

On the Interplay between Mechanical and Computational Intelligence in Robot Hands

Tianjian Chen

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

© 2021

Tianjian Chen

All Rights Reserved

Abstract

On the Interplay between Mechanical and Computational Intelligence in Robot Hands

Tianjian Chen

Researchers have made tremendous advances in robotic grasping in the past decades. On the hardware side, a lot of robot hand designs were proposed, covering a large spectrum of dexterity (from simple parallel grippers to anthropomorphic hands), actuation (from underactuated to fully-actuated), and sensing capabilities (from only open/close states to tactile sensing). On the software side, grasping techniques also evolved significantly, from open-loop control, classical feedback control to learning-based policies. However, most of the studies and applications follow the one-way paradigm that mechanical engineers/researchers design the hardware first and control/learning experts write the code to use the hand.

In contrast, we aim to study the *interplay* between the mechanical and computational aspects in robotic grasping. We believe both sides are important but cannot solve grasping problems on their own, and both sides are highly connected by the laws of physics and should not be developed separately. We use the term “Mechanical Intelligence” to refer to the ability realized by mechanisms to appropriately respond to the external inputs, and we show that incorporating Mechanical Intelligence with Computational Intelligence is beneficial for grasping.

The first part of this thesis is to *derive hand underactuation mechanisms from grasp data*. The mechanical coordination in robot hands, which is one type of Mechanical Intelligence, corresponds to the concept of *dimensionality reduction* in Machine Learning. However, the resulted

low-dimensional manifolds need to be realizable using underactuated mechanisms. In this project, we first collect simulated grasp data without accounting for underactuation, apply a dimensionality reduction technique (we termed it “*Mechanically Realizable Manifolds*”) considering both pre-contact postural synergies and post-contact joint torque coordination, and finally build robot hands based on the resulted low-dimensional models. We also demonstrate a real-world application on a free-flying robot for the International Space Station.

The second part is about *proprioceptive grasping for unknown objects by taking advantage of hand compliance*. Mechanical compliance is intrinsically connected to force/torque sensing and control. In this work, we proposed a series-elastic hand providing embodied compliance and proprioception, and an associated grasping policy using a network of proportional-integral controllers. We show that, without any prior model of the object and with only proprioceptive sensing, a robot hand can make stable grasps in a reactive fashion.

The last part is about developing the Mechanical and Computational Intelligence jointly — *to co-optimize the mechanisms and control policies using deep Reinforcement Learning (RL)*. Traditional RL treats robot hardware as immutable and models it as part of the environment. In contrast, we move the robot hardware out of the environment, express its mechanics as auto-differentiable physics and connect it with the computational policy to create a unified policy, which allows RL algorithms to back-propagate gradients w.r.t both hardware and computational parameters and optimize them in the same fashion. We present a mass-spring toy problem to illustrate this idea, and also a real-world design case of an underactuated hand.

The three projects we present in this thesis are meaningful examples to demonstrate the *interplay* between the mechanical and computational aspects of robotic grasping. In the Conclusion part, we summarize some high-level philosophies and suggestions to integrate Mechanical and Computational Intelligence, as well as the high-level challenges that still exist when pushing this area forward.

Table of Contents

| | |
|---|-----|
| List of Tables | vi |
| List of Figures | vii |
| Acknowledgments | 1 |
| Chapter 1: Introduction | 1 |
| 1.1 Mechanical Intelligence | 1 |
| 1.2 Mechanical and Computational Intelligence in Robot Grasping | 5 |
| 1.3 High-Level Ideas and Contributions | 6 |
| 1.4 Thesis Organization | 8 |
| Chapter 2: Related Work | 11 |
| 2.1 Hand Design and Optimization | 11 |
| 2.2 Grasp Synergies and Underactuation | 12 |
| 2.3 Force/Torque Sensing in Hands | 13 |
| 2.4 Force-based Grasping | 14 |
| 2.5 Mechanical-Computational Co-Optimization | 16 |

I Mechanical Intelligence in Hand Synergies **18**

| | |
|---|----|
| Chapter 3: Data-driven Hand Underactuation Design using Mechanically Realizable Manifolds | 19 |
| 3.1 Dimensionality Reduction and Mechanically Realizable Manifolds | 19 |
| 3.2 Problem Formulation | 21 |
| 3.3 Problem Decomposition | 24 |
| 3.4 Optimization of Mechanically Realizable Torque Manifold | 27 |
| 3.5 Optimization of Mechanically Realizable Posture Manifold for Inter-tendon Behavior | 32 |
| 3.6 Optimization of Mechanically Realizable Posture Manifold for Intra-tendon Behavior | 35 |
| 3.7 Complete Design Method Recap | 37 |
| Chapter 4: Design Cases and Quantitative Comparisons | 39 |
| 4.1 Design Case I: Single-Motor Hand with Roll-Pitch Fingers | 39 |
| 4.2 Design Case II: Dual-Motor Hand with Roll-Pitch Fingers | 45 |
| 4.3 Design Case III: Dual-Motor Hand with Pitch-Yaw Fingers | 46 |
| 4.4 Numerical Evaluation | 48 |
| 4.5 Prototyping and Validation | 50 |
| 4.6 Discussion | 51 |
| Chapter 5: Underactuated Hands with Spring Agonists | 56 |
| 5.1 Design Matrix of Underactuation Paradigms | 56 |
| 5.2 Unmet Design Requirements for the Applications in the International Space Station | 60 |
| 5.3 Tendon Transmission Combining Different Paradigms | 61 |
| 5.4 Spring Cancellation | 62 |

| | | |
|--|---|-----------|
| 5.5 | Prototyping and Validation. | 64 |
| 5.6 | Discussion | 65 |
| Chapter 6: Hand Underactuation Design: Conclusion and Future Work | | 68 |
| 6.1 | Contributions | 68 |
| 6.2 | On-Going and Future Work | 69 |
| II Compliant Force-Based Grasping | | 72 |
| Chapter 7: A Series-Elastic-Actuated Hand with Mechanical and Computational Compliance | | 73 |
| 7.1 | Proprioception | 73 |
| 7.2 | Mechanical and Computational Adaptation: Series Elastic Actuation | 75 |
| 7.3 | SEA Module Design | 76 |
| 7.4 | Electronics | 77 |
| 7.5 | Low-Level Control | 78 |
| 7.6 | Finger and Hand Design | 79 |
| Chapter 8: Grasping with Series Elastic Actuation and Hand Proprioception | | 80 |
| 8.1 | The Proportional-Integral (PI) Network Grasping Controller | 80 |
| 8.2 | Experiments and Results | 84 |
| 8.3 | Discussion | 90 |
| Chapter 9: The <i>ROAM</i> Hand: a Highly Sensorized and Fully-Actuated Hand Designed for Learning | | 92 |
| 9.1 | Kinematic Optimization | 93 |
| 9.2 | Sensing | 93 |

| | |
|---|-----------|
| 9.3 Actuation | 94 |
| Chapter 10: Compliant Force-Based Grasping: Conclusion and Future Work | 96 |
| 10.1 Contributions | 96 |
| 10.2 On-Going and Future Work | 96 |
| III Mechanical-Computational Co-Optimization | 99 |
| Chapter 11: Hardware as Policy: Mechanical and Computational Co-Optimization using Deep Reinforcement Learning | 100 |
| 11.1 Traditional Perspective vs. Co-optimization Perspective of RL | 101 |
| 11.2 Preliminaries and Nomenclature | 103 |
| 11.3 Hardware as Policy | 105 |
| Chapter 12: Mass-Spring Toy Problems | 109 |
| 12.1 Comparison Baselines | 109 |
| 12.2 Co-Optimization of Spring Stiffnesses and Computational Policy | 110 |
| 12.3 Co-Optimization of Bar Lengths and Computational Policy | 114 |
| Chapter 13: Mechanical-Computational Co-Design of an Underactuated Hand | 117 |
| 13.1 Tendon Underactuation Model | 118 |
| 13.2 Problem Formulation | 119 |
| 13.3 Implementation Details | 120 |
| 13.4 Results | 122 |
| 13.5 Prototyping and Validation | 124 |
| Chapter 14: Hardware as Policy: Conclusion and Future Work | 126 |

| | |
|---|------------|
| 14.1 Contributions | 126 |
| 14.2 Discussion | 126 |
| 14.3 On-Going and Future Work | 127 |
| IV Conclusion | 130 |
| Chapter 15: Conclusion | 131 |
| 15.1 Contributions | 131 |
| 15.2 Suggestions for Roboticists | 133 |
| 15.3 Limitations and Improvements | 134 |
| 15.4 Long-Term Challenges and Opportunities | 136 |
| 15.5 Summary | 137 |
| References | 149 |
| Appendix A: Derivation of the Matrices of Design Case III | 150 |
| Appendix B: Details of the Experiment Protocols | 152 |

List of Tables

| | | |
|------|--|-----|
| 4.1 | Optimized parameters of Design Case I. | 43 |
| 4.2 | Optimized parameters of Design Case II. | 44 |
| 4.3 | Optimized parameters of Design Case III. | 47 |
| 4.4 | Comparison of optimization metrics. | 48 |
| 5.1 | Design matrix of tendon transmission in underactuated hands. | 58 |
| 12.1 | Optimization results of total stiffness in the toy problem. | 113 |
| 13.1 | The optimized pulley radii, joint spring stiffnesses and preload angles. | 122 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Examples of Mechanical Intelligence | 3 |
| 3.1 | Illustration of the synergy manifold. | 20 |
| 3.2 | Underactuation parameters in a joint. | 23 |
| 3.3 | Illustration of the optimization decomposition. | 26 |
| 3.4 | Summary of the optimization for Mechanically Realizable Torque Manifolds. . . . | 32 |
| 3.5 | Summary of the optimization for Mechanically Realizable Posture Manifold for inter-tendon kinematic behavior. | 35 |
| 3.6 | Summary of the optimization for Mechanically Realizable Posture Manifold for intra-tendon kinematic behavior. | 37 |
| 3.7 | Summary of the proposed three-step optimization of Mechanically Realizable Man- ifolds. | 38 |
| 4.1 | The <i>Astrobee</i> free-flying robot in ISS. | 40 |
| 4.2 | (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case I. | 40 |
| 4.3 | sample grasps and the opening configuration created in <i>GraspIt!</i> simulator for Design Case I. | 42 |
| 4.4 | (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case II. | 45 |
| 4.5 | (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case III. | 46 |

| | | |
|------|---|----|
| 4.6 | The visualization of the Mechanically Realizable Torque Manifolds. | 49 |
| 4.7 | The visualization of the Mechanically Realizable Posture Manifolds. | 50 |
| 4.8 | Finger trajectory and the types of grasp along the trajectory of Design Case I. . . . | 51 |
| 4.9 | Finger trajectory of Design Case II. | 52 |
| 4.10 | Example grasps using the hand prototypes. | 53 |
| 5.1 | The tendon routing design of the proposed underactuated hand | 62 |
| 5.2 | One-dimensional analogy of the tendon-spring system to illustrate the spring force cancellation. | 63 |
| 5.3 | The plot of torques from springs on the motor shaft. | 64 |
| 5.4 | The example grasps with different objects. | 65 |
| 5.5 | The power-off pose-keeping behavior in different hand configurations | 66 |
| 5.6 | Power-off human intervention | 66 |
| 5.7 | Preliminary integration test with the <i>Astrobee</i> free-flying robot. | 67 |
| 6.1 | Desired (non-uniform) finger rolling (adduction) vs. uniform finger rolling using a linear manifold. | 70 |
| 6.2 | Non-circular pulley | 70 |
| 7.1 | The SEA module. | 76 |
| 7.2 | The prediction-vs-truth plot of SEA torque estimation. | 76 |
| 7.3 | The electronics architecture. | 77 |
| 7.4 | The low-level position control. | 78 |
| 7.5 | The low-level torque control. | 78 |
| 7.6 | The schematic and photo of the SEA hand. | 79 |

| | | |
|------|--|-----|
| 8.1 | Illustration and analogy of grasping with SEA. | 81 |
| 8.2 | The PI Network Grasping Controller. | 84 |
| 8.3 | Typical scenarios in the fingertip grasp experiments. | 86 |
| 8.4 | Experiment results of fingertip grasping. | 87 |
| 8.5 | Grasps of different objects with random initial touch poses. | 88 |
| 8.6 | In-hand manipulation tasks. | 89 |
| 9.1 | The <i>ROAM</i> Hand kinematics optimized using GraspIt! simulator. | 93 |
| 9.2 | The parameterization of the <i>ROAM</i> Hand finger locations. | 93 |
| 9.3 | The <i>ROAM</i> Hand torque-sensed actuation unit. | 94 |
| 9.4 | The tendon transmission of the <i>ROAM</i> Hand. | 94 |
| 9.5 | The <i>ROAM</i> Hand prototype. | 95 |
| 10.1 | The MuJoCo simulation environment for the blind bin-picking task. | 98 |
| 11.1 | Hardware as Policy overview. | 102 |
| 11.2 | Hardware as Policy-Minimal Implementation. | 108 |
| 12.1 | The mass-spring system — optimizing spring stiffnesses and computational policy. | 110 |
| 12.2 | The computational graphs for the toy problem — optimizing spring stiffnesses and computational policy. | 112 |
| 12.3 | The training curves of the toy problem — optimizing spring stiffnesses and computational policy. | 113 |
| 12.4 | The mass-spring system — optimizing the bar lengths and computational policy. | 115 |
| 12.5 | The computational graphs for the toy problem — optimizing bar lengths and computational policy. | 115 |

| | | |
|------|---|-----|
| 12.6 | The training curves of the toy problem — optimizing bar lengths and computational policy. | 116 |
| 13.1 | The Underactuated hand co-design optimization problem. | 118 |
| 13.2 | The computational graphs for the hand mechanical-computational co-design problem. | 121 |
| 13.3 | Training curves for the hand co-design problem. | 123 |
| 13.4 | Successful grasps in simulation and on a physical hand. | 124 |
| 13.5 | Policy deployment on a real robot. | 125 |
| A.1 | The 2-DoF universal proximal joint in Design Case III. | 151 |
| B.1 | Object sizes, object locations and initial touch poses. | 153 |

Acknowledgements

At the end of my PhD journey, there are a lot of grateful words to say and a lot of precious moments to recall. I spent more than five years at Columbia. Everything in the campus, including the Alma Mater, the Low Steps, the path I take to the Mudd Building every day, the red-themed closets and chairs in the lab, the riser step near the lab door that always trips new visitors, the robot hand prototypes I constantly break, all become unforgettable memories. However, what makes this journey really fantastic, are the people around me.

First and foremost, I would like to thank my PhD advisor, Matei Ciocarlie, for his long-term help and support during my PhD. I join the lab with a traditional mechanical background, and now I have become an interdisciplinary researcher. I start my PhD with no clear research topic, and now I can stand in an appropriate spot in this field. None of these can be achieved without Matei's support and guidance. In addition to the technical aspects, I also learn from him to have big thinking. "What is the ultimate problem?", "How does this technology change the field of robotics?" — these are the questions he always asks me, and these will be the questions I will always ask myself. I do feel honored and lucky to be a student of Matei. As an old Chinese saying goes, "A teacher for one day is a teacher for the whole life", I hope to bring what I learned from him into the future life journey.

I would like to express my gratitude to my committee, Prof. Hod Lipson, Matei Ciocarlie, Peter Allen, Kenneth Salisbury, and Aaron Dollar, for reading my thesis and providing valuable feedback. I could not have asked for a better group of experts to review the work in this dissertation.

I am very glad I spend my PhD years with my great labmates. Five of us (Max, Cassie, Sangwoo, Pedro and me) joined together as Matei's first batch of PhD students, and I really appreciate their help in terms of grasp theory, learning, electronics and so on. Emily, Gagan, Zhanpeng and Ava joined later and it is also a pleasure to work with them. I especially want to thank Zhanpeng and Max for collaborating with me in various projects. I also enjoyed talking with the postdocs in our lab — Long and Bing — for their insightful opinions. Besides, there are a lot of great master and undergrad students who worked with me, to name a few, Tianyi, Aditya, Yang, Abhijeet, Jerry, Jinzhao, and I would like to thank their contributions to the projects. I sincerely wish everyone can have a great future.

I would also like to thank the major funding source of my projects — the NASA Early Stage Innovations (ESI) program. This is a cool opportunity to get closer to one of my childhood dream — the outer space. Even though it still requires a lot of extra re-design and validation before we can actually launch our robot hand into space, I still feel fascinated and honored to contribute a small part to this great mission.

I also want to thank my Mom and Dad and all family members for their unconditional love, understanding, support and life advice. Finally, thanks to my girlfriend, Xixi, who cheers me up every day, calms me down when I encounter difficulties, sits by my side every day during the quarantine time, and enjoys every meal with me after a whole day of work. Having you in my life is another wonderful thing apart from academics during my PhD years.

