the numerical and NN-solvers. Error metrics for position and orientation between the neural controller and numeric solvers are also given.

DLO	Solver Times [s] & MAEs	Patterns with Number of Nodes			
		Letter “U” 190 nodes	Letter “C” 190 nodes	Letter “L” 190 nodes	Letter “A” 194 nodes
#1	Numeric-Solver	3954.12	4015.24	4777.30	4666.55
	NN-Solver	0.431	0.431	0.400	0.417
	Position Error [m]	0.0008	0.0007	0.0008	0.0008
	Orientation Error	0.0025	0.0020	0.0020	0.0021
#2	Numeric-Solver	2286.66	2226.73	2720.08	2933.90
	NN-Solver	0.419	0.408	0.404	0.406
	Position Error [m]	0.0018	0.0016	0.0020	0.0017
	Orientation Error	0.0042	0.0020	0.0058	0.0030
#3	Numeric-Solver	1984.63	1972.39	2405.71	2639.44
	NN-Solver	0.407	0.420	0.405	0.411
	Position Error [m]	0.0020	0.0016	0.0021	0.0018
	Orientation Error	0.0055	0.0023	0.0054	0.0032

9.6 Use Cases

9.6.1 Cable Placement

In this section, we showcase the application of the deployment scheme for cable placement. The importance of cable management has surged, particularly in engineering contexts involving tasks like wire harnessing, infrastructure development, and office organization (Sanchez et al., 2018; Lattanzi and Miller, 2017). Given cables' inherent high bending stiffness, shaping them to specific forms can be challenging, often necessitating external fixtures to maintain the desired configuration. When humans perform cable management manually, meticulous placement along the designated pattern is essential, coupled with the use of fixtures to secure the cable in place. However, a robotic system can autonomously execute cable placement with our designed optimal deployment strategy.

In our experimental setup, we preinstalled external fixtures into the stainless steel

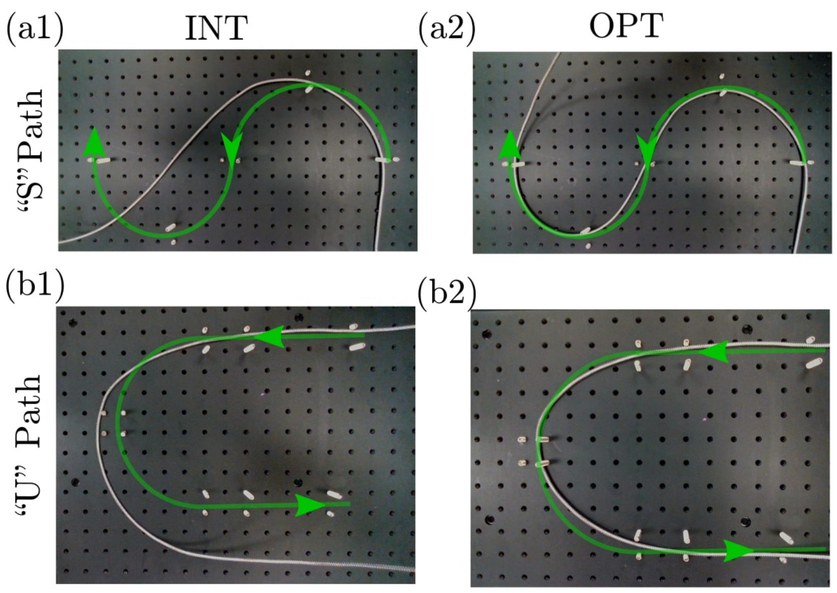


Figure 9.12: A demonstration of cable placement along different prescribed patterns with both intuitive and optimal control schemes.

breadboard to delineate the intended patterns. These fixtures also counteract the cable's rigid nature, preventing it from reverting to its original shape. The deployment results can be visualized in Figure 9.12. Compared to the failure placement results with the intuitive scheme, our optimal deployment scheme can place the cable along the prescribed pattern "U" and "S" on the substrate. We conducted 10 experimental trials for each deployment task illustrated in Figure 9.12. Notably, the optimal deployment approach achieved an impressive 90% (9/10) success rate for both patterns, whereas the intuitive method failed in all trials (0/10), as shown in Table 9.6.

9.6.2 Knot Tying

Since our optimal deployment scheme can control the shape of various DLOs with excellent accuracy, we can use this scheme to tie knots. First, the manipulated rod is deployed along a predesigned pattern on the substrate. Users can draw the predesigned pattern so that only a few extra manipulations are required. Then, the camera will scan the drawn pattern and send it as input to our designed scheme. The deployed pattern is designed in a way that only a few simple pick-and-place operations on certain knot segments is