2020 is a year that will be written into the history book. We all experienced anxiety and uncertainty due to the global pandemic and economic recession. However, now comes the year 2021 — a brand new year with hope. I feel lucky that I can finish my PhD in time, and I feel even luckier to find the field I truly love and want to devote myself to. Thanks to all the people who helped me during my PhD, and I will keep going.

Chapter 1: Introduction

1.1 Mechanical Intelligence

Nowadays, robots have already been widely used in factory settings (e.g. known environments, pre-programmed actions) to perform repeated tasks. However, even though researchers and engineers already made great advances to make robots “automatic”, it is still inappropriate to claim such robots are “intelligent”. In unstructured (with no or only limited prior models) and uncertain (with the presence of perception error) environments, such as homes, there are still a lot of challenges for robotic applications. Robot “intelligence” is truly needed in these scenarios but not fully achieved yet.

Even though the exact definition of robot “*intelligence*” is a sophisticated philosophical question and may vary in different contexts, one fundamental meaning is that the agent can accept input information and automatically produce appropriate actions in order to achieve certain goals [1, 2].

Different from pure computing devices that can run intelligent algorithms, robots, on the other hand, are expected to physically interact with the environments. This requires the “embodied intelligence” — the ability to not only “think” but also “feel” and “act” in the physical world, i.e. the ability to automatically generate appropriate physical responses to internal and external stimuli.

In contrast to the common idea of “*Computational Intelligence*” which uses algorithms to generate such responses, we take the term “*Mechanical Intelligence*” [3] to refer to such ways of responses implemented in robot mechanisms.

The history of “Mechanical Intelligence” is actually much longer than its computational counterpart, and there are some notable examples. Here we give some examples of different types of “Mechanical Intelligence” as follows:

- *Mechanical coordination or synergies*. This commonly refers to mechanisms that can couple

the movements (and force/torque generation in some cases) of different parts or degrees-of-freedom (DoFs) in a system. The wind-powered linkage-driven walker (Fig. 1.1a) is a good example of this category, where the movements of each joint are coupled by the multi-bar linkages.

- *Mechanical compliance.* This type of Mechanical Intelligence is based on compliant self-adaptive mechanisms or materials that can passively conform to the environment it is interacting with. The jamming gripper (Fig. 1.1b) [4] is a good example here, where the granular materials inside the membrane can first adapt to the shape of the object in hand passively, and then get jammed by vacuum to apply grasping forces.
- *Mechanical feedback.* This means a mechanism is designed as a “sensor” to provide feedback. A well-known example in this category is the Watt’s centrifugal governor (Fig. 1.1c) — when the engine rotation speed goes higher, the “flying balls” are raised which will, in turn, close the aperture of the throttle valve and lower the engine speed.
- *Mechanisms designed with special dynamic properties.* This category contains a variety of sub-cases. For example, researchers discovered that a proper serial robot mechanism design can linearize the Equation of Motion (EoM) of the system which significantly simplified the controller design [5]. For another example, a passive bipedal walker is designed with a limit-cycle that is suitable for dynamic walking [6] (Fig. 1.1d).

Compared to Computational Intelligence with the “sense-compute-actuate” paradigm, we believe Mechanical Intelligence has the following advantages in many scenarios:

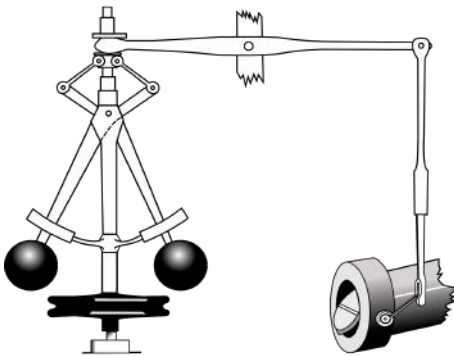
- *Reduction of run-time computation.* Since the actions are generated by mechanisms, the run-time computation can be reduced. Instead, a lot of computation can be moved to the design-time optimization of the mechanisms. We believe this effort is still valuable even though the computers nowadays are fast, because (i) the physical interactions with the environment require very fast real-time computation, and (ii) a lot of low-level controllers are low-power embedded devices that have limited computing capabilities.



(a) Wind-powered walker



(b) Jamming gripper



(c) Watt's centrifugal governor for steam engines



(d) Passive dynamic walking robot

Figure 1.1: Examples of Mechanical Intelligence

- *Reduction on the number of sensors, processors and actuators.* Sensors, processors and actuators will increase cost, and actuators are usually heavy and bulky. The use of intelligent mechanisms can help in certain cases where the size, weight and cost are crucial.

A specific advantage under this point is the *passive adaptation*, even without the need for sensors, processors and actuators. For example, compliant materials and mechanisms can automatically conform to the geometry of the objects they are interacting with, differential mechanisms (e.g. differential gears in automobiles) can automatically adjust the distribution of outputs based on loads, etc.

- *Better Reliability and Robustness.* Assuming a good design, hardware implementation of

intelligent behaviors can be more reliable and robust in many cases. For example, in safety-critical domains with extreme conditions such as aerospace applications, it is not rare that people choose to realize certain functionalities in a mechanical fashion instead of the “sense-compute-actuate” paradigm.

On the other hand, we have to admit that the mechanical realization of intelligence also has limitations, and the bullet points above generally hold true but not always. For example, a strict mechanical coupling (such as a gear transmission) between different DoFs means these DoFs can not be controlled independently even when needed; mechanisms do not always save weight and spaces since they also have weights and take spaces; if the cost is not a big concern, modern sensors and actuators can also be very reliable, so mechanical solutions may not always have advantages for reliability. We, as designers, need to evaluate such trade-offs.

The synthesis of intelligent mechanisms requires extra effort in design time. In order to achieve the desired trajectories/behaviors/performances, optimization techniques are commonly utilized. For example, researchers optimize for mechanism parameters in [7, 8, 9, 10, 11] and optimize for mechanism topology in [12, 13, 14]. These studies generated useful robot designs. Especially, with optimization, the “intelligence” — the appropriate way of physical responses — is embedded into the mechanisms in design time. We believe this is particularly important.

The recent influx of high-fidelity physics simulation [15, 16, 17] made it possible for the task-driven design of intelligent mechanisms. Instead of hand-engineering an abstract performance metric, we can put a parameterized virtual prototype into the simulated environment, execute the intended control policies, collect data in the simulator to evaluate the performance, and iterate and update the design. Several sim-to-real transfer techniques, such as Domain Randomization [18], can be used to increase the chance that the mechanisms optimized in simulation also work well in reality.

1.2 Mechanical and Computational Intelligence in Robot Grasping

Grasping is one of the fundamental topics in robotics research. Since the birth of the first industrial robotic manipulator – the Unimate [19] – in the 1950s, researchers and engineers have been working for decades in this area.

For years, various studies have been pushing this field forward in either the hardware or software aspects. A number of achievements have been made in the mechanical design of robotic hands, ranging from parallel jaw grippers (e.g. [20, 21, 22]) to multi-fingered dexterous hands (e.g. [23, 24, 25, 26, 27, 28, 29, 30, 31]), from underactuated (e.g. [32, 33, 34, 35, 36, 11, 37, 38]) to fully-actuated hands (e.g. [24, 25, 27, 28, 29, 30, 31]). On the other hand, as the computation techniques — especially machine learning — develop rapidly, complex motor skills (e.g. pick-and-place, tool use, in-hand manipulation) can be acquired using the state-of-the-art algorithms (e.g. [39, 40, 41, 42, 43, 44, 45]).

In contact-rich robotic grasping in unstructured and uncertain environments, the advantages of Mechanical Intelligence presented in the previous section can get amplified. Here we elaborate on this idea with some examples in the following paragraphs.

As discussed in the previous section, one major benefit of incorporating Mechanical Intelligence in grasping is that it can simplify the run-time control. If we take the “sense-compute-actuate” paradigm, the grasping planning problem is usually high-dimensional (for multi-DoF hands), non-convex and discontinuous, which is computation heavy. In contrast, using Mechanical Intelligence, such grasp planning can be significantly simplified. For example, we can make use of appropriate mechanical coordination between different joints in the hand, a.k.a. synergies. The joint space for a multi-DoF hand is high-dimensional, but only a subspace or a manifold is useful for grasping. If the relationship to express this manifold can be found (through, for example, grasp data) and realized in mechanisms, the the dimensionality of control algorithm can be significantly reduced. For another example, with appropriate compliance in fingers, a robot hand can grasp without carefully planning the hand poses, as the compliance can provide stability and also protect

the hand.

Another major benefit of Mechanical Intelligence in grasping is that it enables passive adaptation. In cases where the complete physics model (e.g. mass, friction coefficient) of the object is not available, it is important to adapt to the object, and mechanical adaptation can do the job. Besides, since adaptive grasping approaches do not require the detailed information of the object, they are also robust to modeling or perception errors. For example, with embodied hand compliance, the hand can adapt to the shape of the objects passively even in simple open-loop grasps. For another example, with the differential behavior (a.k.a. breakaway behavior) implemented in underactuated mechanisms (which means some finger links can continue to close even when other links on the same transmission get stalled), the hand can shape itself to envelop the object and create stable grasps in a passive fashion.

Of course, Mechanical Intelligence cannot solve the real-world grasp problem on its own, and the “sense-compute-actuate” paradigm — the Computational Intelligence — is of equal importance. Taking human hands for example, we have over 30 sensorized muscles providing proprioception and driving the fingers, around 17,000 touch receptors and free nerve endings in the palm [46], and more importantly, we have a highly advanced brain to process the information and give commands for all the complex tasks. It is the “Computational” Intelligence that makes our hands dexterous and versatile. Inspired by human hands and motivated by the aforementioned studies on robot hands, we think that Computational Intelligence is another goal to pursue.

We believe it is beneficial to integrate the mechanical and computational aspects of robotic grasping and manipulation, and take advantage of both well-designed mechanisms and algorithms. We posit that the mechanical and computational aspects are tightly related and it would be better not to develop them separately. We, as designers, can leverage the advantages of both worlds.

1.3 High-Level Ideas and Contributions

We believe the integration and interplay of physics and algorithms are still underexplored in the field of robotic grasping and manipulation. In many problems with perfect models such as

Go or video games, modern Artificial Intelligence (AI) already reached or even exceeded human-level performance. However, if we switch to a real-world scenario, for example, asking a robot to make a salad, things become much more challenging — on one hand, the physics of the robot’s own body and the outside world needs to be understood and utilized by the algorithm, which is a problem not fully solved yet; on the other hand, designing robot’s body to enhance their algorithmic performance is also an interesting but under-investigated area.

My PhD research focuses on the intersection area of robot hardware and computational methods. Specifically, the major part of my research is to explore the interplay between mechanical design and *data-driven/machine learning* techniques, in the context of robotic grasping and manipulation. Here are some high-level ideas we follow in this thesis:

- *We show it is useful to perform robot hardware design by leveraging the data from the environment to interact with.* Specifically, in grasping, the “environment” refers to the set of graspable objects and the “data” refers to the information of the grasps. Of course, in the process of designing a robot, we do not have a physical prototype to collect the data with. However, simulation data can be utilized. Such data can be prespecified by humans for desired behaviors (see Chapter 3), or can be collected on-the-fly using a simulated robot in the simulated environment during the design iterations (see Chapter 11).
- *We bring machine learning techniques into the design of robot hardware.* After gathering the data, we show that methods traditionally used in the learning/statistics domain can also be applied in robot hardware design. For example, we propose a "mechanical dimensionality reduction" for hand underactuation design introduced in Chapter 3, and a Reinforcement-Learning-based hardware-software co-design introduced in Chapter 11.

Specifically, our major contributions in these directions can be summarized as follows:

- *We are the first to introduce “Mechanically Realizable Manifolds” for hand underactuation design — a dimensionality reduction technique to extract useful low-dimensional synergies from grasp data and concurrently guarantee such synergies can be physically realized by*

underactuated mechanisms. This follows the aforementioned high-level idea of introducing the data-driven method — specifically unsupervised manifold learning — into the domain of robot design.

- *We are the first to present a tendon underactuation paradigm leveraging springs as prime movers (agonists) for certain joints in hands*. This design not only exhibits the desired underactuation and grasping behaviors, but also allows power-off pose-keeping and human-intervention.
- *We are the first to demonstrate an effective grasping controller that can leverage the series elasticity embodied in the hand*. We show that, only with the proprioception provided by the series elastic actuation, a robot hand can make stable fingertip and enveloping grasps, as well as some simple in-hand manipulation tasks.
- *We are the first to show a co-optimization of mechanisms and control policies using deep Reinforcement Learning (RL)*. By leveraging auto-differentiable physics, our algorithm can propagate gradients w.r.t both hardware and computational parameters using auto-differentiation and optimize them simultaneously. This follows the aforementioned high-level idea to bring RL into hardware optimization.

These contributions are encompassing the core idea of the integration and interplay of Mechanical and Computational Intelligence. We believe these are important steps towards the goal of dexterous and versatile grasping and manipulation with the power from both sides.

1.4 Thesis Organization

Part I: Mechanical Intelligence in Hand Synergies:

- In Chapter 3, we propose a data-driven method to determine low-dimensional models that can realize Mechanical Intelligence in underactuated hands. Specifically, we use simulated

grasp data (without considering underactuation), and then apply a “mechanical dimensionality reduction” technique (we termed it “*Mechanically Realizable Manifolds*”) considering both pre-contact postural synergies and post-contact joint torque coordination.

- In Chapter 4, we compare three different hands designed using the “Mechanical Realizable Manifolds” in a quantitative fashion. We also physically built the best two and evaluate the hand in the real world.
- In Chapter 5, we formalize a design matrix of the tendon underactuation schemes in hands, and present a new design combining two paradigms and leveraging springs as prime movers (a.k.a agonists) for some finger joints. We show that this design can not only be a good performer for general-purpose grasping, but also meet specific requirements — power-off pose-keeping and human-intervention — in the application in the International Space Station (ISS).

Part II: Compliant Force-Based Grasping:

- In Chapter 7, we demonstrate a hardware platform of a hand with embodied compliance and proprioceptive sensing. Specifically, we propose a hand design with Series Elastic Actuation (SEA) which can provide both passive compliance and active force-based adaptation.
- In Chapter 8, we show the force-based grasping controller associated with the Series-Elastic-Actuated hand, which can leverage the compliance and proprioception to grasp unknown objects. Specifically, we present the control policy in a format of a network of proportional-integral (PI) controllers. We also performed real-world experiments to show its effectiveness in the fingertip and enveloping grasps, as well as some simple manipulation tasks.
- In Chapter 9, we present another hand — the *ROAM* Hand — that has more dexterity and sensing capability. This is a fully-actuated three-fingered hand, equipped with both proprioception and tactile sensing. We believe this hand has great potential for learning-based manipulation tasks using force/torque information.

Part III: Mechanical-Computational Co-Optimization:

- In Chapter 11, we show a co-design technique of Mechanical and Computational Intelligence using deep Reinforcement Learning (RL). Specifically, we express hardware mechanics as auto-differentiable physics and connect it with the computational policy to create a unified policy (we term this method “*Hardware as Policy*”), which allows the RL algorithm to back-propagate gradients of the actions w.r.t both hardware and computational parameters and optimize them in the same fashion.
- In Chapter 12, we use two toy problems of mass-spring systems to illustrate the idea of the “*Hardware as Policy*” method. We also demonstrate a performance comparison with our baselines.
- In Chapter 13 we present a concrete real-world example of the “*Hardware as Policy*” method — the co-design of the mechanisms and control policy of an underactuated hand. We present both the simulation result as well as the real-world experiments.
- In Chapter 15 we summarize this thesis, provide some high-level thoughts and suggestions for integrating Mechanical and Computational Intelligence, and finally discuss some challenges and opportunities in this intersection area.

Chapter 2: Related Work

In this chapter, we list a number of previous works upon which this thesis is built. We will first cover the hand design topics related to synergistic hand discussed in Part I, subdivided into the hand design optimization, grasp synergies and underactuation. Then we list the hand force sensing and control which are linked to the topic of hand force/torque sensing and control in Part II. Finally, we discuss the previous mechanical-computational co-optimization works related to Part III.

2.1 Hand Design and Optimization

There is a long history of robotic hand design, and hundreds and thousands of papers and patents are in this area. It is difficult to list all of them, but we will show several in each category from which we get inspiration.

Fully-articulated robot hands are invented to create dexterous hand postures. Many early robotic hands belong to this paradigm, for example, Utah/MIT Hand [24], the Stanford/JPL Hand [23, 25] and the NTU Hand [26]. There are hands developed more recently in this category, especially in the subclass of anthropomorphic dexterous hands, such as DLR/HIT Hand [47], the ACT Hand [29], the Shadow Hand [28], and the Sandia Hand [30]. The fully-articulated hands are able to actively shape themselves to arbitrary desired shapes for different objects, and also dexterous enough to perform in-hand manipulation. However, due to the high dimensionality of joint space, the control is difficult and the run-time computation burden is heavy.

In contrast, researchers have developed underactuated hands that do not require sophisticated control. Hirose's soft gripper [48] is a notable early example in this category. After that, researchers and engineers developed a lot of underactuated hands, for example, the Barret Hand [3], Harvard Hand [34], iHY Hand [36], Robotiq Hand [33], Velo Gripper [11], Pisa/IIT Hand [37] and

Ocean One robot hand [38]. Underactuated hands have the advantages of simplicity and adaptivity: they only require simple control, and can adapt to the objects and make grasps by the virtue of carefully-designed mechanisms. However, underactuation does not provide as much dexterity as full actuation. Many of the hands above can only perform certain types of grasps, or lack the flexibility to choose the configuration after making the grasp.

To fulfill certain requirements of a hand design, one common way is to select parameters via optimization. There is a lot of literature in this category: for example, Birglen et al. [7] summarized the optimization studies in detail for linkage-driven underactuated hands. Dollar and Howe [8], Ciocarlie and Allen [9], Dong et al. [10] and Ciocarlie et al. [11] presented studies that optimize various aspects of tendon-driven underactuated hands. We believe, with the development of the large-scale black-box optimization techniques (e.g. Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) [49], Bayesian Optimization (BO) [50]), more and more hand designs will be optimized instead of empirically determined. These optimized features that benefit grasping become part of Mechanical Intelligence.

2.2 Grasp Synergies and Underactuation

Grasp synergies refer to the coordination of multiple degrees-of-freedom (DoFs) in a hand, providing a simple way to use the hands with high-dimensional joint spaces.

A common place to apply the idea of synergies for robotic hands is in the planning or control for fully-actuated dexterous hands. Various studies are presented in this category, though the term for synergies may be different, e.g. “eigengrasps” or “eigenpostures”. Planning in the low-dimensional subspace can significantly reduce the computation complexity of the grasp search. The studies from Ciocarlie and Allen [51] and Rosell et al. [52] are good examples of this idea. Moreover, the technique of synergy is also adopted in the (run-time) control of robotic hands, meaning the joints are commanded in a coupled fashion. For example, the work from Wimbock et al. [53] showed a synergy-level impedance controller for a multi-finger hand. One specific domain in this category is to use low-dimensional synergies as a human-robot interface for teleoperation,

learning by demonstration, and prosthetics, in order to reduce the required communication bandwidth between the human and the robot. This method has been shown in the studies from Gioioso et al. [54], Meeker et al. [55], Matrone et al. [56] [57] as well as Tsoli and Jenkins [58]. However, these studies mostly consider postural control without accounting for grasping force equilibrium.

On the other hand, since the methods above implement synergies in software, we can also transfer this idea by coupling joints together in hardware, i.e. underactuation. This leads to the mechanical realization of predefined synergies using underactuated mechanisms. For example, Brown and Asada [59] designed a mechanical implementation of Principal Component Analysis (PCA) results for a hand using pulley-slider systems to realize inter-finger coordination. Xu et al. [60] [61], Li et al. [62], Chen et al. [63], Xiong et al. [64] also proposed several studies to enable hardware synergies for anthropomorphic hands, based on different types of mechanisms such as gears, continuum mechanisms, cams, and sliders. These studies only consider postural behaviors without the notion of force, so they do not have a guarantee for grasp stability. To deal with this issue, a series of works from Gabiccini et al. [65], Prattichizzo et al. [66], Grioli et al. [67], and Catalano et al. [37], presented the concept of “soft synergies” and “adaptive synergies” and provided the models and tools for the underactuated hands with such features to account for force generation and force equilibrium. They also used this theory in the design and control of a multi-finger dexterous hand (the Pisa/IIT hand). Though all studies above present feasible ways to implement synergies mechanically, they are limited to anthropomorphic hands for which the synergies can be extracted from human data.

2.3 Force/Torque Sensing in Hands

In terms of the adaptation discussed in the Introduction chapter, in addition to passive adaptation via mechanisms, active adaptation using sensing and control is also an effective way. Furthermore, even with passively adaptive mechanisms as mechanical intelligence, various sensors can also provide rich information about the grasp, which enables run-time computational intelligence.

In this thesis, we focus on hand proprioception, especially force or torque sensing. In general,

there are three ways to sense force or torque: by motor current sensors, force/torque sensors, and series elastic actuators (SEA) [68]. Researchers have developed several robotic hands with these three principles. For example, the Shadow hand [28] and the Yale OpenHand [69] (equipped with Dynamixel servos) provide current sensing, which is a rough proxy for motor torque. The Stanford/JPL Hand [23], Robonaut Hand [27] and the DLR Hand II [70] have strain gauges or force-torque sensors embedded in their fingers. The hand of the DOMO robot [71] and the hand of the Obrero robotic platform [72] make use of Series Elastic Actuators (SEA).

2.4 Force-based Grasping

With force sensing to close the loop, a controller or policy can be executed to perform the adaptive and reactive grasp. Researchers have been exploring force-based closed-loop control as a good companion or even an alternative to the vision-based grasps.

Classical control was first used in force-based grasping. Assuming object information is available, model-based controllers can perform grasping or in-hand manipulation. For example, early work from Yoshikawa et al. presented studies (e.g. [73]) on hybrid force-position control for manipulation. Arimoto et al. [74] derived the dynamics of a dual-finger gripper and proposed a controller that can regulate the object position and orientation. These methods require complete information about the hand-object system. When the models of the objects are not available, researchers either relied on assumptions about the contacts, or used sensor-based techniques for grasping. For example, Schneider and Cannon [75] studied object impedance control using multiple manipulators. Arimoto et al. [76] and Yoshida et al. [77] studied blind grasping using two fingertips. However, these studies assume the contacts only happen at the end points or the end hemispheres of the fingertips. Wang et al. [78] proposed a controller that can search appropriate finger contact locations using haptic feedback and can switch between control modes for different surfaces. Platt et al. [79] presented a study on changing the contact configuration by following the gradient of grasping objective functions using six-axis load cell data. Hsiao et al. [80] proposed a contact-reactive method using tactile sensing to deal with uncertainty. Felip et al. [81] presented a

finite-state-machine-based blind grasping strategy using proprioception and tactile sensing.

The development of data-driven and learning methods also opens up new possibilities for force-based blind grasping. For example, Dang et al. [82] proposed a supervised learning framework to predict grasp quality from tactile data using Support Vector Machine (SVM). Murali et al. [83] uses a tactile-based particle filter to localize the object just by touching, and then execute a learned grasping policy to pick it up. The work from Wu et al. [84] presented a Reinforcement Learning framework to use tactile sensing to close the loop which improves the grasping success rate compared to the vision-based open-loop grasping.

Most of the studies above use tactile sensors, which require sophisticated hardware. In contrast, we aim to perform force-based grasping only with proprioception, which we will show in Part II.

With only proprioception, one notable category of related work is contact reasoning and estimation. Even though this category does not directly provide the control strategy, the estimated contact information can be crucial for grasping or manipulation. For hands, early works from Huber and Gruben [85] presented a model-based observer design to estimate contact location based on hand proprioception. Belzile and Birglen [86] also proposed a model-based contact estimation framework that leverages the mechanics of underactuated hands. These studies highly rely on the analytical model of the hand-object system. In addition to hands, researchers also studied proprioceptive contact estimation using multi-DoF manipulators and even humanoids. Back in the 1990s, Gordon and Townsend [87] and Eberman and Salisbury [88] discussed localization of single contact in the most distal link of the Whole-Arm-Manipulator (WAM) robot in 2D and 3D cases respectively using classical mechanics, but also with a lot of assumptions about the contacts and robot geometries. Recently, Zwiener et al. proposed a supervised-learning-based method for contact localization using proprioception from a 6-DoF manipulator. Manuelli and Tedrake [89] proposed a proprioceptive “contact particle filter” and demonstrated it in a simulated humanoid robot. Generally speaking, due to the sparsity of proprioception, it is a non-trivial task to estimate contacts from proprioception: we usually need to either make assumptions that reduce the number of unknowns, or include the information from more joints (that contribute to counter-balancing

contact forces) to increase the number of equations.

2.5 Mechanical-Computational Co-Optimization

Finally, we come to the topic of mechanical-computational co-optimization that can integrate Mechanical and Computational Intelligence seamlessly. Previous studies on this topic can be roughly divided into three categories: using analytical models and classical control, using evolutionary computation, and recent advances using reinforcement learning.

The first category of related work comprises studies using analytical dynamics and classical control [90], a number of which optimized mechanical and control or planning parameters for legged locomotors [91, 92, 93]. All studies above require an analytical model of the complete mechanical-control system, which is non-trivial in complex problems. A recent study uses classical control but directly iterates on real microrobot hardware without analytical modeling [94]. However, its goal is different from ours: this study aims to decrease the number of real-world design evaluations, which is avoided in our work in Part III of this thesis by training in simulation and transfer to the real world later.

Evolutionary computation provides another way to approach this problem. This research path originated from studies on the evolution of artificial creatures [12], where the morphology and the neural systems are both encoded as graphs and generated using genetic algorithms. Lipson and Pollack [95] introduced the automatic lifeform design technique using bars, joints, and actuators as building blocks of the morphology, with neurons attached to them as controllers. A series of works from Cheney et al. [96, 97] studied the morphology-computation co-evolution of cellular automata, in the context of locomotion. Nygaard et al. [98] presented a method that optimizes the morphology and control of quadruped robot using real-world evaluation of the robot. Evolutionary strategies, which are gradient-free, have significant promise, but also exhibit high computational complexity and data-inefficiency compared to recent gradient-based optimization methods.

The recent influx of reinforcement learning provides a new perspective on the co-optimization problem. Ha [99] augmented the REINFORCE algorithm with rewards calculated using the me-

chanical parameters. Schaff et al. [100] proposed a joint learning method to construct and control the agent, which models both design and control in a stochastic fashion and optimizes them via a variation of Proximal Policy Optimization (PPO). Vermeer et al. [101] perform two-dimensional linkage mechanism synthesis using a Decision-Tree-based mechanism representation fused with RL. Luck et al. [102] presented a method for data-efficient co-adaptation of morphology and behaviors based on Soft Actor-Critic (SAC), leveraging previous information to estimate the performance of new candidates. In all the studies above, hardware parameters are still optimized iteratively and separately from the computational policies, whereas we aim to optimize both together in a unified framework, which we will show in Part III of this thesis. In addition, none of these works show physical prototypes based on the co-optimized agent.

Recent work on auto-differentiable physics [103, 104, 105, 106, 107] is also relevant to our work. We hope to make use of advances in general differentiable physics simulation in future iterations of our method introduced in Part III of this thesis.

Part I

Mechanical Intelligence in Hand Synergies

Chapter 3: Data-driven Hand Underactuation Design using Mechanically Realizable Manifolds¹

In this and the following chapters, we focus on one type of Mechanical Intelligence of robotic hands — the coordination or synergies between joints, achieved by underactuated mechanisms.

Underactuation is a heated topic for multi-fingered hand design for several reasons. First and foremost, it can reduce the complexity of actuation: one actuator can drive several joints since they are linked mechanically, therefore the number of actuators can be greatly reduced. Second, with a proper design, underactuation can provide “intelligent” adaptive behaviors — for example, automatic conformity to object shapes, automatic grasp type selection based on initial contact, etc.

A big question to ask is: how to obtain joint coordination patterns realizable by mechanisms? Our approach is to find such patterns from grasp data. We will show the general concept and the computational details in this chapter.

3.1 Dimensionality Reduction and Mechanically Realizable Manifolds

Grasp synergies, or the correlation of multiple joints in a hand, provide an effective way to realize versatile grasping in a simple fashion. The idea of grasp synergies originates in studies of human hands [110]. By performing Dimensionality Reduction techniques (such as Principal Component Analysis (PCA)) on human grasp data, we can obtain a low-dimensional model that can explain most of the grasps, which means the most useful subspace of grasping is near a low-dimensional manifold in joint space.

This idea has also been adopted in many robotic applications, including synergistic grasp planning and control and synergistic underactuated hands. For example, synergies can be used in the

¹The work presented in this and the next chapter is published as [108] in 2018 IEEE International Conference on Robotics and Automation (ICRA) and [109] in 2020 IEEE Transactions on Robotics (T-RO)

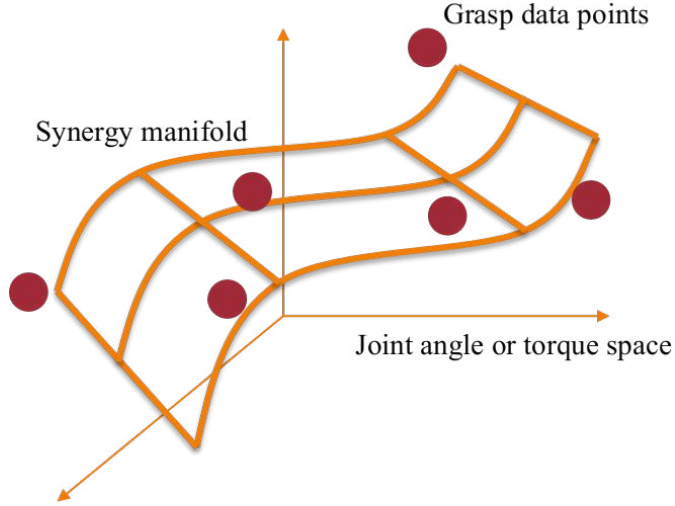


Figure 3.1: Illustration of the synergy manifold.

planning or control algorithms for robotic hands [51, 52, 53]. Synergies can also be embedded into the mechanical design of underactuated hands, moving some of the control intelligence to the hardware. We are interested in the latter in this study.

In joint posture space, a synergy of a (fully- or under-actuated) hand can be represented by a low-dimensional manifold (shown in Fig.3.1), along which the point standing for a hand configuration can slide. In joint torque space, a low-dimensional torque coordination scheme also corresponds to a manifold where the torques can take values.

We introduce the term *Mechanically Realizable Manifolds* to refer to the aforementioned manifolds in either joint angle or torque space *that can be physically realized by underactuated mechanisms*. These manifolds are parameterized based on mechanical specifications, and can be altered by changing values of the design parameters to exhibit different shapes corresponding to different hand behaviors. Conversely, if we can find good Mechanically Realizable Manifolds, we can use its parameters to build underactuated hands.

It is important to note that realizing an arbitrary manifold using mechanisms under physical constraints is a non-trivial task. One possible way is by using additional mechanical transmissions (reviewed in the Related Work chapter), which usually leads to a complex and bulky design. Another option, which is the direction we take, is to carefully design the must-have actuation

mechanisms (such as pulleys and springs for tendon transmissions) to exhibit the desired behavior. However, even though certain mechanical parameters are allowed to vary, the structure of the hand and the design choices will limit the range of manifolds that can be obtained. For example, circular pulley-tendon mechanisms always impose linear relationships between different joint torques on one tendon, and thus cannot be made to represent a nonlinear target manifold. In addition, design parameters are also limited due to real-world constraints, so the shape of the corresponding manifold can only vary in a limited range. These are examples of issues that we attempt to address in this study.

3.2 Problem Formulation

The key question we want to answer is the following: *How can we design highly underactuated hands to perform a given set of grasps?* Or in the language of data-driven methods, *how can we obtain a low-dimensional model that fits a set of grasp data points but can also be physically realized by underactuated mechanisms?*

More concretely, we start from: (i) a hand model with a known morphology and kinematic configuration (e.g. number and dimensions of fingers, tendon connectivity pattern), but *undetermined actuation parameters* (e.g. exact tendon routes, restoring spring parameters), and (ii) a set of sample grasps, created using the aforementioned hand models but *without accounting for underactuation*. Our specific goals are to (i) design the *Mechanically Realizable Posture Manifold* to approach the desired grasp postures, and (ii) design the *Mechanically Realizable Torque Manifold* to generate grasp forces close to equilibrium.

In other words, we aim to develop a “*mechanical dimensionality reduction*” technique for the set of grasp data, by extracting a useful manifold that spans the most data and can also be realized by mechanisms.

This is a realization of “Mechanical Intelligence” — when the resulted hand needs to make a new grasp close to the sample grasps, by following these manifolds, it should automatically make a reasonable grasping posture, and generate forces and torques close to equilibrium.

We elaborate on this formulation in the following paragraphs.