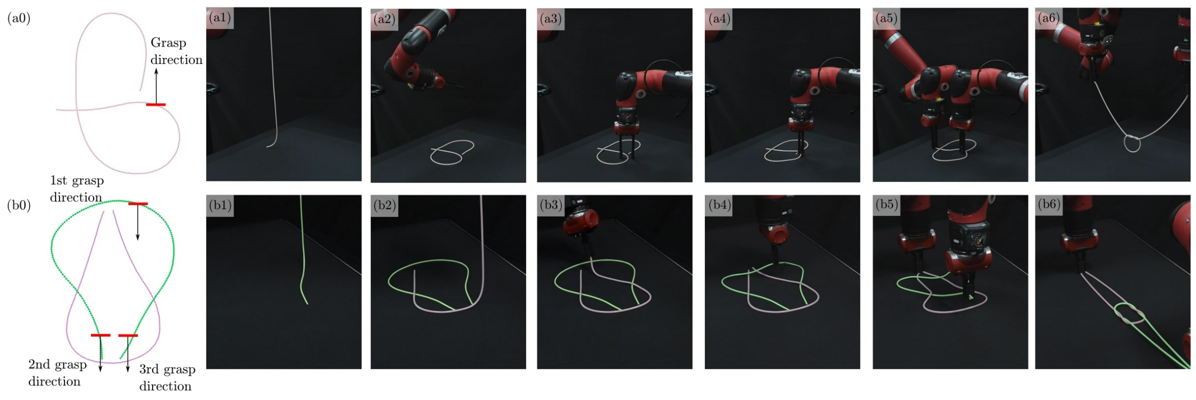


Figure 9.13: A demonstration of two knot-tying cases using the DLO deployment scheme. (a0) and (b0) are designed patterns for the trefoil knot and the reef knot, respectively. Time marches for trefoil knot from (a1) to (a6) and reef knot from (b1) to (b6).

required to complete the tying sequence. Since the prescribed pattern's shape is known in advance, we can let the robot execute the pick-and-place procedure without perception feedback. So long as the initial deployment is accurate and repeatable, the subsequent pick-and-place procedure should succeed most of the time.

We showcase two knot-tying sequences in Figure 9.13. The top row showcases a trefoil knot, one of the most fundamental knots in engineering (Crowell and Fox, 2012). For this knot, we used DLO #4. Another case is a reef knot, a prevalent knot widely used in for various applications including shoelaces, packaging, sewing, etc. When tying the reef knot, we used DLOs #2 and #4. Although these two DLOs have totally different material properties, our generalizable neural controller allows two robots to deploy both DLOs accurately along the designed patterns. With the help of the deployed patterns, reef knots can be tied with simple pick-and-place procedures. Such knot-tying cases strongly support the potential of our deployment scheme in various engineering applications.

We show the results of the knot-tying tasks in Table 9.6. The successful rate of knot-tying is remarkable. We achieve a success rate of 90 % (9 successful trials out of 10) for tying a trefoil knot and a success rate of 70 % (7 successful trials out of 10) with the optimal control method. Based on our observations, all the failure cases were caused by the rod slipping out of the gripper. In contrast, the intuitive control method achieves a success rate of 0% for both cases as the initially deployed pattern does not match the

Table 9.6: Real-world application experiment results. Success rates for the various cable placement and knot tying tasks are shown.

Experiment Type	Scheme	Success Rate
"S" Cable Placement	INT	0/10
	OPT	9/10
"U" Cable Placement	INT	0/10
	OPT	9/10
Trefoil Knot	INT	0/10
	OPT	9/10
Reef Knot	INT	0/10
	OPT	7/10

intended pattern.

Therefore, the intuitive control method would require some visual feedback to choose the pick-and-place motion adaptively for the trefoil knot case. As for the reef knot case, due to the deployment results are totally wrong, even though the visual feedback is applied, it is still hard to achieve a complete reef knot with intuitive method.

Therein, we can see the potential of the deployment scheme in high-level robotic tasks like knot tying. In future work, the optimal deployment scheme will be incorporated with the perception system to automatically tie any prescribed knots with the robotics system.

9.7 Conclusion

In this chapter, we have introduced a novel deployment scheme that allows for robust and accurate control of the shape of DLOs using a single manipulator. Our framework integrates techniques from various disciplines, including physical simulation, machine learning, and scaling analysis, and has been demonstrated to be highly effective in real robotic experiments. Our results highlight the advantages of incorporating physics into robotic manipulation schemes and showcase impressive performance on complex tasks such as writing letters with elastic rods, cable placement, and tying knots.

Looking to the future, we plan to leverage the precision and efficiency of our deployment scheme to tackle some high-level robotic tasks systematically, for example, robotic knot tying. While exact shape control is not strictly required during such manipulations, our deployment scheme offers sufficient accuracy and efficiency to design the configurations of the middle states of a manipulated DLO, which is essential for robots to successfully tie complex knots. We also aim to explore the use of generalized problem formulations and data-driven control schemes, such as reinforcement learning, to develop more flexible and adaptive solutions to the challenges of robotic manipulation. By continuing to push the boundaries of robotic manipulation, we hope to advance the state of the art in this field and enable new and exciting applications of robotic technology.

CHAPTER 10

Conclusions

10.1 Summary

In this thesis, we have presented robust, physically insightful solutions in the key areas of simulation, perception, and robotic manipulation, all with an emphasis on deformable structures.

In the first half of the thesis, we focused on creating physically realistic and efficient simulation frameworks for slender deformable structures. Using discrete differential geometry-based frameworks to simulate elasticity, we focused on formulating accurate elastic contact and friction (IMC). Afterwards, we ambitiously developed the first general purpose DDG-based simulation framework for the robotics community (DisMech) capable of hyperaccurate real2sim modelling of soft robots. For all our simulation-based work, we provided rigorous validation in terms of physical accuracy as well as extensive analysis on runtime improvements over the state of the art.