First of all, since the design space of a robot hand is very large, we narrow down our problem by requiring a hand model with predefined morphology and kinematics; *the unknowns are the underactuation parameters* which determine the hand motion and force generation behavior. The given hand kinematics specifies the number of fingers and links, the shape and dimensions of the fingers, the tendon connectivity pattern (specifying which tendons drive which joints), etc. The unknown parameters include tendon moment arms, restoring spring parameters, etc. We point out that we are not presenting a method to conduct the initial design for hand dimensions or kinematics, but only the (under-)actuation parameters that determine hand behavior. However, for different prespecified kinematic design options, the results computed from our optimization algorithms can be used as performance metrics to compare between them, as we will discuss in Section 4.6.

In addition, we assume we have collected a set of stable simulated grasps as desired grasp data, using the hand model with predefined kinematics. These grasps do *not* account for underactuation, i.e. the joints are considered independent and the hand can exhibit arbitrary poses as needed for grasping a certain object. Meanwhile, all of these grasps are required to have force-closure, i.e. there exists a set of contact forces that produce zero resultant wrenches on the object while satisfying friction constraints. Each desired grasp contains the information of (i) the hand pose and (ii) the set of possible contact forces and joint torques embedded in the equilibrium conditions in such a hand pose.

Our goal is to design the underactuation mechanism for a hand to (approximately) conduct a set of sample grasps using much fewer actuators than joints. This creates two simultaneous requirements: even with constraints due to underactuation, the hand needs to both (i) reach the desired postures and (ii) apply the needed forces to load the grasps stably. Using the concept of underactuation manifolds, we thus aim to optimize the actuation parameters under real-world constraints, in order to (i) shape the *Mechanically Realizable Posture Manifold* to approach the desired grasping poses in the pre-contact phase, and also (ii) optimize the *Mechanically Realizable Torque Manifold* to provide the joint torques as close to equilibrium as possible in the post-contact

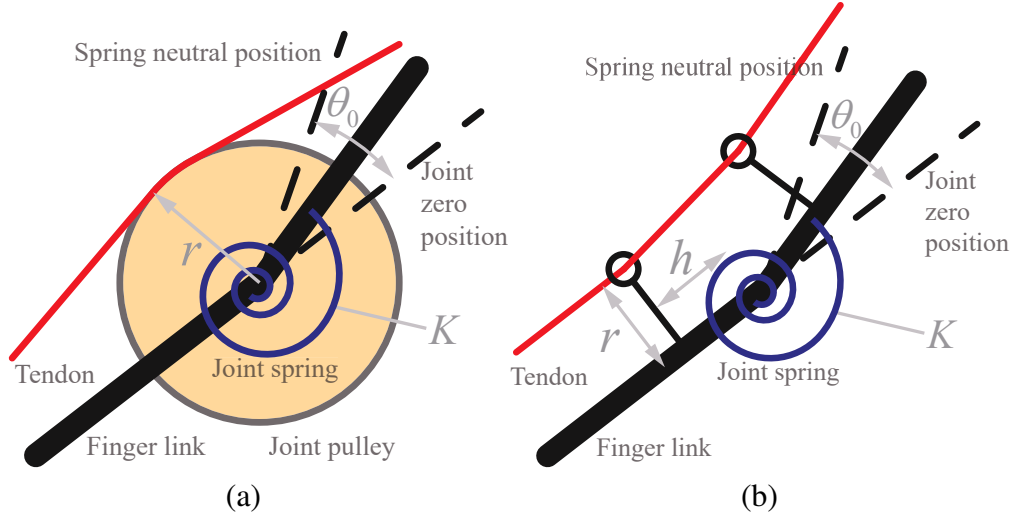


Figure 3.2: Underactuation parameters in a joint. (a) shows the parameterization for the design using pulleys and (b) shows the design with via-points.

phase.

In our case, the actuation parameters we wish to optimize are: the tendon moment arms (or the pulley radii) in the joints, the restoring spring stiffnesses, and the spring preload angles (defined as the spring flexion when joint angles are zeros). Fig. 3.2 shows the parameterization of joint actuation of two different cases where (a) shows the design with pulleys and (b) shows the design with tendon via-points.

Here we tie back to the idea of “dimensionality reduction”. Regular dimensionality reduction methods, such as PCA, look for the low-dimensional structures by optimizing Euclidean-distance-based metrics. In contrast, our method, which is a “mechanical dimensionality reduction”, aims to optimize physics-based metrics, which we will further explain in the next section.

The optimization needs to obey certain constraints. For example, restoring spring stiffnesses are constrained by the physical dimensions of the allowed mounting space, and they are only available in a discrete series of values offered by the manufacturer. The ability to deal with real-world constraints is one of the advantages of our method compared to the direct implementation of synergies in [59, 60, 61, 62, 63, 64].

We contrast our method against two traditional approaches for obtaining low-dimensional manifolds (or synergies), reviewed in the Related Work chapter. *The first type* is to obtain a manifold

by simply fitting a function (e.g. linear fit through Principal Component Analysis (PCA)) to a set of data (e.g. target grasp postures). However, these methods *have no guarantee that the resulting manifold can be implemented in an underactuated mechanism* under design constraints; thus, the corresponding synergies can often be implemented only at a software level. Furthermore, operating exclusively in posture space does not guarantee grasp stability – a force equilibrium problem. Instead, we aim to use physics-based metrics for the dimensionality reduction, which in nature consider force equilibrium. *The second traditional approach* is to implicitly define a manifold by designing a hand empirically, but no method exists that can *make the resulting manifold fit a specific set of sample grasps*. We aim to combine the advantages of both methods: by fitting Mechanically Realizable Manifolds to target grasps in both posture and torque spaces, we ensure that the results are realizable in practice via underactuation and suitable for a set of grasps that the resulting hand can execute in a stable fashion.

3.3 Problem Decomposition

The aforementioned optimization is a multi-objective problem in the same design space: we need to optimize for both pre-contact kinematic behavior and post-contact force generation behavior. While formulating some weighted combination of these objectives is possible, it would require arbitrarily assigned weights, which we would like to avoid. Besides, it is also a high-dimensional optimization if we search all parameters simultaneously. It would thus be beneficial if we could split the optimization in an appropriate way.

Our insight is that we can solve our problem by decomposing it into three parts:

- The optimization of Mechanically Realizable Torque Manifold
- The optimization of Mechanically Realizable Posture Manifold for *inter*-tendon behavior
- The optimization of Mechanically Realizable Posture Manifold for *intra*-tendon behavior

We explain this decomposition as follows.

To begin with, we can split the problem into the optimizations for pre-contact and post-contact behavior, by requiring that the hand is in (or close to) equilibrium in both of these phases so there is no (or negligible) post-contact movement. As we show in Section 3.4, this equilibrium condition becomes one of the goals of our optimization; if the result indicates this goal cannot be achieved for any grasp in our set, we have measures to exclude such grasps and conduct the optimization again.

For a joint (either single DoF or multi-DoF such as a universal joint), equilibrium just before the contacts are made can be expressed as (3.1), which models the general case where multiple tendons are connecting to a joint. \mathbf{r} , \mathbf{t}_{pre} , $\boldsymbol{\theta}$, $\boldsymbol{\tau}_s$ are the vectors of tendon moment arms, pre-contact tensions, joint values and joint spring torques. p is the number of tendons connected to this joint. Once additional torque is applied and the grasp is loaded, equilibrium can be expressed as (3.2), where \mathbf{t}_{post} is the post-contact tendon tension.

Combining (3.1) and (3.2) shows that the net joint torque is only affected by the moment arms but not spring parameters, shown in (3.3). Therefore, we can optimize for the *post-contact* grasp equilibrium first (which results in zero post-contact movement as the contact forces are increased), and then optimize the rest of the parameters for the *pre-contact* kinematic behavior.

$$\sum_{j=1}^p \mathbf{t}_{pre,j} \times \mathbf{r}_j + \boldsymbol{\tau}_s(\boldsymbol{\theta}) = \mathbf{0} \quad (3.1)$$

$$\sum_{j=1}^p \mathbf{t}_{post,j} \times \mathbf{r}_j + \boldsymbol{\tau}_s(\boldsymbol{\theta}) = \boldsymbol{\tau}_{net} \quad (3.2)$$

$$\boldsymbol{\tau}_{net} = \sum_{j=1}^p (\mathbf{t}_{post,j} - \mathbf{t}_{pre,j}) \times \mathbf{r}_j = \sum_{j=1}^p \mathbf{t}_{net,j} \times \mathbf{r}_j \quad (3.3)$$

Furthermore, the optimization for the pre-contact stage has two aspects: the *inter*-tendon kinematic behavior and the *intra*-tendon kinematic behavior. In the case of inter-tendon kinematics, different tendons are driven by the same motor (we do not use differential mechanisms such as

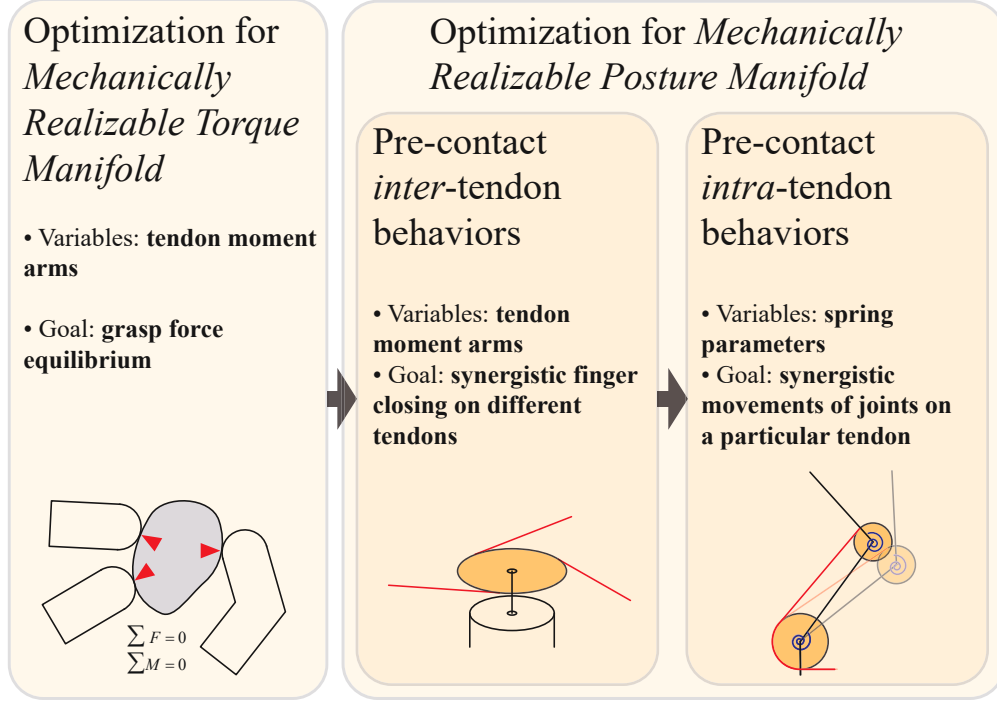


Figure 3.3: Illustration of the optimization decomposition. The hand optimization is divided into three steps as above, and each step aim to optimize different aspects of the hand behaviors.

floating pulleys [9, 111]), so the amount of tendon shortening of each tendon is independent. In contrast, in the case of intra-tendon kinematics, multiple joints are driven by the same tendon, so their behaviors are coupled. Thus, the inter-tendon optimization ensures the coordination between the kinematic chains on different tendons is as desired, and the intra-tendon optimization ensures the correlation of individual joints connected on the same tendon is as desired. We select different groups of design parameters in different optimizations: the inter-tendon optimization determines the tendon moment arms, and the intra-tendon optimization determines the spring parameters.

The decomposition is shown in Fig. 3.3. In the first step, we alter the tendon moment arms to optimize the Mechanically Realizable Torque Manifold to create post-contact equilibrium. As the optimization in this step has an infinite number of minima, which we will explain in the next subsection, we pick one that also solves the second step: the optimization of Mechanically Realizable Posture Manifold for inter-tendon behavior. At this point, we can fully determine the optimal tendon moment arms. After that, we optimize the spring parameters (stiffnesses and preloads) to make sure that the Mechanically Realizable Posture Manifold for each tendon passes closely to the

sample grasps.

3.4 Optimization of Mechanically Realizable Torque Manifold

The objective of this optimization is to have the underactuated hand and object as close to equilibrium as possible in the post-contact phase. The variables we alter are the tendon moment arms in all joints. We will explain at the end of this subsection that there are infinite solutions, so the optimal tendon moment arms will be determined in the next step with an extra objective. We highlight that, since we aim to optimize the grasps to be in equilibrium, we do not consider dynamics in the loop, i.e. all analysis and algorithms are quasi-static.

Grasp Analysis. To form an objective for the optimization, an evaluation of grasp stability is needed. Here we employ a well-established grasp analysis formulation [9]. We use the (linearized) model of point-contact with friction. For contact k , contact wrench \mathbf{c}_k can be expressed as linear combinations of $\boldsymbol{\beta}_k$ — the amplitudes of the frictional and normal components, related by a matrix \mathbf{D}_k , as shown in (3.4). The matrix \mathbf{D}_k and the vector $\boldsymbol{\beta}_k$ are also known as friction pyramid “generator” and “weights” in [112]. Equations (3.5) and (3.6) model the effect that contact force must be constrained inside the friction cone (or pyramid). The details about the construction of matrices \mathbf{D}_k and \mathbf{F}_k can be found in [112].

$$\mathbf{c}_k = \mathbf{D}_k \boldsymbol{\beta}_k \quad (3.4)$$

$$\mathbf{F}_k \boldsymbol{\beta}_k \leq \mathbf{0} \quad (3.5)$$

$$\boldsymbol{\beta}_k \geq \mathbf{0} \quad (3.6)$$

In general, a grasp is stable if the following conditions are satisfied:

- *Hand equilibrium*: the active joint torques are balanced by contact forces.

$$\mathbf{J}^T \mathbf{c} = \mathbf{J}^T \mathbf{D} \boldsymbol{\beta} = \boldsymbol{\tau}_{eq} \quad (3.7)$$

- *Object equilibrium*: the resultant object wrench is zero.

$$\mathbf{G}\mathbf{c} = \mathbf{G}\mathbf{D}\boldsymbol{\beta} = \mathbf{0} \quad (3.8)$$

- *Friction constraints*: the contact forces are constrained inside the friction cone (or pyramid).

$$\mathbf{F}\boldsymbol{\beta} \leq \mathbf{0} \quad (3.9)$$

$$\boldsymbol{\beta} \geq \mathbf{0} \quad (3.10)$$

In these formulations, \mathbf{J} is the contact Jacobian, \mathbf{G} is the Grasp Map matrix, \mathbf{D} and \mathbf{F} are block-diagonal matrices constructed by \mathbf{D}_k and \mathbf{F}_k respectively, \mathbf{c} and $\boldsymbol{\beta}$ are stacked vectors constructed by \mathbf{c}_k and $\boldsymbol{\beta}_k$ respectively, and $\boldsymbol{\tau}_{eq}$ is the desired joint torque vector to create hand equilibrium.

For a force-closure grasp with a given pose (as the sample grasps in this paper), there may be infinite equilibrium torque combinations that satisfy (3.7) – (3.10). To select one, this formulation can be turned into an optimization problem by switching any one of the conditions to an objective. We further discuss the optimization formulation in the next subsection.

Optimization Formulation. The overview of our formulation is as follows: we incorporate a dual-layer optimization framework, in which the inner layer is an optimization to calculate a metric of a specific grasp given a certain set of design parameters, and the outer layer is a search over the parameters for all considered grasps.

In our problem, since the hand is underactuated, the joint torques are not independent. Instead, the actually generated net torque $\boldsymbol{\tau}_{net}^{gen}$ follows the relationship:

$$\boldsymbol{\tau}_{net}^{gen} = \mathbf{A}\mathbf{t}_{net} \quad (3.11)$$

where \mathbf{A} is the Actuation Matrix, which is a function of the tendon moment arms r_1, r_2, \dots, r_m (m is the number of DoFs), and may or may not be configuration-dependent. \mathbf{t}_{net} is the net tendon

tension vector compared to the tension just before touch.

For a set of given tendon moment arms and a given grasp pose, we wish to find the contact force magnitudes $\boldsymbol{\beta}$ and the net tension in each tendon \mathbf{t}_{net} which solve (3.7) – (3.11). We change the search for the exact solution to an optimization problem by turning hand equilibrium (3.7) from a constraint into an objective function. The unbalanced joint torque vector $\Delta\boldsymbol{\tau}_{post}$ is shown in (3.12) (subscript *post* meaning “post-contact”). We use the norm of this vector $\|\Delta\boldsymbol{\tau}_{post}\|$ as the stability metric, and a lower value is considered better.

$$\Delta\boldsymbol{\tau}_{post} = \boldsymbol{\tau}_{eq} - \boldsymbol{\tau}_{net}^{gen} = \mathbf{J}^T \mathbf{D} \boldsymbol{\beta} - \mathbf{A} \mathbf{t}_{net} \quad (3.12)$$

For any grasp, the value of this objective after optimization might be (i) exactly zero, (ii) a small value (below an empirically determined threshold), or (iii) a large value (exceeding the same threshold). For case (i), no post-contact movement will occur. For case (ii), we assume that the small unbalanced torques will be compensated by either friction or a negligible reconfiguration of the hand which will not affect stability. For case (iii), we have measures to exclude such grasps and conduct the optimization again, as we will discuss at the end of this section.

The inner layer problem – to find the minimal norm of unbalanced torques for one given grasp – is a convex Quadratic Program (QP) as follows, shown in (3.13) – (3.18).

find:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{t}_{net} \end{bmatrix} \quad (3.13)$$

minimize:

$$\|\Delta\boldsymbol{\tau}_{post}\|^2 = \|\mathbf{Q}\mathbf{x}\|^2 = \mathbf{x}^T \mathbf{Q}^T \mathbf{Q} \mathbf{x} \quad (3.14)$$

$$\text{where } \mathbf{Q} = \begin{bmatrix} \mathbf{J}^T \mathbf{D} & -\mathbf{A} \end{bmatrix}$$

subject to:

$$\begin{bmatrix} \mathbf{G} \mathbf{D} & \mathbf{O} \end{bmatrix} \mathbf{x} = \mathbf{0} \quad (3.15)$$

$$\begin{bmatrix} \mathbf{F} & \mathbf{O} \end{bmatrix} \mathbf{x} \leq \mathbf{0} \quad (3.16)$$

$$\mathbf{x} \geq \mathbf{0} \quad (3.17)$$

$$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{J}^T \mathbf{D} & \mathbf{O} \end{bmatrix} \mathbf{x} = 1 \quad (3.18)$$

The constraints (3.15) – (3.17) are extended versions of (3.8) – (3.10) and the last one (3.18) prevents the trivial solution where all contact forces and joint torques are zeros, by constraining the sum of joint torques equal to one. In this way, the calculated grasp stability metric $\|\Delta\boldsymbol{\tau}_{post}\|$ is a normalized unitless torque.

Since the aforementioned inner layer can give a stability metric for a specific desired grasp, the outer-layer is a global search over the tendon moment arms for all considered grasps using the inner layer results. The objective function for the global search is an overall metric using the root of squared sum of all grasps' quality metrics, shown in (3.20). The outer layer global optimization is formulated as:

search:

$$r_1, r_2, \dots, r_m \quad (3.19)$$

minimize:

$$f_{trq}(r_1, r_2, \dots, r_m) = \left(\sum_{i=1}^n \|\Delta\boldsymbol{\tau}_{post,i}\|^2 \right)^{\frac{1}{2}} \quad (3.20)$$

subject to:

$$r_i \in [r_{lb}, r_{ub}], \quad i = 1, 2, \dots, m \quad (3.21)$$

save:

$$f_{trq}^{min} = \min(f_{trq}(r_1, r_2, \dots, r_m)) \quad (3.22)$$

where $\|\Delta\boldsymbol{\tau}_{post,i}\|$ is the individual stability metrics computed by the QP in (3.13) – (3.18) for each

simulated desired grasp, n is the number of grasps, r_{lb} and r_{ub} are the lower and upper bounds of tendon moment arms.

Although the inner-layer is convex, the outer-layer is not, and is not trivial to be reformulated as a convex problem. Therefore we decided to use a stochastic global search. The optimizer we chose is the Covariance Matrix Adaptation – Evolutionary Strategy (CMA-ES) [49] [113]. It is a stochastic, derivative-free algorithm for black-box global optimization, in which the covariance matrix of the distribution of the candidate solutions is updated adaptively in each generation. This method learns a stochastic second-order approximation of the objective, and drives the candidate solutions to the optimum, even when the function is ill-conditioned.

We aim to find a Mechanically Realizable Manifold that minimizes the unbalanced torque $\|\Delta\tau_{post,i}\|$ for all grasps. In practice, we exclude from this optimization the difficult-to-achieve grasps where the unbalanced torque is found to be larger than an empirically determined threshold, and then we iterate again and solve the optimization problem for the rest of the grasps. In this way, we can design the hand to perform as well as possible on the grasps possible to create, instead of attempting to also satisfy equilibrium for impossible grasps. Moreover, the number of grasps excluded in this way is an important metric of the hand’s overall capabilities to achieve our design goals. All our results, presented later in the paper, will thus report both the number of excluded grasps, and the values of all optimization objectives (computed over the grasps that have been kept).

Here we explain why the minimum is not unique. Let us assume we have found a set of optimal tendon moment arms, with a certain \mathbf{x} (thus a certain set of \mathbf{t}_{net}) in (3.13). As long as the QP (3.13) – (3.18) can find an \mathbf{x} (and thus \mathbf{t}_{net}) that can keep $\mathbf{A}\mathbf{t}_{net}$ the same when we alter the tendon moment arms (one possible way is to scale \mathbf{x} (or \mathbf{t}) while scaling entries in \mathbf{A}), the metric of the inner layer (the QP (3.13) – (3.18)) remains minimal, then the objective function value of the outer layer (the global search (3.20) (3.21)) remains minimal. Therefore there are other optimal solutions for the tendon moment arms.

The optimization of Mechanically Realizable Torque Manifold is summarized as Fig. 3.4.

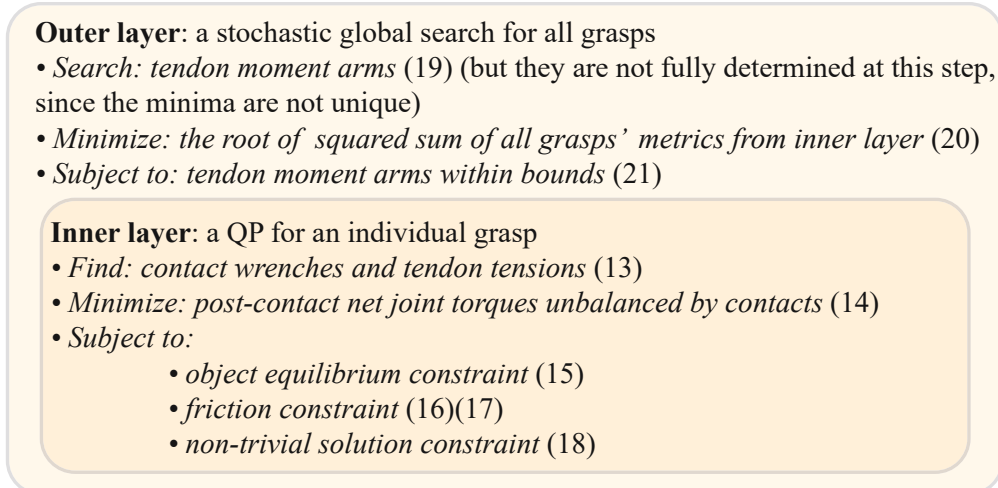


Figure 3.4: Summary of the optimization for Mechanically Realizable Torque Manifolds. The algorithm consists of two layers: the outer layer optimization search over tendon moment arms while the inner layer optimization gives a stability metric for the outer layer.

After this step, we can find a Mechanically Realizable Torque Manifold, on which the joint torque distribution, determined by the tendon moment arms, is as close to equilibrium as possible for all the considered grasps. We will pick the one set of values for tendon moment arms from the non-unique and equally good solutions with an extra objective in the next subsection.

3.5 Optimization of Mechanically Realizable Posture Manifold for Inter-tendon Behavior

In this step, we aim to get a coordinated finger movement among different tendons. As the hand moves from a reference starting pose into a grasp pose, each tendon must travel a specific amount, dictated by the movement of each joint as well as the tendon's moment arm around each joint. (This assumes that tendons are inextensible, and also are not allowed to become slack as they lose force transmission abilities.) However, tendon travel must also be in accordance with the travel of the motor that the tendon is connected to, a problem that becomes non-trivial for the case of multiple tendons connected to the same motor.

We again translate this requirement into an optimization problem. For each of the desired grasp poses, we minimize the mismatch between the actual tendon travel collected by the motor and the tendon travel required by the sample grasps, for all tendons connected to the same motor, with

motor travel angle(s) as a free variable.

This optimization is only related to the tendon moment arms, which are picked from the set of equally good solutions in the previous optimization, and can be fully determined with the extra objective in this step.

We note that the pool of sample grasps is a superset of the one in the previous subsection: in addition to sample grasps, we also include the fully open configuration, in which all joints are opened to their mechanical limits. For each grasping pose in the original grasp pool, we add an opening pose, such that opening and closing poses are always in pairs. In this way, we can ensure the hand can actually open instead of moving between desired grasping poses.

We also follow the dual-layer optimization framework as in the previous subsection. The inner layer minimizes the norm of an error vector whose entries are the difference between the tendon travel the motor collects and the one the grasp requires. The outer layer is also a stochastic global optimization to minimize the overall metric for all grasps, with a constraint that the objective function in the previous optimization needs to reach its minimum.

In the inner layer, the error vector can be expressed as (3.23), where the matrix \mathbf{M} is a motor-tendon connection matrix whose entries can be either motor pulley radius r_1, r_2, \dots, r_m (meaning the corresponding tendon is connected to the corresponding motor) or zero (meaning not connected), $\boldsymbol{\theta}_{mot}$ is a vector of angles that the motors moved, and \mathbf{s} is the tendon travel vector whose entries are the travel of the corresponding tendon required by the joint values in the desired grasp (compared to zero positions).

$$\mathbf{e} = \mathbf{M}\boldsymbol{\theta}_{mot} - \mathbf{s} \quad (3.23)$$

Minimizing the norm of the error vector $\|\mathbf{e}\|$ is a QP over the angles the motors moved $\boldsymbol{\theta}_{mot}$, with no constraints. It is shown in the QP (3.24) (3.25) below.

find:

$$\boldsymbol{\theta}_{mot} \quad (3.24)$$

minimize:

$$\|\mathbf{e}\|^2 = \boldsymbol{\theta}_{mot}^T \mathbf{M}^T \mathbf{M} \boldsymbol{\theta}_{mot} - 2\mathbf{s}^T \mathbf{M} \boldsymbol{\theta}_{mot} + \mathbf{s}^T \mathbf{s} \quad (3.25)$$

The outer layer, which is a global optimization over the tendon moment arms to minimize the overall metric $f_{inter}(r_1, r_2, \dots, r_m)$ (the root of the squared sum of inner layer results), has to obey the constraint that the objective of the previous optimization (of Mechanically Realizable Torque Manifold) needs to take its minimum value, shown in (3.28). With this constraint, the tendon moment arms can only take values that minimize the objective in the previous optimization. Then we can find a solution to both the previous and the current optimization. The constraint is handled in a soft way by giving a high penalty if the constraint is violated, and more penalty if the candidate solutions are farther away from the feasible region. The outer layer optimization is as follows:

find:

$$r_1, r_2, \dots, r_m \quad (3.26)$$

minimize:

$$f_{inter}(r_1, r_2, \dots, r_m) = \left(\sum_{i=1}^n \|\mathbf{e}_i\|^2 \right)^{\frac{1}{2}} \quad (3.27)$$

subject to:

$$f_{trq}(r_1, r_2, \dots, r_m) = f_{trq}^{min} \quad (3.28)$$

$$r_i \in [r_{lb}, r_{ub}], \quad i = 1, 2, \dots, m \quad (3.29)$$

where $\|\mathbf{e}_i\|$ is the individual inner layer metric for each grasp calculated from the QP (3.24) (3.25), f_{trq}^{min} is the minimal function value saved from the previous step, and r_{lb} and r_{ub} are the lower and upper bounds of the pulley radii r . We also incorporate CMA-ES for the global search.

At this point, the tendon moment arms r_1, r_2, \dots, r_m are fully determined. The optimization of Mechanically Realizable Posture Manifold for inter-tendon kinematic behavior can be summarized as Fig. 3.5.

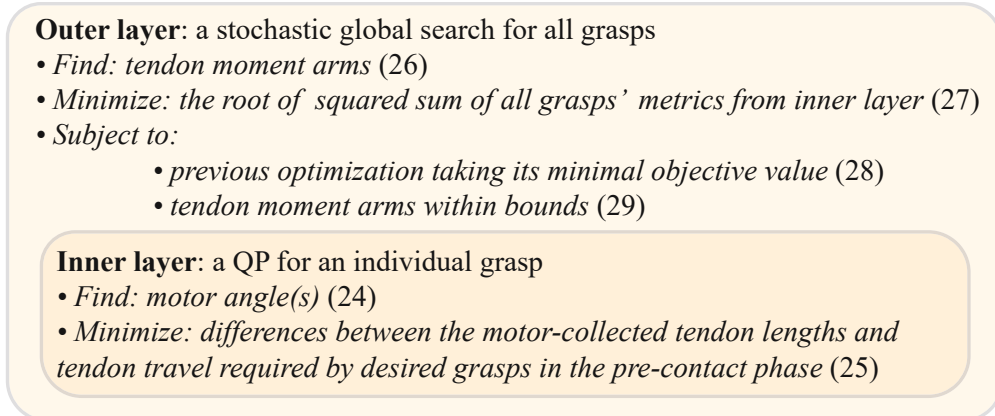


Figure 3.5: Summary of the optimization for Mechanically Realizable Posture Manifold for inter-tendon kinematic behavior. The algorithm consists of two layers: the outer layer optimization search over tendon moment arms while the inner layer optimization gives a tendon travel error metric for the outer layer.

3.6 Optimization of Mechanically Realizable Posture Manifold for Intra-tendon Behavior

The goal of this optimization is to coordinate the movements of different joints along one certain tendon to have the Mechanically Realizable Posture Manifold close to the desired grasp poses. We import the tendon moment arms from the previous optimization, and the remaining parameters to optimize are: the spring stiffnesses and the spring preloads.

We translate the goal of quasi-statically reaching sample grasps to the one of minimizing the pre-contact unbalanced spring torques if the hand is posed in the desired grasp configurations. Lower unbalanced torque means the Mechanically Realizable Posture Manifold is closer to the desired grasp. We emphasize that in this part we only consider the equilibrium of the hand itself, without the object.

The unbalanced spring torque vector can be calculated as (3.30), where the matrix \mathbf{A} is the aforementioned Actuation Matrix, and $\boldsymbol{\tau}_{spr}$ is a vector of spring torques calculated by given spring parameters and given poses (shown in (3.31)). We note that here the \mathbf{t} is the absolute tendon tension, which is different from the \mathbf{t}_{net}

$$\Delta\boldsymbol{\tau}_{pre} = \mathbf{A}\mathbf{t} - \boldsymbol{\tau}_{spr} \quad (3.30)$$

$$\boldsymbol{\tau}_{spr} = [K_1(\theta_1 + \theta_{01}), \dots, K_m(\theta_m + \theta_{0m})]^T \quad (3.31)$$

We wish to find the \mathbf{t} vector resulting in a minimum norm of unbalanced joint torques, which is also a convex QP as shown below.

find:

$$\mathbf{t} \quad (3.32)$$

minimize:

$$\|\Delta\boldsymbol{\tau}_{pre}\|^2 = \mathbf{t}^T \mathbf{A}^T \mathbf{A} \mathbf{t} - 2\boldsymbol{\tau}_{spr}^T \mathbf{A} \mathbf{t} + \boldsymbol{\tau}_{spr}^T \boldsymbol{\tau}_{spr} \quad (3.33)$$

subject to:

$$\mathbf{t} \geq \mathbf{0} \quad (3.34)$$

We still incorporate the dual-layer optimization. In the outer layer, we use CMA-ES to find the global optimum of the overall metric $f_{intra}(K_1, \dots, K_m, \theta_{01}, \dots, \theta_{0m})$, as shown in (3.35) – (3.37). In practice, since the intra-tendon kinematic behavior is only related to the joint along that tendon, we can perform optimization only for those joints separately at a time, in order to reduce the search dimensions. It is also needed to note that the spring stiffnesses can only take discrete values offered by the manufacturer, so we incorporate the integer handling in CMA-ES.

find:

$$K_1, K_2, \dots, K_m, \theta_{01}, \theta_{02}, \dots, \theta_{0m} \quad (3.35)$$

minimize:

$$f_{intra}(K_1, \dots, K_m, \theta_{01}, \dots, \theta_{0m}) = \left(\sum_{i=1}^n \|\Delta\boldsymbol{\tau}_{pre,i}\|^2 \right)^{\frac{1}{2}} \quad (3.36)$$

subject to:

$$K_i \in \mathbb{K}, i = 1, 2, \dots, m \quad (3.37)$$

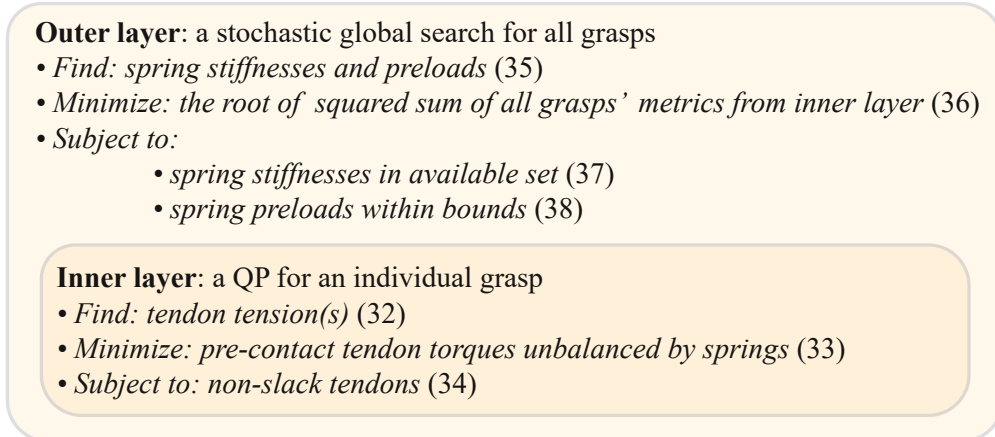


Figure 3.6: Summary of the optimization for Mechanically Realizable Posture Manifold for intra-tendon kinematic behavior. The algorithm consists of two layers: the outer layer optimization search over spring parameters while the inner layer optimization gives a metric for the outer layer.

$$\theta_{0i} \in [\theta_{0lb}, \theta_{0ub}], i = 1, 2, \dots, m \quad (3.38)$$

Here, the $\|\Delta\tau_{pre,i}\|$ is the individual metric for each grasp from the inner layer optimization (3.32) – (3.34), \mathbb{K} is the set of discrete spring stiffnesses provided by the manufacturer, and θ_{0lb} and θ_{0ub} are the lower and upper bounds of the spring preload angle θ_0 .