In the second half of the thesis, we directed our focus on difficult problems in robotic deformable object manipulation and perception. We used a combination of several disciplines spanning physical simulation, scaling analysis, optimization, and machine learning in order to come up with full sim2real solutions for robotic paper sheet folding and DLO deployment. For the DLO deployment, we also introduced a novel DLO detection algorithm that outperformed the state of the art using a physically intuitive bending energy optimization scheme. Overall, superhuman performance was achieved for all deformable object manipulation tasks.

10.2 Future Work

The work presented in this thesis opens up many exciting avenues. For elastic contact and friction, promising avenues include developing solutions for faster, more reliant convergence. More reliant convergence is especially important for hard impact (i.e., high velocity) and high friction settings, scenarios notoriously known for being difficult to simulate accurately. For DisMech, the aim to incorporate shell structures into DisMech is very promising as well as its expansion for developing soft robot control algorithms. As time progresses, DisMech's growth via the open-source community will also be something we strive to nurture.

With regard to our work in physically insightful robotic deformable object manipulation, a shift from highly specific solutions to more general AI strategies will be immensely important. Though we developed robust general solutions with respect to object material and geometry, our solutions were strictly formulated for the tasks of paper sheet folding and DLO deployment. Therefore, switching to more general task-agnostic solutions will be crucial to developing intelligent AI capable of scaling to wider problem domains. Such solutions will most likely involve incorporating reinforcement learning and foundation models, both areas that we plan to pursue.

REFERENCES

Alart, P. and Curnier, A. (1991). A mixed formulation for frictional contact problems prone to newton like solution methods. *Computer methods in applied mechanics and engineering*, 92(3):353–375. 7

Alora, J. I., Cenedese, M., Schmerling, E., Haller, G., and Pavone, M. (2023). Data-Driven Spectral Submanifold Reduction for Nonlinear Optimal Control of High-Dimensional Robots. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2627–2633. 9

Audoly, B., Clauvelin, N., Brun, P.-T., Bergou, M., Grinspun, E., and Wardetzky, M. (2013). A discrete geometric approach for simulating the dynamics of thin viscous threads. *Journal of Computational Physics*, 253:18–49. 21

Audoly, B., Clauvelin, N., and Neukirch, S. (2007). Elastic knots. *Physical Review Letters*, 99(16):164301. 45, 47, 74, 75, 130

Audoly, B. and Pomeau, Y. (2010). *Elasticity and geometry: from hair curls to the non-linear response of shells*. Oxford University Press. 21

Baek, C. and Reis, P. M. (2019). Rigidity of hemispherical elastic gridshells under point load indentation. *Journal of the Mechanics and Physics of Solids*, 124:411–426. 21

Baek, C., Sageman-Furnas, A. O., Jawed, M. K., and Reis, P. M. (2018). Form finding in elastic gridshells. *Proceedings of the National Academy of Sciences*, 115(1):75–80. 21

Balkcom, D. J. and Mason, M. T. (2008). Robotic origami folding. *The International Journal of Robotics Research*, 27(5):613–627. 10

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM. 21

Batty, C., Uribe, A., Audoly, B., and Grinspun, E. (2012). Discrete viscous sheets. *ACM Transactions on Graphics (TOG)*, 31(4):113. 21

Becker, K., Teeple, C., Charles, N., Jung, Y., Baum, D., Weaver, J. C., Mahadevan, L., and Wood, R. (2022). Active entanglement enables stochastic, topological grasping. *Proc Natl Acad Sci U S A*, 119(42):e2209819119. 86, 93

Bergou, M., Audoly, B., Vouga, E., Wardetzky, M., and Grinspun, E. (2010). Discrete viscous threads. *ACM Trans. Graph.*, 29(4). 9, 18, 21, 22, 85

Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E. (2008). Discrete elastic rods. In *ACM SIGGRAPH '08 Conference*, pages 63:1–12. 9, 10, 18, 21, 85

Bertrand, F., Leclaire, L.-A., and Levecque, G. (2005). Dem-based models for the mixing of granular materials. *Chemical Engineering Science*, 60(8-9):2517–2531. 6

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). Yolact: Real-time instance segmentation. In *IEEE/CVF International Conference on Computer Vision*, pages 9157–9166. 14

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2020). Yolact++: Better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):1108–1121. 14

Borri-Brunetto, M., Chiaia, B., and Ciavarella, M. (2001). Incipient sliding of rough surfaces in contact: a multiscale numerical analysis. *Computer methods in applied mechanics and engineering*, 190(46-47):6053–6073. 6

Buckham, B., Driscoll, F. R., and Nahon, M. (2004). Development of a finite element cable model for use in low-tension dynamics simulation. *J. Appl. Mech.*, 71(4):476–485. 18

Caporali, A., Galassi, K., Zanella, R., and Palli, G. (2022a). FASTDLO: Fast deformable linear objects instance segmentation. *IEEE Robotics and Automation Letters*, 7(4):9075–9082. 15, 16, 131, 134, 136, 140, 145

Caporali, A., Galassi, K., Žagar, B. L., Zanella, R., Palli, G., and Knoll, A. C. (2023a). Rt-dlo: Real-time deformable linear objects instance segmentation. *IEEE Transactions on Industrial Informatics*, pages 1–10. 15, 16, 141, 145, 148

Caporali, A., Pantano, M., Janisch, L., Regulin, D., Palli, G., and Lee, D. (2023b). A weakly supervised semi-automatic image labeling approach for deformable linear objects. *IEEE Robotics and Automation Letters*, 8(2):1013–1020. 15