The optimization of Mechanically Realizable Posture Manifold for intra-tendon kinematic behavior can be summarized as Fig. 3.6.

3.7 Complete Design Method Recap

Fig. 3.7 is the recap of our method. The design process starts from prespecified sample grasps and hand kinematics, then goes through the aforementioned three steps for different aspects of hand behavior: the optimization of Mechanically Realizable Torque Manifold, as well as the Mechanically Realizable Posture Manifold for inter- and intra-tendon behavior. Finally, it results in a set of optimal actuation parameters for both posture shaping and force generation.

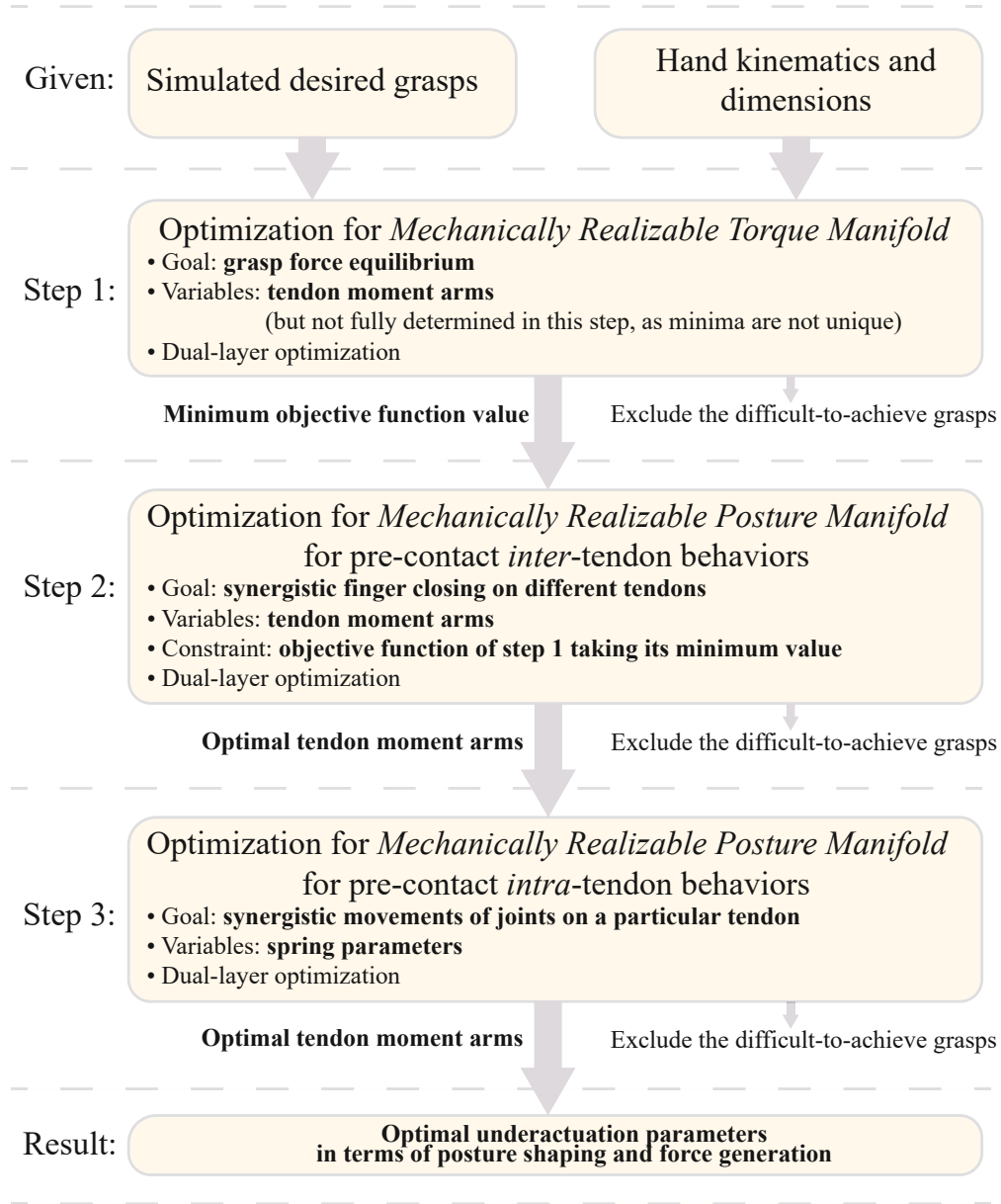


Figure 3.7: Summary of the proposed three-step optimization of Mechanically Realizable Manifolds.

Chapter 4: Design Cases and Quantitative Comparisons

In this chapter, we applied the aforementioned “Mechanically Realizable Manifolds” technique in three concrete design cases. Then we present the idea to use the numeric results of the optimizations as “hand quality metrics” to compare different hand kinematic designs. Finally, we show the physical prototypes and real-world evaluations of the best two well-performing designs.

The intended application of these hands is for the *Astrobee* robot [114] (Fig. 4.1) in the International Space Station (ISS). The *Astrobee* is a cube-shaped free-flying assistive robot, designed to help the astronauts with in-cabin monitoring and many other tasks. We aim to enable it to do object retrieval by mounting a simple arm and a versatile hand to its payload bay.

Our method is suitable for this design task for several reasons. First of all, a highly synergistic underactuated but versatile hand is needed for this application because of the limited onboard space and control signals. Second, the objects in the ISS are known and relatively unchanged, meaning that we have a given set of objects and there is a need for *data-driven design* which is exactly what our method requires. Third, the available room to store the hand inside the *Astrobee* robot is given, so the dimensions of the hand can be specified beforehand as required by our method.

4.1 Design Case I: Single-Motor Hand with Roll-Pitch Fingers

The first design case is a three-finger single-motor hand with a roll-pitch finger configuration (we define the finger flexion as “pitch” and the rotation around axes perpendicular to the palm as “roll”). The hand has altogether eight joints: two joints (proximal and distal joints) on the thumb, and three joints (the finger roll joint, proximal joint and distal joint) on the opposing two fingers. The hand models including kinematic configuration and tendon connectivity pattern are shown in Fig. 4.2.



Figure 4.1: The *Astrobee* free-flying robot in ISS.

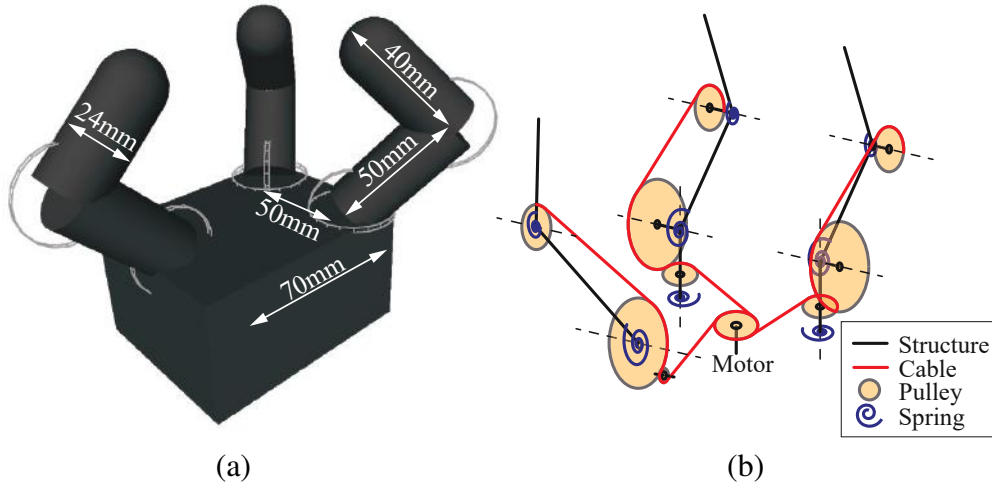


Figure 4.2: (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case I.

Grasp Collection

The hand models are created in *GraspIt!* Simulator [115] without considering underactuation, and 21 sample grasps are created for 15 commonly-used ISS objects selected according to a case study with a domain expert, mainly including food (such as cans) and tools (such as screwdrivers). All grasps have force-closure property, checked with the Ferrari-Canny ϵ -metric [116] by $\epsilon > \epsilon_0$, where ϵ_0 is a positive threshold to have some safety margin. The grasp types are determined empirically, and if multiple types of grasps as possible for a certain object they are all included. All of the sample grasps for Design Case I are shown in Fig. 4.3. The mechanical limits are $-\pi/4$

to $\pi/4$ radians for roll joints and proximal joints, and 0 to $\pi/2$ radians for distal joints, defining the zero-position as the pose in which all links are perpendicular to the palm and all proximal and distal joint axes are parallel.

The “raw” grasp data contains the joint angles, the contact locations and the friction coefficients. Then we can calculate the grasp Jacobian \mathbf{J} , the grasp map \mathbf{G} , the friction constraint matrix \mathbf{F} and the friction pyramid generator \mathbf{D} in (3.7) – (3.10), and pass them to the following optimization steps.

Optimization of Mechanically Realizable Torque Manifold

In this step, we optimize the joint pulley radii (tendon moment arms) r_{tp} , r_{td} , r_{fr} , r_{fp} , r_{fd} , where the subscripts t and f represent thumb and finger, and r , p , and d represent the roll, proximal and distal joints we consider the two fingers to be mirrored versions of each other). These parameters are illustrated in Fig. 3.2 (a).

The actuation scheme we designed is also shown in Fig. 4.2, where each finger is actuated by one tendon, and all tendons are rigidly connected to the actuator. We note that the finger tendons wrap around the roll joint pulleys, after which the plane of routing rotates 90 degrees and the tendons travel to the proximal and distal joints in the fingers.

The vector of generated net joint torque in (3.11) $\mathbf{t}_{net}^{gen} \in \mathbb{R}^8$, the vector $\mathbf{t}_{net} \in \mathbb{R}^3$ (each element represents the tension on one tendon), and the Actuation Matrix has the specific form of

$$\mathbf{A} = \begin{bmatrix} r_{tp} & & & & & & & \\ r_{td} & & & & & & & \\ & -r_{fr} & & & & & & \\ & r_{fp} & & & & & & \\ & r_{fd} & & & & & & \\ & & r_{fr} & & & & & \\ & & r_{fp} & & & & & \\ & & r_{fd} & & & & & \end{bmatrix} \quad (4.1)$$

In global optimization (3.19) - (3.22), the range of pulley radii is set to 2 – 12 mm, such that the pulley is large enough to be manufacturable but small enough to be mounted into the joints. Using a threshold of 0.1 (unitless normalized torque) for unbalanced torque, one outlier grasp is excluded in this step. The convergence tolerance is set to 10^{-10} for inner layer QP and 10^{-6} for

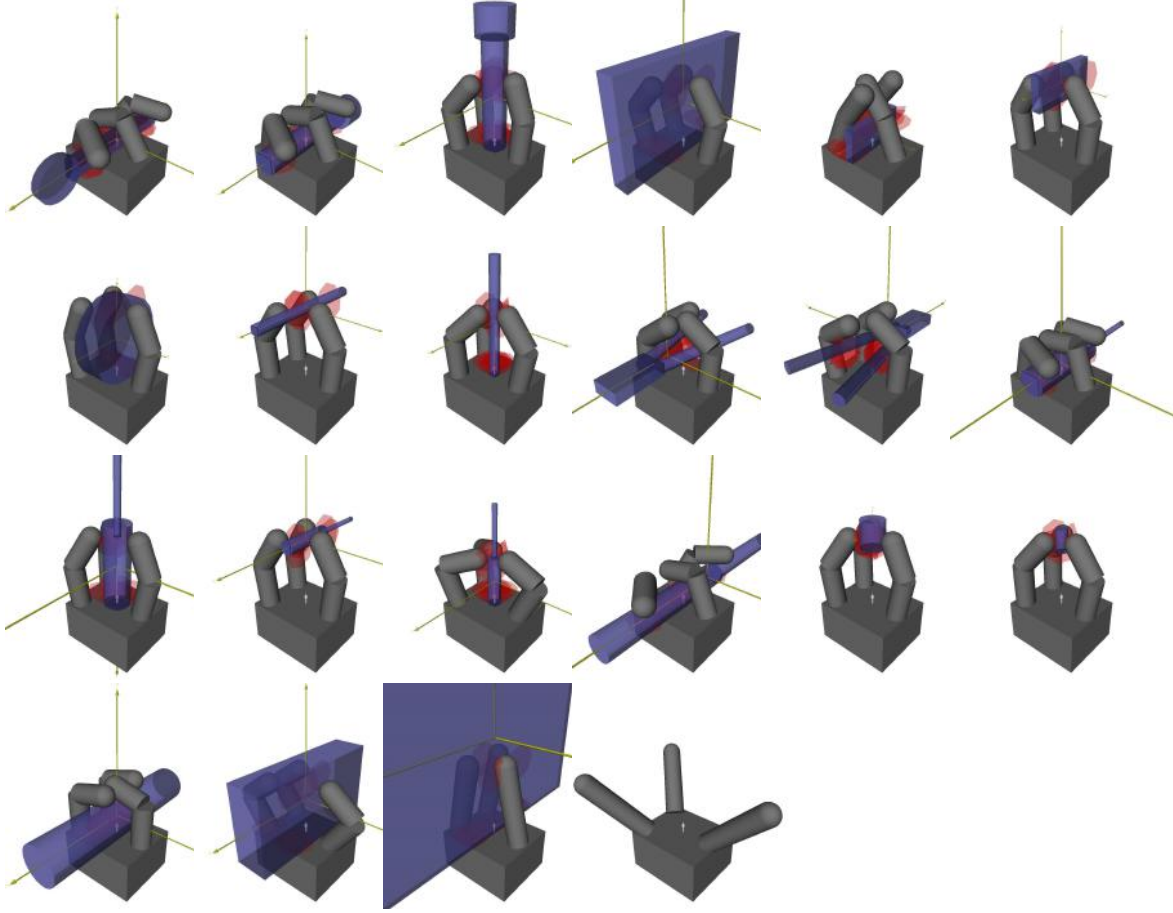


Figure 4.3: sample grasps (the first 21 images) and the opening configuration (the last one) created in *GraspIt!* simulator for Design Case I. The hands are shown in grey, the objects are shown in blue, and contacts (friction cones) are shown in red. All grasps do not consider underactuation and have force-closure property.

outer layer CMA-ES. The above conditions are set the same for this design case and Design Case II and III. We note again that in this step, there are non-unique solutions, so the pulley radii are not fully determined yet. The minimum objective function value is recorded for the next step.

The computation time on a commodity desktop computer (quad-core 3.4 GHz CPU) is 10 minutes, using the *cvxopt* (for QP) and *pycma* (for CMA-ES) packages.

Optimization of the Mechanically Realizable Posture Manifold for Inter-tendon Kinematic behavior

As discussed in the subsection 3.5, we need to pick a unique solution of pulley radii (r_{tp} , r_{td} , r_{fr} , r_{fp} , r_{fd}) from the non-unique solutions from the previous step. The goal is to minimize the mismatch of actual and required tendon travel for different tendons.

The motor-tendon connection matrix \mathbf{M} in (3.23) has a specific form of (4.2), and r_{mot} is the motor pulley radius.

$$\mathbf{M} = [r_{mot}, r_{mot}, r_{mot}]^T \quad (4.2)$$

Besides, the tendon travel vector \mathbf{s} has a specific form of (4.3), where $\boldsymbol{\theta}$ is the vector of joint angles in a desired grasp configuration.

$$\mathbf{s} = \mathbf{A}^T \boldsymbol{\theta} \quad (4.3)$$

In the outer layer, the candidate solutions need to satisfy the constraint that the previous optimization takes its minimal value with a tolerance of 10^{-3} , and the CMA-ES algorithm finds the optimal set of pulley radii with function value convergence tolerance of 10^{-3} . No outliers are found in this step using a threshold length error of 2 mm.

The resulting optimal pulley radii are shown in table 4.1. The computation time is 30 minutes.

Table 4.1: Optimized parameters of Design Case I (in mm, Nmm/rad, rad, respectively)

| Parameter | r_{tp} | r_{td} | r_{fr} | r_{fp} | r_{fd} |
|-----------|----------------|----------------|----------------|----------------|----------------|
| Value | 12.0 | 4.6 | 2.0 | 11.8 | 4.5 |
| Parameter | K_{tp} | K_{td} | K_{fr} | K_{fp} | K_{fd} |
| Value | 5.94 | 2.25 | 3.60 | 19.25 | 7.56 |
| Parameter | θ_{0tp} | θ_{0td} | θ_{0fr} | θ_{0fp} | θ_{0fd} |
| Value | 4.71 | 3.93 | 4.34 | 4.71 | 3.78 |

Optimization of the Mechanically Realizable Posture Manifold for Intra-tendon Kinematic behavior

In this step, we optimize spring stiffnesses K_{tp} , K_{td} , K_{fr} , K_{fp} , K_{fd} , and spring preload angles θ_{0tp} , θ_{0td} , θ_{0fr} , θ_{0fp} , θ_{0fd} , where the subscripts have the same meaning as previous. These parameters are also illustrated in Fig. 3.2.

In practice, since the pre-contact kinematic behavior of each finger is independent of every other one, we can search for each finger separately, and thus reduce the search dimensionality.

The spring stiffnesses are limited by the physical dimensions allowed in the mounting area in the joints. Also, they can only take discrete numbers offered by the manufacturer. In this design, the available stiffnesses are 10 discrete values from 2.25 to 19.25 Nmm/rad.

In addition, the spring preload angles are limited between the maximum allowed torsional angles by the datasheet and the minimum torsion angles to provide enough restoring torques over the entire range of motion. In this design case, the preload angles range from $\pi/4$ to $7\pi/4$ radians for the roll joints, $\pi/4$ to $3\pi/2$ radians for the proximal joints, and 0 to $3\pi/2$ radians for the distal joints.

In this step, there are three outlier grasps excluded using a threshold unbalanced spring torque of 2 Nmm for the thumb and 5 Nmm for fingers.

The optimal spring parameters are shown in table 4.1. The computation time is 5 minutes.

Table 4.2: Optimized parameters of Design case II (in mm, Nmm/rad, rad, respectively)

| Parameter | r_{tp} | r_{td} | r_{fr} | r_{fp} | r_{fd} |
|-----------|----------|----------|-----------|----------|----------|
| Value | 12.0 | 4.5 | arbitrary | 12.0 | 4.5 |

| Parameter | K_{tp} | K_{td} | K_{fp} | K_{fd} |
|-----------|----------|----------|----------|----------|
| Value | 5.94 | 2.25 | 5.94 | 2.25 |

| Parameter | θ_{0tp} | θ_{0td} | θ_{0fp} | θ_{0fd} |
|-----------|----------------|----------------|----------------|----------------|
| Value | 4.71 | 3.86 | 4.71 | 3.82 |

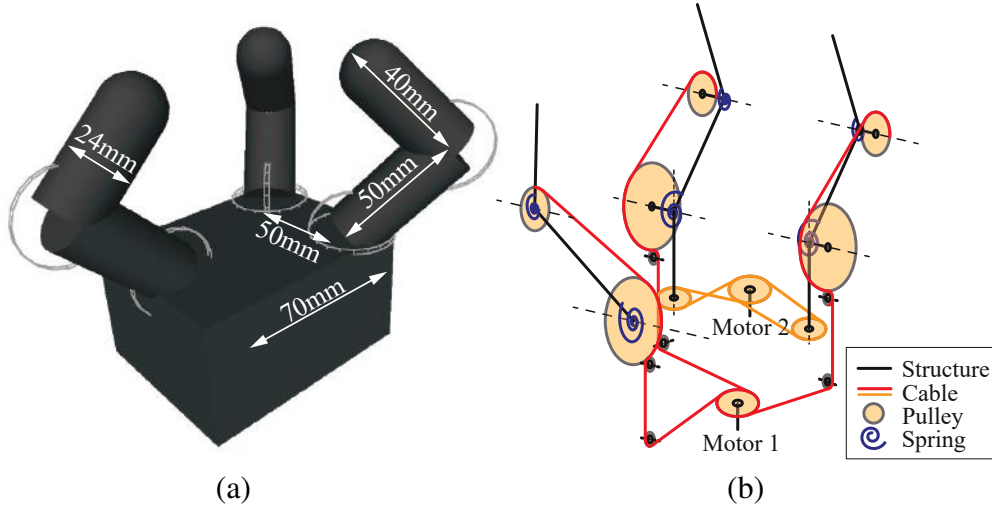


Figure 4.4: (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case II.

4.2 Design Case II: Dual-Motor Hand with Roll-Pitch Fingers

In the previous design, all eight joints are actuated by a single motor, but it is interesting to see the benefits of having an additional motor controlling part of the hand motion separately. Therefore, we propose a variation of the previous design: a dual-motor hand with roll-pitch finger configuration, where one motor is in charge of the flexion of all fingers, and the other motor is in charge of the finger rolling. The hand kinematic configuration is shown in Fig. 4.4.

Here, the tendon tension vector in (3.11) $\mathbf{t}_{net} \in \mathbb{R}^5$ (the first three elements are the tensions on three tendons going to the thumb and fingers, the last two are the forces on the roll transmission connected to the second motor). The Actuation Matrix \mathbf{A} in (3.11) has the specific form of

$$\mathbf{A} = \begin{bmatrix} r_{tp} & & & & \\ r_{td} & & & & \\ & r_{fp} & & -r_{fr} & \\ & r_{fd} & & & \\ & & r_{fp} & & r_{fr} \\ & & r_{fd} & & \end{bmatrix} \quad (4.4)$$

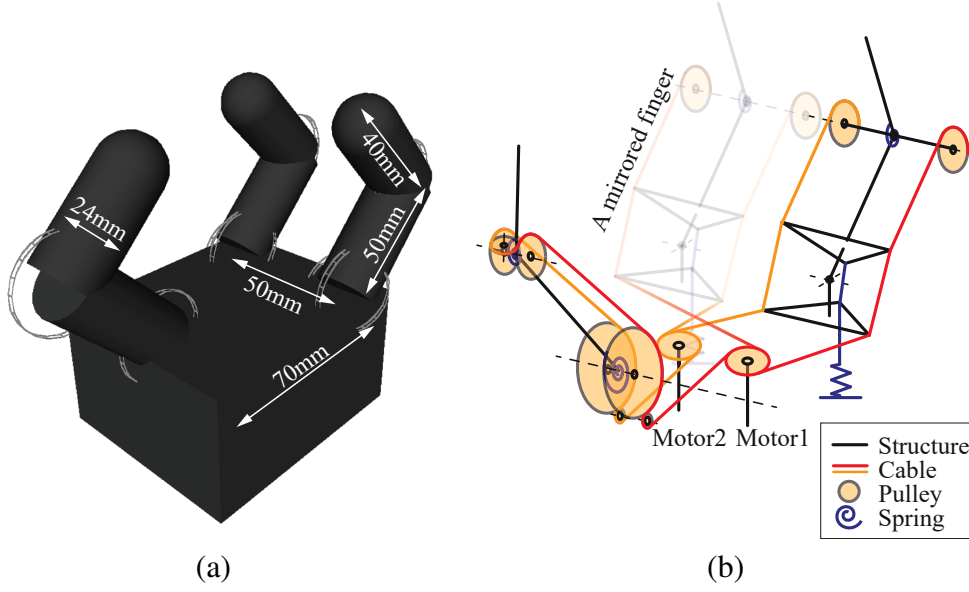


Figure 4.5: (a) Hand model with predefined kinematics and (b) actuation scheme of Design Case III.

The motor-tendon connection matrix M in (3.23) has a specific form of

$$M = \begin{bmatrix} r_{mot1} \\ r_{mot1} \\ r_{mot1} \\ r_{mot2} \\ r_{mot2} \end{bmatrix} \quad (4.5)$$

and the tendon travel vector s in (3.23) is also $A^T\theta$, where θ is the vector of joint angles in a desired grasp configuration.

All design details are the same as in Design Case I, with the exception that the roll joints do not have springs, and thus not having associated spring parameters. There are altogether four grasps excluded using the same criteria as in Design Case I. The optimal pulley radii and spring parameters are shown in table 4.2.

4.3 Design Case III: Dual-Motor Hand with Pitch-Yaw Fingers

The third design case is an underactuated hand with pitch-yaw fingers. The pitch-yaw two-DoF proximal joint is realized by a universal joint with a three-tendon parallel mechanism. The back tendon of the joint is connected to a spring with preload. The front two tendons are actively

controlled. Besides, they are not terminated in the proximal joint, but connected to distal joint pulleys.

The two actively controlled tendons within a finger are connected to different motors. Meanwhile, the symmetric tendons on different fingers share a common motor. In the thumb, two tendons from the two motors are routed passing the proximal and distal joints and connected via an idler inside the fingertip. Fig. 4.5 illustrates the tendon routing scheme.

In this design case, the Actuation Matrix has the form of (4.6), where ρ 's are the configuration-dependent moment arms of the tendon forces, which can be calculated via geometric relationships. The derivation is shown in the Appendix A. The subscripts p and y mean pitch and yaw, the numbers in the subscripts are combinations of finger number and motor number.

$$\mathbf{A} = \begin{bmatrix} 2r_{tp} & & & & \\ 2r_{td} & & & & \\ & \rho_{fpp11} & \rho_{fpp12} & & \\ & \rho_{fpy11} & \rho_{fpy12} & & \\ & r_{fd} & r_{rd} & & \\ & & & \rho_{fpp21} & \rho_{fpp22} \\ & & & \rho_{fpy21} & \rho_{fpy22} \\ & & & r_{fd} & r_{rd} \end{bmatrix} \quad (4.6)$$

The motor-tendon connection matrix \mathbf{M} in (3.23) has a specific form of

$$\mathbf{M} = \begin{bmatrix} r_{mot} & r_{mot} \\ r_{mot} & 0 \\ 0 & r_{mot} \\ r_{mot} & 0 \\ 0 & r_{mot} \end{bmatrix} \quad (4.7)$$

Table 4.3: Optimized parameters of Design case III (in mm, Nmm/rad, rad or mm, respectively)

| Parameter | r_{tp} | r_{td} | h_{fp} | r_{fp} | r_{fd} |
|-----------|----------|----------|----------|----------|----------|
| Value | 4.65 | 2.00 | 6.29 | 12.00 | 2.00 |

| Parameter | K_{tp} | K_{td} | K_{fp} | K_{fd} |
|-----------|----------|----------|----------|----------|
| Value | 5.94 | 2.25 | 0.18 | 19.25 |

| Parameter | θ_{0tp} | θ_{0td} | l_{0fp} | θ_{0fd} |
|-----------|----------------|----------------|-----------|----------------|
| Value | 4.71 | 4.45 | 16.67 | 0.23 |

Table 4.4: Comparison of optimization metrics (in unitless torque, mm, Nmm for column 1, 2 and 3)

| Optimization of: | Mechanically Realizable Torque Manifold | Mechanically Realizable Posture Manifold (inter-tendon) | Mechanically Realizable Posture Manifold (intra-tendon) |
|------------------|--|---|---|
| Design Case I | 0.13 (1 grasp excluded) | 5.82 (0 grasps excluded) | 8.22 (3 grasps excluded) |
| Design Case II | 0.08 (1 grasp excluded) | 3.05 (0 grasps excluded) | 2.82 (3 grasps excluded) |
| Design Case III | 0.33 (0 grasps excluded) | 6.02 (0 grasps excluded) | 14.06 (14 grasps excluded) |

And the tendon travel vector s in (3.23) is

$$s = \begin{bmatrix} 2(\theta_{tp}r_{tp} + \theta_{td}r_{td}) \\ \theta_f d r_{fd} + \Delta l_{fp11} \\ \theta_f d r_{fd} + \Delta l_{fp12} \\ \theta_f d r_{fd} + \Delta l_{fp21} \\ \theta_f d r_{fd} + \Delta l_{fp22} \end{bmatrix} \quad (4.8)$$

where the Δl 's are the tendon length changes between zero-configuration and grasp configuration.

We present the details of the derivation of the above matrices in the Appendix A.

There are 14 grasps excluded using the same criteria. The optimal design parameters are shown in Table 4.3.

4.4 Numerical Evaluation

All the metrics provided by our optimization framework for all three design cases, and the number of excluded grasps in each design step within each case, are summarized in Table 4.4.

In addition to the numerical metrics, it can also be informative to visualize the Mechanically Realizable Torque Manifold and Posture Manifold. However, these manifolds are high-dimensional and cannot be visualized directly. Therefore, we plot the slices of the spaces: The thumb manifolds are shown in two-dimensional joint space and the finger manifolds are shown in three-dimensional joint space separately. In each plot, the dimensionality of the manifold equals the number of motors connected to that thumb or finger.

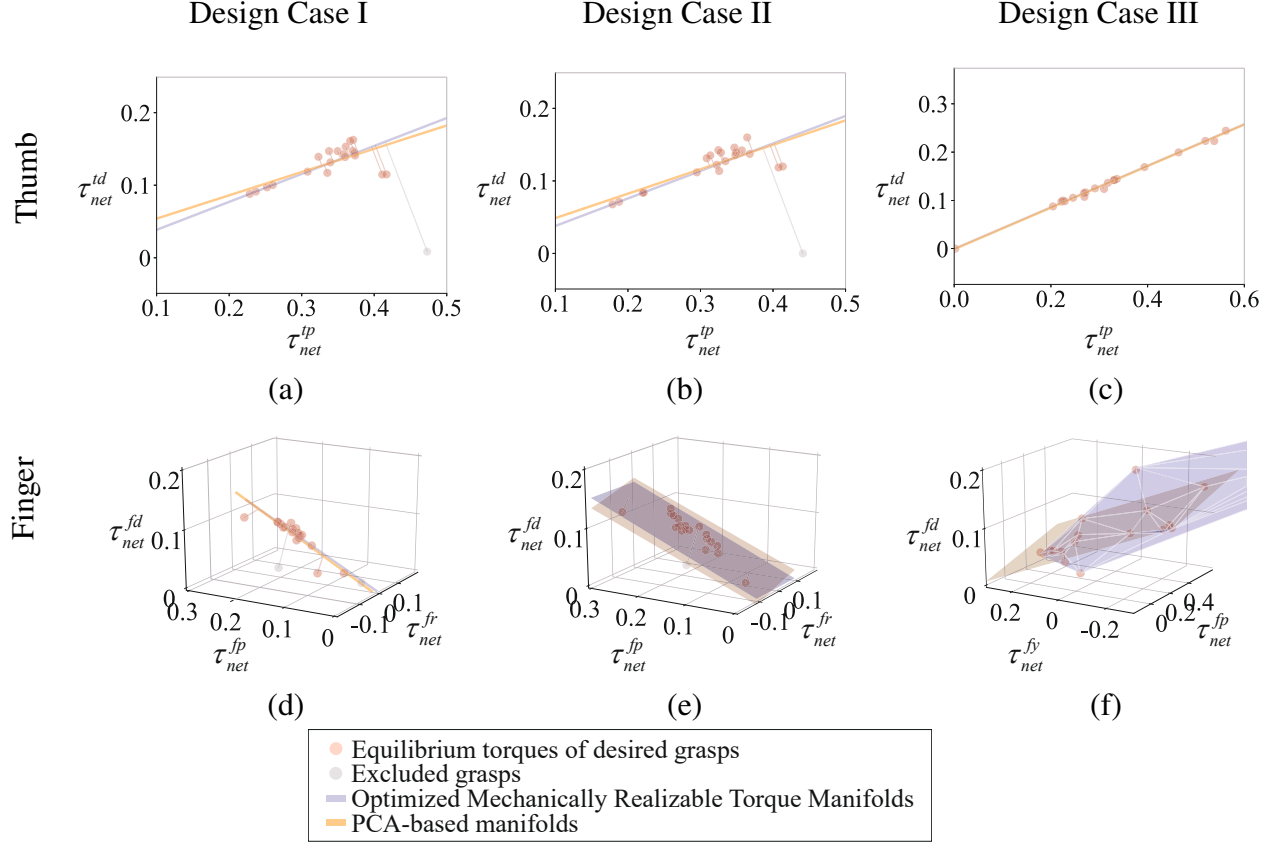


Figure 4.6: The visualization of the Mechanically Realizable Torque Manifolds. The first row shows 2D plots for the two joints in one thumb, and the second row shows 3D plots for the three joints in one finger. Different columns are different design cases. All torques are normalized unitless torques.