Caporali, A., Zanella, R., Greogrio, D. D., and Palli, G. (2022b). Ariadne+: Deep learning–based augmented framework for the instance segmentation of wires. *IEEE Transactions on Industrial Informatics*, 18(12):8607–8617. 15, 16, 145

Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., and Yan, Y. (2020). Blendmask: Top-down meets bottom-up for instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8573–8581. 14

Choe, B., Choi, M. G., and Ko, H.-S. (2005). Simulating complex hair with robust collision handling. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–160. 30

Choi, A., Jawed, M. K., and Joo, J. (2022). Preemptive motion planning for human-to-robot indirect placement handovers. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4743–4749. xx

Choi, A., Jing, R., Sabelhaus, A., and Jawed, M. K. (2023a). Dismech: A discrete differential geometry-based physical simulator for soft robots and structures. xx, 3, 22

Choi, A., Tong, D., Jawed, M. K., and Joo, J. (2021). Implicit Contact Model for Discrete Elastic Rods in Knot Tying. *Journal of Applied Mechanics*, 88(5):051010. xix, 2, 7, 8, 9, 22, 53, 56, 58, 130

Choi, A., Tong, D., Park, B., Terzopoulos, D., Joo, J., and Jawed, M. K. (2023b). mbest: Realtime deformable linear object detection through minimal bending energy skeleton pixel traversals. *IEEE Robotics and Automation Letters*, 8(8):4863–4870. xx, 4, 150, 167, 169

Choi, A., Tong, D., Terzopoulos, D., Joo, J., and Jawed, M. K. (2023c). Deep learning of force manifolds from the simulated physics of robotic paper folding. xx, 3, 18, 84, 94, 165

Coevoet, E., Morales-Bieze, T., Largilliere, F., Zhang, Z., Thieffry, M., Sanz-Lopez, M., Carrez, B., Marchal, D., Goury, O., Dequidt, J., and Duriez, C. (2017). Software toolkit for modeling, simulation, and control of soft robots. *Advanced Robotics*, 31:1–17. 9, 85

Cortez, R. (2018). Regularized stokeslet segments. *Journal of Computational Physics*, 375:783–796. 65

Coumans, E. and Bai, Y. (2016–2021). Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`. 8

Crowell, R. H. and Fox, R. H. (2012). *Introduction to knot theory*, volume 57. Springer Science & Business Media. 181

Daviet, G. (2020). Simple and scalable frictional contacts for thin nodal objects. *ACM Transactions on Graphics (TOG)*, 39(4):61–1. 7

Daviet, G., Bertails-Descoubes, F., and Boissieux, L. (2011). A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–12. 7

De Gregorio, D., Palli, G., and Di Stefano, L. (2018). Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation. *arXiv preprint arXiv:1810.04461*. 14, 16

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., and Katam, H. (2019). BlenderProc. *arXiv preprint arXiv:1911.01911*. 15

Doumanoglou, A., Kargakos, A., Kim, T.-K., and Malassiotis, S. (2014). Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In *2014 IEEE international conference on robotics and automation*, pages 987–993. IEEE. 11

Doumanoglou, A., Stria, J., Peleka, G., Mariolis, I., Petrik, V., Kargakos, A., Wagner, L., Hlaváč, V., Kim, T.-K., and Malassiotis, S. (2016). Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478. 11

Durville, D. (2012). Contact-friction modeling within elastic beam assemblies: an application to knot tightening. *Computational Mechanics*, 49(6):687–707. 52

Elbrechter, C., Haschke, R., and Ritter, H. (2012). Folding paper with anthropomorphic robot hands using real-time physics-based modeling. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 210–215. IEEE. 13

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, pages 226–231. 137

Faure, F., Duriez, C., Delingette, H., Allard, J., Gilles, B., Marchesseau, S., Talbot, H., Courtecuisse, H., Bousquet, G., Peterlik, I., and Cotin, S. (2012). *SOFA: A Multi-Model Framework for Interactive Physical Simulation*, pages 283–321. Springer Berlin Heidelberg, Berlin, Heidelberg. 8, 9

Gao, Y., Lucas, B. N., Hay, J. C., Oliver, W. C., and Pharr, G. M. (2006). Nanoscale incipient asperity sliding and interface micro-slip assessed by the measurement of tangential contact stiffness. *Scripta materialia*, 55(7):653–656. 6

Gazzola, M., Dudte, L., McCormick, A., and Mahadevan, L. (2018). Forward and inverse problems in the mechanics of soft filaments. *Royal Society open science*, 5(6):171628. 8, 9, 10, 85, 86, 87, 97

Geblinger, N., Ismach, A., and Joselevich, E. (2008). Self-organized nanotube serpentines. *Nature nanotechnology*, 3(4):195–200. 150

Graule, M. A., McCarthy, T. P., Teeple, C. B., Werfel, J., and Wood, R. J. (2022). Somogym: A toolkit for developing and evaluating controllers and reinforcement learning algorithms for soft robots. *IEEE Robotics and Automation Letters*, 7(2):4071–4078. 9, 85

Graule, M. A., Teeple, C. B., McCarthy, T. P., St. Louis, R. C., Kim, G. R., and Wood, R. J. (2021). Somo: Fast and accurate simulations of continuum robots in complex environments. In *2021 IEEE International Conference on Intelligent Robots and Systems (IROS)*, page In Review. IEEE. 8, 9

Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. (2003). Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, page 62–67, Goslar, DEU. Eurographics Association. 21, 86, 97

Guler, P., Pauwels, K., Pieropan, A., Kjellström, H., and Kragic, D. (2015). Estimating the deformability of elastic materials using optical flow and position-based dynamics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 965–971. IEEE. 18

Han, S. M., Benaroya, H., and Wei, T. (1999). Dynamics of transversely vibrating beams using four engineering theories. *Journal of Sound and Vibration*, 225(5):935–988. 88

Haouchine, N., Kuang, W., Cotin, S., and Yip, M. (2018). Vision-based force feedback estimation for robot-assisted surgery using instrument-constrained biomechanical three-dimensional maps. *IEEE Robotics and Automation Letters*, 3(3):2160–2165. 18

Hilbert, D. and Cohn-Vossen, S. (2021). *Geometry and the Imagination*, volume 87. American Mathematical Soc. 102

Huang, W., Huang, X., Majidi, C., and Jawed, K. (2020). Dynamic simulation of articulated soft robots. *Nature Communications*, 11. 9, 84, 85

Huang, W. and Jawed, M. K. (2019). Newmark-Beta Method in Discrete Elastic Rods Algorithm to Avoid Energy Dissipation. *Journal of Applied Mechanics*, 86(8):084501. 87, 89

Huang, W. and Jawed, M. K. (2021). Numerical simulation of bundling of helical elastic rods in a viscous fluid. *Computers & Fluids*, 228:105038. 66

Iqbal, J., Khan, Z. H., and Khalid, A. (2017). Prospects of robotics in food industry. *Food Science and Technology*, 37:159–165. 130

Jawed, M., Dieleman, P., Audoly, B., and Reis, P. M. (2015a). Untangling the mechanics and topology in the frictional response of long overhand elastic knots. *Physical review letters*, 115(11):118302. 28, 74, 75, 76, 130

Jawed, M. and Reis, P. M. (2017). Dynamics of a flexible helical filament rotating in a viscous fluid near a rigid boundary. *Physical Review Fluids*, 2(3):034101. 21, 66

Jawed, M. K., Brun, P.-T., and Reis, P. M. (2015b). A geometric model for the coiling of an elastic rod deployed onto a moving substrate. *Journal of Applied Mechanics*, 82(12):121007. 21

Jawed, M. K., Da, F., Joo, J., Grinspun, E., and Reis, P. M. (2014). Coiling of elastic rods on rigid substrates. *Proceedings of the National Academy of Sciences*, 111(41):14663–14668. 9, 21

Jawed, M. K., Hadjiconstantinou, N., Parks, D., and Reis, P. (2018a). Patterns of carbon nanotubes by flow-directed deposition on substrates with architectured topographies. *Nano Lett.* 22

Jawed, M. K., Khouri, N. K., Da, F., Grinspun, E., and Reis, P. M. (2015c). Propulsion and instability of a flexible helical rod rotating in a viscous fluid. *Physical review letters*, 115(16):168101. 21, 22, 66

Jawed, M. K., Novelia, A., and O'Reilly, O. M. (2018b). *A Primer on the Kinematics of Discrete Elastic Rods*. Springer. 21

Jawed, M. K. and Reis, P. M. (2014). Pattern morphology in the elastic sewing machine. *Extreme Mechanics Letters*, 1:76–82. 21

Jawed, M. K. and Reis, P. M. (2016). Deformation of a soft helical filament in an axial flow at low reynolds number. *Soft Matter*, 12(6):1898–1905. 21

Jean, M. and Moreau, J. J. (1987). Dynamics in the presence of unilateral contacts and dry friction: a numerical approach. In *Unilateral Problems in Structural Analysis—2*, pages 151–196. Springer. 7

Jean, M. and Moreau, J. J. (1992). Unilaterality and dry friction in the dynamics of rigid body collections. In *1st Contact Mechanics International Symposium*, pages 31–48. 7

Jia, Y.-B., Guo, F., and Lin, H. (2014). Grasping deformable planar objects: Squeeze, stick/slip analysis, and energy-based optimalities. *The International Journal of Robotics Research*, 33(6):866–897. 12

Johanns, P., Grandgeorge, P., Baek, C., Sano, T. G., Maddocks, J. H., and Reis, P. M. (2021). The shapes of physical trefoil knots. *Extreme Mechanics Letters*, 43:101172. 131

Johnson, K. and Greenwood, J. (1997). An adhesion map for the contact of elastic spheres. *Journal of colloid and interface science*, 192(2):326–333. 6

Johnson, K. L., Kendall, K., and Roberts, a. (1971). Surface energy and the contact of elastic solids. *Proceedings of the royal society of London. A. mathematical and physical sciences*, 324(1558):301–313. 6

Kaufman, D. M., Tamstorf, R., Smith, B., Aubry, J.-M., and Grinspun, E. (2014). Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Transactions on Graphics (TOG)*, 33(4):1–12. 7

Kaufmann, P., Martin, S., Botsch, M., and Gross, M. (2009). Flexible simulation of deformable models using discontinuous galerkin fem. *Graphical Models*, 71(4):153–167. 18

Keipour, A., Bandari, M., and Schaal, S. (2022). Deformable one-dimensional object detection for routing and manipulation. *IEEE Robotics and Automation Letters*, 7(2):4329–4336. 14

Kim, H. K. H., Bourne, D., Gupta, S., and Krishnan, S. S. (1998). Automated process planning for robotic sheet metal bending operations. *Journal of Manufacturing Systems*, 17(5):338 – 360. 10

Kim, M., Bird, J. C., Van Parys, A. J., Breuer, K. S., and Powers, T. R. (2003). A macroscopic scale model of bacterial flagellar bundling. *Proceedings of the National Academy of Sciences*, 100(26):15481–15485. 71, 73

Kirchhoff, G. (1859). Uber das gleichgewicht und die bewegung eines unendlich dunnen elastischen stabes. *J. Reine Angew. Math.*, 56:285–313. 9, 23

Kita, Y., Kanehiro, F., Ueshiba, T., and Kita, N. (2011). Clothes handling based on recognition by strategic observation. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 53–58. IEEE. 11, 18

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3. 8

Lam, S. K., Pitrou, A., and Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA. Association for Computing Machinery. 50

Largilliere, F., Verona, V., Coevoet, E., Sanz-Lopez, M., Dequidt, J., and Duriez, C. (2015). Real-time control of soft-robots using asynchronous finite element modeling. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2550–2555. 9

Lattanzi, D. and Miller, G. (2017). Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, 23(3):04017004. 179

Lee, A. X., Gupta, A., Lu, H., Levine, S., and Abbeel, P. (2015a). Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5265–5272. 11

Lee, A. X., Huang, S. H., Hadfield-Menell, D., Tzeng, E., and Abbeel, P. (2014). Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4402–4407. IEEE. 17

Lee, A. X., Lu, H., Gupta, A., Levine, S., and Abbeel, P. (2015b). Learning force-based manipulation of deformable objects from multiple demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 177–184. IEEE. 11

Lee, R., Hamaya, M., Murooka, T., Ijiri, Y., and Corke, P. (2021). Sample-efficient learning of deformable linear object manipulation in the real world through self-supervision. *IEEE Robotics and Automation Letters*, 7(1):573–580. 19

Li, J., Daviet, G., Narain, R., Bertails-Descoubes, F., Overby, M., Brown, G. E., and Boissieux, L. (2018). An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–15. 27, 33, 55

Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., and Kaufman, D. M. (2020a). Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM Transactions on Graphics (TOG)*, 39(4). 8, 57, 58, 64, 65, 68

Li, X., Huang, W., and Jawed, M. K. (2020b). A discrete differential geometry-based approach to numerical simulation of timoshenko beam. *Extreme Mechanics Letters*, 35:100622. 97

Li, Y., Yue, Y., Xu, D., Grinspun, E., and Allen, P. K. (2015). Folding deformable objects using predictive simulation and trajectory optimization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6000–6006. 12

Lin, X., Wang, Y., Olkin, J., and Held, D. (2021). Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. *arXiv*. 11, 84

Lumelsky, V. J. (1985). On fast computation of distance between line segments. *Inf. Process. Lett.*, 21:55–61. 27, 30, 31, 34, 58

Lv, N., Liu, J., and Jia, Y. (2022). Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations. *IEEE Transactions on Robotics*. 18, 19, 149

Macklin, M., Müller, M., and Chentanez, N. (2016). Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54. 18

Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12. 18

Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE. 11, 17

Majidi, C. (2014). Soft robotics: A perspective—current trends and prospects for the future. *Soft Robotics*, 1(1):5–11. 8

Mandal, S., Nicolas, M., and Pouliquen, O. (2020). Insights into the rheology of cohesive granular media. *Proceedings of the National Academy of Sciences*, 117(15):8366–8373. 6

Matas, J., James, S., and Davison, A. J. (2018). Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR. 11, 19

Mathew, A. T., Hmida, I. M. B., Armanini, C., Boyer, F., and Renda, F. (2022). Sorosim: A matlab toolbox for hybrid rigid-soft robots based on the geometric variable-strain approach. *IEEE Robotics & Automation Magazine*, pages 2–18. 8, 9, 85, 86, 97

Maugis, D. (1992). Adhesion of spheres: the jkr-dmt transition using a dugdale model. *Journal of colloid and interface science*, 150(1):243–269. 6

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., and Schmidhuber, J. (2008). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537. 130

McConachie, D., Dobson, A., Ruan, M., and Berenson, D. (2020). Manipulating deformable objects by interleaving prediction, planning, and control. *The International Journal of Robotics Research*, 39(8):957–982. 149

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. (2017). Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103. 35

Miller, S., van den Berg, J., Fritz, M., Darrell, T., Goldberg, K., and Abbeel, P. (2012). A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31(2):249–267. 11

Mitrano, P., McConachie, D., and Berenson, D. (2021). Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170. 149

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118. 18

Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., and Levine, S. (2017). Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE. 19, 149

Namiki, A. and Yokosawa, S. (2015). Robotic origami folding with dynamic motion primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5623–5628. IEEE. 13

Naughton, N., Sun, J., Tekinalp, A., Parthasarathy, T., Chowdhary, G., and Gazzola, M. (2021). Elastica: A compliant mechanics environment for soft robotic control. *IEEE Robotics and Automation Letters*, 6(2):3389–3396. 10

Pacheco Garcia, J. C., Jing, R., Anderson, M. L., Ianus-Valdivia, M., and Sabelhaus, A. P. (2023). A comparison of mechanics simplifications in pose estimation for thermally-actuated soft robot limbs. In *Smart Materials, Adaptive Structures and Intelligent Systems*. American Society of Mechanical Engineers. 86, 93, 94

Pan, K., Phani, A. S., and Green, S. (2020). Periodic folding of a falling viscoelastic sheet. *Physical Review E*, 101(1):013002. 171

Panetta, J., Konaković-Luković, M., Isvoranu, F., Bouleau, E., and Pauly, M. (2019). X-shells: A new class of deployable beam structures. *ACM Transactions on Graphics (TOG)*, 38(4):83. 21

Patil, V. P., Sandt, J. D., Kolle, M., and Dunkel, J. (2020). Topological mechanics of knots and tangles. *Science*, 367(6473):71–75. 8, 131

Petrík, V. and Kyrki, V. (2019). Feedback-based fabric strip folding. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 773–778. 11, 13, 98

Petrík, V., Smutný, V., Krsek, P., and Hlaváč, V. (2016). Physics-based model of a rectangular garment for robotic folding. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 951–956. 12, 13, 98, 99, 113, 119, 120, 123

Petrík, V., Smutný, V., and Kyrki, V. (2020). Static stability of robotic fabric strip folding. *IEEE/ASME Transactions on Mechatronics*, 25(5):2493–2500. 12, 13, 98, 99, 113, 120

Qiu, W. and Yuille, A. (2016). UnrealCV: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer. 15

Rambow, M., Schauß, T., Buss, M., and Hirche, S. (2012). Autonomous manipulation of deformable objects based on teleoperated demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2809–2814. IEEE. 11

Riley, E. E., Das, D., and Lauga, E. (2018). Swimming of peritrichous bacteria is enabled by an elastohydrodynamic instability. *Scientific reports*, 8(1):1–7. 73

Rodenborn, B., Chen, C.-H., Swinney, H. L., Liu, B., and Zhang, H. (2013). Propulsion of microorganisms by a helical flagellum. *Proceedings of the National Academy of Sciences*, 110(5):E338–E347. 66

Saha, M. and Isto, P. (2007). Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics*, 23(6):1141–1150. 149

Sanchez, J., Corrales, J.-A., Bouzgarrou, B.-C., and Mezouar, Y. (2018). Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey. *The International Journal of Robotics Research*, 37(7):688–716. 12, 18, 98, 179

Sano, T. G., Johanns, P., Grandgeorge, P., Baek, C., and Reis, P. M. (2022). Exploring the inner workings of the clove hitch knot. *Extreme Mechanics Letters*, page 101788. 131

Schegg, P., Ménager, E., Khairallah, E., Marchal, D., Dequidt, J., Preux, P., and Duriez, C. (2022). SofaGym: An open platform for reinforcement learning based on soft robot simulations. *Soft Robot*, 10(2):410–430. 9

Schulman, J., Gupta, A., Venkatesan, S., Tayson-Frederick, M., and Abbeel, P. (2013). A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4111–4117. IEEE. 18, 130

Servin, M. and Lacoursiere, C. (2008). Rigid body cable for virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):783–796. 18

She, Y., Wang, S., Dong, S., Sunil, N., Rodriguez, A., and Adelson, E. (2021). Cable manipulation with a tactile-reactive gripper. *The International Journal of Robotics Research*, 40(12-14):1385–1401. 17, 149

Shen, Z., Huang, J., Chen, W., and Bao, H. (2015). Geometrically exact simulation of inextensible ribbon. In *Computer Graphics Forum*, volume 34, pages 145–154. Wiley Online Library. 21

Shi, J. and Tomasi (1994). Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. 115

Spillmann, J. and Teschner, M. (2008). An adaptive contact model for the robust simulation of knots. In *Computer Graphics Forum*, volume 27, pages 497–506. Wiley Online Library. 7, 27, 30, 45, 49, 50, 51

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. 168

Sun, J., Peng, Z., Zhou, W., Fuh, J. Y., Hong, G. S., and Chiu, A. (2015). A review on 3d printing for customized food fabrication. *Procedia Manufacturing*, 1:308–319. 150

Sundaresan, P., Grannen, J., Thananjeyan, B., Balakrishna, A., Laskey, M., Stone, K., Gonzalez, J. E., and Goldberg, K. (2020). Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE. 19

Takizawa, M., Kudoh, S., and Suehiro, T. (2015). Method for placing a rope in a target shape and its application to a clove hitch. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 646–651. IEEE. 19, 149, 151, 152

Tang, T., Wang, C., and Tomizuka, M. (2018). A framework for manipulating deformable linear objects by coherent point drift. *IEEE Robotics and Automation Letters*, 3(4):3426–3433. 17

Teo, W. E. and Ramakrishna, S. (2006). A review on electrospinning design and nanofibre assemblies. *Nanotechnology*, 17(14):R89. 150

Terzopoulos, D. and Fleischer, K. (1988a). Deformable models. *The Visual Computer*, 4(6):306–331. 21

Terzopoulos, D. and Fleischer, K. (1988b). Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 88)*, pages 269–278. 21

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 87)*, pages 205–214. 21

Terzopoulos, D. and Qin, H. (1994). Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics (TOG)*, 13(2):103–136. 18

Thakur, S. C., Morrissey, J. P., Sun, J., Chen, J., and Ooi, J. Y. (2014). Micromechanical analysis of cohesive granular materials using the discrete element method with an adhesive elasto-plastic contact model. *Granular Matter*, 16(3):383–400. 6

Thieffry, M., Kruszewski, A., Duriez, C., and Guerra, T.-M. (2019). Control Design for Soft Robots Based on Reduced-Order Model. *IEEE Robotics and Automation Letters*, 4(1):25–32. 9

Tian, Z., Shen, C., and Chen, H. (2020). Conditional convolutions for instance segmentation. In *European Conference on Computer Vision*, pages 282–298. Springer. 14

Timoshenko, S. P. and Gere, J. M. (2009). *Theory of elastic stability.* Courier Corporation. 158

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. 8

Tong, D., Borum, A., and Jawed, M. K. (2021). Automated stability testing of elastic rods with helical centerlines using a robotic system. *IEEE Robotics and Automation Letters*, 7(2):1126–1133. 9, 14, 22, 130

Tong, D., Choi, A., Joo, J., Borum, A., and Khalid Jawed, M. (2023a). Snap Buckling in Overhand Knots. *Journal of Applied Mechanics*, 90(4):041008. xx, 9, 22

Tong, D., Choi, A., Joo, J., and Jawed, M. K. (2023b). A fully implicit method for robust frictional contact handling in elastic rods. *Extreme Mechanics Letters*, 58:101924. xx, 2, 8, 9, 22, 25, 87

Tong, D., Choi, A., Qin, L., Huang, W., Joo, J., and Jawed, M. K. (2023c). Sim2real neural controllers for physics-based robotic deployment of deformable linear objects. *The International Journal of Robotics Research*, 0(0):02783649231214553. xx, 4, 21, 84, 160

Twardon, L. and Ritter, H. (2015). Interaction skills for a coat-check robot: Identifying and handling the boundary components of clothes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3682–3688. IEEE. 11

Vepa, A., Choi, A., Nakhaei, N., Lee, W., Stier, N., Vu, A., Jenkins, G., Yang, X., Shergill, M., Desphy, M., Delao, K., Levy, M., Garduno, C., Nelson, L., Liu, W., Hung, F., and Scalzo, F. (2022). Weakly-supervised convolutional neural networks for vessel segmentation in cerebral angiography. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 585–594. xx

Wakamatsu, H., Arai, E., and Hirai, S. (2006). Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25(4):371–395. 149

Wakamatsu, H. and Hirai, S. (2004). Static modeling of linear object deformation based on differential geometry. *The International Journal of Robotics Research*, 23(3):293–311. 12

Wang, A., Kurutach, T., Liu, K., Abbeel, P., and Tamar, A. (2019). Learning robotic manipulation through visual planning and acting. *arXiv preprint arXiv:1905.04411*. 19

Whitcomb, L. L. (2000). Underwater robotics: Out of the research laboratory and into the field. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 709–716. IEEE. 150

Yan, M., Zhu, Y., Jin, N., and Bohg, J. (2020a). Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE Robotics and Automation Letters*, 5(2):2372–2379. 14, 17

Yan, W., Vangipuram, A., Abbeel, P., and Pinto, L. (2020b). Learning predictive representations for deformable objects using contrastive estimation. *arXiv preprint arXiv:2003.05436*. 12

Yang, P.-C., Sasaki, K., Suzuki, K., Kase, K., Sugano, S., and Ogata, T. (2017). Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robotics and Automation Letters*, 2(2):397–403. 11

Yin, H., Varava, A., and Kragic, D. (2021). Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803. 12, 18, 98

Yu, M., Lv, K., Zhong, H., Song, S., and Li, X. (2022a). Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach. *IEEE Transactions on Robotics*. 19

Yu, M., Zhong, H., and Li, X. (2022b). Shape control of deformable linear objects with offline and online learning of local linear deformation models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1337–1343. IEEE. 130

Zanella, R., Caporali, A., Tadaka, K., De Gregorio, D., and Palli, G. (2021). Auto-generated wires dataset for semantic segmentation with domain-independence. In *International Conference on Computer, Control and Robotics (ICCCR)*, pages 292–298. IEEE. 15

Zhang, T. Y. and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239. 134

Zhang, X., Chan, F., Parthasarathy, T., and Gazzola, M. (2019). Modeling and simulation of complex dynamic musculoskeletal architectures. *Nature Communications*, 10(1):1–12. 10

Zheng, Y., Veiga, F. F., Peters, J., and Santos, V. J. (2022). Autonomous learning of page flipping movements via tactile feedback. *IEEE Transactions on Robotics*. 11

Zhu, J., Cherubini, A., Dune, C., Navarro-Alarcon, D., Alambeigi, F., Berenson, D., Ficuciello, F., Harada, K., Kober, J., Li, X., et al. (2022). Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77. 12

Zhu, J., Navarro, B., Fraisse, P., Crosnier, A., and Cherubini, A. (2018). Dual-arm robotic manipulation of flexible cables. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 479–484. IEEE. 14, 130

Zhu, J., Navarro, B., Passama, R., Fraisse, P., Crosnier, A., and Cherubini, A. (2019). Robotic manipulation planning for shaping deformable linear objects withenvironmental contacts. *IEEE Robotics and Automation Letters*, 5(1):16–23. 149

Čertík, O. (2019). Symengine. `https://github.com/symengine/symengine.git`. 35