The Mechanically Realizable Torque Manifolds for all three design cases are shown as Fig. 4.6. Each column shows one design case, and each row shows the plots for a thumb or a finger. The red dots represent the considered grasps, while the gray ones represent the excluded ones. The blue lines or planes are the Mechanically Realizable Torque Manifolds, and the orange lines or planes are the least-square fittings of red dots based on PCA for comparison. We note that, for each fully-articulated desired grasp, there are infinitely many solutions for equilibrium torques; among these desired points in torque space, we chose to display on the plots (as a red or gray dot) the torques *closest* to the Mechanically Realizable Torque Manifold.

Similarly, the Mechanically Realizable Posture Manifolds for all the design cases are shown in Fig. 4.7. The red dots represent the considered grasps, while the gray ones represent the excluded

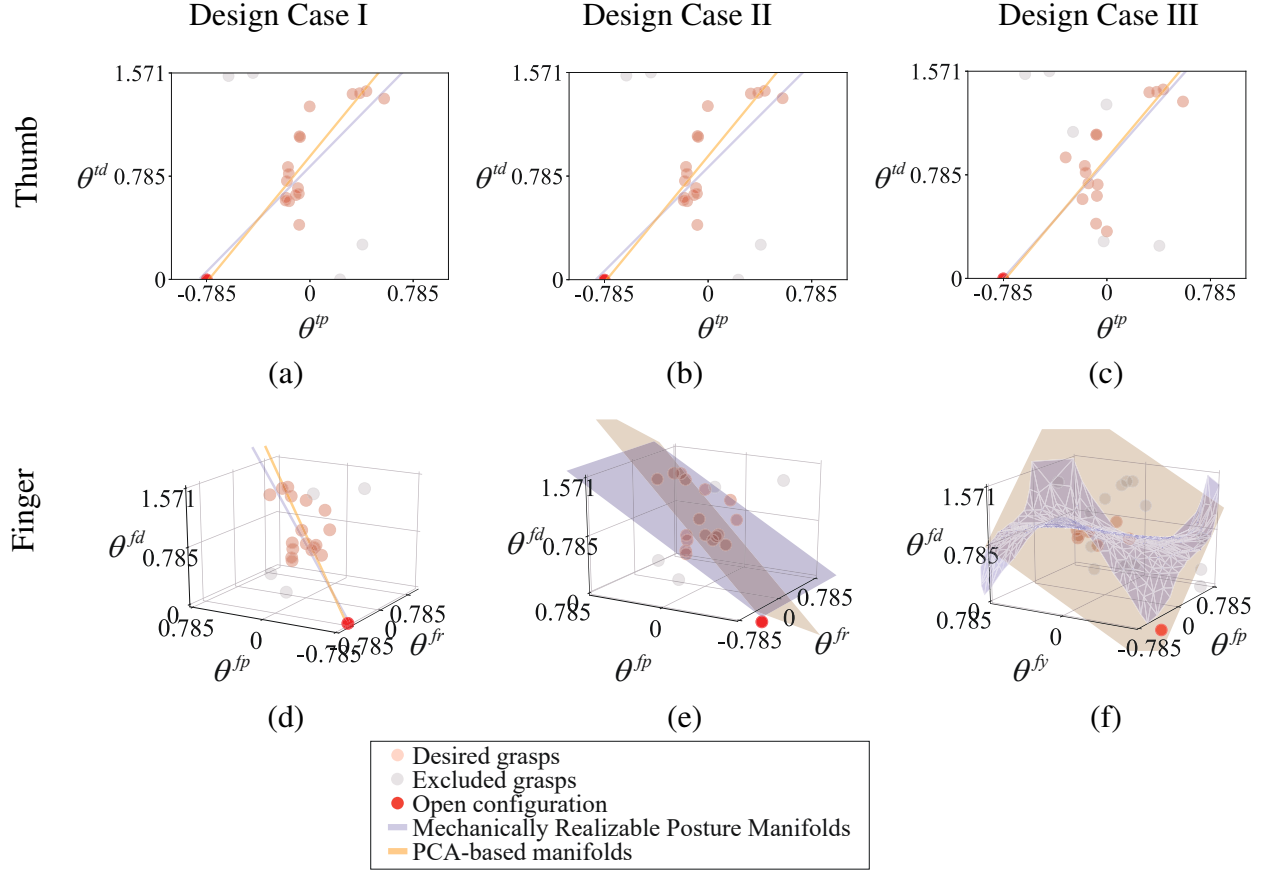


Figure 4.7: The visualization of the Mechanically Realizable Posture Manifolds. The first row shows 2D plots for the two joints in one thumb, and the second row shows 3D plots for the three joints in one finger. Different columns are different design cases. All angles are in radians.

grasps. The lines or surfaces in blue are the Mechanically Realizable Posture Manifolds, and the lines or planes in orange are the PCA-based manifolds.

4.5 Prototyping and Validation

We constructed physical prototypes for Design Case I and II, shown in Fig. 4.8 and Fig. 4.9.

In Design Case I, all joints are actuated by a single motor. Fig. 4.8 (and the accompanying multimedia attachment) demonstrates the finger trajectories, in which the hand first closes toward the center, making a spherical grasp posture and then a pinch grasp posture. Continuing to close, the fingertips do not collide but rather pass each other (due to motion in the roll degree of freedom). Finally, the hand creates an enveloping grasp. Fig. 4.10 (a) – (f) show several grasps using this

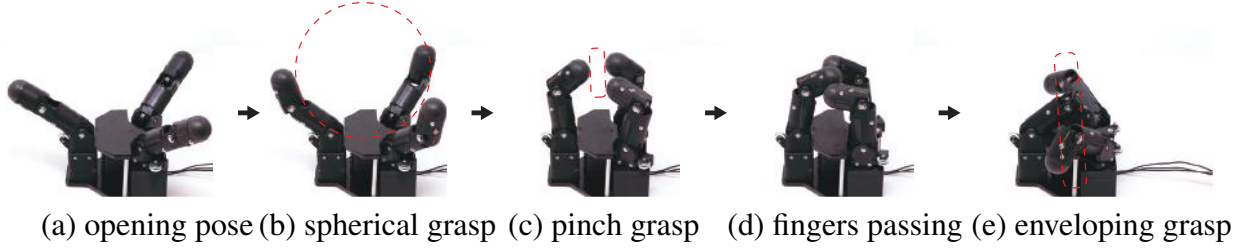


Figure 4.8: Finger trajectory and the types of grasp along the trajectory of Design Case I.

hand, displaying the versatility of the hand. We can see the hand can perform stable pinch grasps (d)(e), spherical grasps (c)(f), and power grasps (a)(b). We note that this interesting behavior that fingers first close towards then pass each other is not human-engineered but completely discovered by the data-driven optimization. This finger trajectory also makes this hand unique — to the best of our knowledge, this is the only three-fingered hand that can do spherical, (small) pinch, and power grasps just by one motor.

In Design Case II, the tendons of three fingers are connected to the pulley of the main motor, and the roll joints are actuated by a smaller motor via gear transmission (which shares the same mathematical expression as the antagonistic tendon transmission in Fig. 4.4 (b)). Therefore, the closing and rolling motion are controlled separately. As shown in Fig. 4.9, the hand can close the fingers towards the center to pinch small objects, or close the fingers in a parallel fashion so the finger can pass each other and make an enveloping grasp. Fig. 4.10 (g) - (l) show some example grasps of the resulted hand of Design Case II. We note that this design can perform pinch grasps of very small objects as shown in (j) and (k). We also highlight the fact that Fig. 4.10 also includes objects outside the desired grasp pool (e.g.(c)(h)(k)).

As for Design Case III, since the optimization result indicates that the performance is worse than the other two designs shown here, we did not build a physical prototype.

4.6 Discussion

The results for the three design cases demonstrate that our optimization method is effective: the proposed optimization framework can indeed shape the Mechanically Realizable Torque and

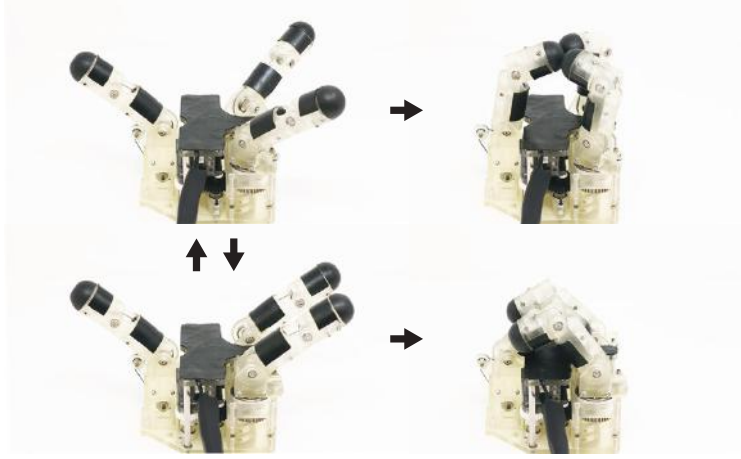


Figure 4.9: Finger trajectory of Design Case II. The first row shows the non-parallel closing, resulting in a pinch grasp. The second row shows the parallel-finger closing, resulting in an enveloping grasp. Unlike in Design Case I, the finger roll angle can be actively controlled in Design Case II, which leads to the different finger trajectories.

Postural Manifolds to fit the sample grasps.

Fig. 4.6 and Fig. 4.7 show that the optimized Mechanically Realizable Manifolds closely fit the data points defined by the sample grasps in both posture and torque spaces (except for the finger manifolds of Design Case III). In other words, the hands with corresponding underactuation designs can create stable grasps that are close to the desired ones in terms of both posture shaping and force generation. In each plot in Fig. 4.6, the distance between the manifold and the desired grasp points (not considering excluded grasps) is either exactly or close to zero, indicative of similarly low unbalanced joint torques.

The comparison with PCA results in Fig. 4.6 and Fig. 4.7 also illustrates the effectiveness of our method. Assuming linear manifolds, the PCA-based manifolds explain the most variance (measured by Euclidean distance) of the sample grasps. However, due to the constraints, those theoretically optimal manifolds cannot be reached exactly. In contrast, our method calculates manifolds that are close to the PCA results, by minimizing the aforementioned equilibrium-based metrics (instead of explicitly minimizing the Euclidean distances), and also considering physical constraints. In addition, the proposed method can also find nonlinear manifolds that make use of mechanism characteristics. For example, Design Case III shows the Mechanically Realizable



(a) Large screw driver



(b) Pliers



(c) Apple



(d) Food bag



(e) Pen



(f) Food can



(g) Wrench



(h) Multimeter



(i) Dremel



(j) Small screw driver



(k) Key



(l) Tape

Figure 4.10: Example grasps using the hand prototypes. (a) – (f): Design Case I, (g) – (l): Design Case II.

Manifolds that are not limited to the linear domain.

In Table 4.4, the numeric values of objective functions, as well as the number of grasps excluded due to being beyond the resulted hand's capability, provide evaluations of a design. Even

though the problem of initial kinematic design is out of the scope of our work (we require pre-specified kinematics), one can use these metrics calculated in our method to compare different kinematic designs, which provides insights for the choice of kinematics:

- Comparing the roll-pitch single motor (Design Case I) with dual-motor design (Design Case II), the results match our intuition: With the same kinematics, the dual-motor design (having a two-dimensional Mechanically Realizable Posture Manifold as shown in Fig. 4.7 (g)) can better span the joint posture space, i.e. has more versatility in terms of the postures (smaller numbers in column 2 and 3 in Table 4.4). For example, the corresponding hand can pick smaller objects as shown in Fig. 4.10 (j) and (k). Also, Design Case II displays better force generation capability in order to create stability for different grasps (smaller number in column 1 in Table 4.4), due to the decoupling between roll joints and the other joints.
- Comparing the roll-pitch configurations (Design Case I and II) with the pitch-yaw configuration (Design Case III), we can see that the pitch-yaw design is worse (larger numbers in all columns in Table 4.4), especially for the intra-tendon optimization. From this result, we can conclude that such a combination of pitch-yaw proximal joints with the certain tendon routing scheme in Fig. 4.5 has very limited capability, as the parameterization of this manifold does not have enough capacity to express most of the sample grasps under the given mechanical constraints.

Following this idea, we note that the numeric values of optimization results actually provide a “*hand quality metric*”. Given a set of grasps and hand kinematics and tendon transmission topology, we can use this metric to compare the potential of different kinematic design, and perform optimization on top of that.

The dual-layer optimization framework combining the non-convex stochastic global search in the outer-layer and the convex optimization in the inner-layer is a useful formulation. Except for specific kinematic designs such as the floating pulley transmission assumed in [9], it is generally not possible to formulate the parameter selection problem as a globally convex problem. In con-

trast, the dual-layer framework is capable of dealing with various kinematic configurations and actuation methods.

Our framework explicitly optimizes underactuation designs for a given set of sample grasps. This is useful for cases where the range of objects a hand is expected to interact with is largely known in advance (as is the case for the use case of a free-flying robot on the ISS). However, our experiments show that the resulting hands can also create stable grasps outside the set that they are explicitly optimized for, as illustrated in Fig. 4.10 (c)(h)(k). This suggests that, rather than “overfitting” to the sample grasps, the optimized manifolds fit between them in a useful fashion, extracting an inherent structure in posture and torque spaces that is useful for grasping common objects beyond the ones explicitly optimized for.

There are still limitations of this work. The first limitation stems from the need to manually prespecify hand kinematics and tendon connectivity patterns, which provides the parameterization needed to initialize our framework for a specific design. For example, in the three design cases discussed, the parameterization of actuation matrix \mathbf{A} and motor-tendon connection matrix \mathbf{M} are set manually, which makes it impossible to fully automate the framework. Second, our framework also starts from a list of manually defined sample grasps. The grasps we use here all have force closure, but are not optimized with respect to any other grasp quality metric (e.g. Ferrari-Canny metric).

Chapter 5: Underactuated Hands with Spring Agonists¹

The method of *Mechanically Realizable Manifolds* proposed in the previous chapter can only optimize the transmission parameters, but requires a predefined underactuation transmission scheme, specifically, the tendon routing paradigm for tendon-driven systems. Using different tendon transmission paradigms, the optimization method will find different Mechanically Realizable Manifolds.

In this chapter, we further investigate the tendon routing paradigms. we first formalize a design matrix of the tendon transmission topology, and then we propose a novel tendon transmission topology/connectivity based on this matrix, combining a classic way of tendons as prime movers (agonists) as well as a novel paradigm of springs as prime movers.

From the application perspective, we also further tailor the hand design to meet the specific requirements for the *Astrobee* free-flying robot in ISS, including power-off pose-keeping behavior and the possibility for human intervention.

5.1 Design Matrix of Underactuation Paradigms

Tendons can only pull. To enable bi-directional motion, the tendon can be replaced by a belt, or two motors can be used — one for each direction; however, both options negate the compactness that is so attractive in the use of tendons in the first place. In robot hands, particularly underactuated ones, designers often choose a different option: use a passive spring at each joint to account for one direction of motion. This means that only one direction is actively actuated, while the other is passive. However, that is generally acceptable for hands where movement in one direction (closing to grasp) is of more interest than its opposite (opening to release). Using the terminology of human physiology, we define the mechanism generating movement in the direction of interest as

¹This work is preprinted as [117].

the “*agonist*”, and the opposite side as the “*antagonist*”. In the rest of the study, we will consider finger flexion and adduction to represent the directions of interest for grasping.

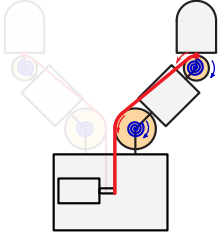
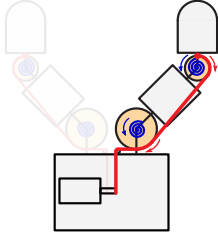
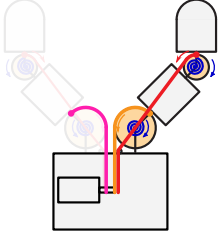
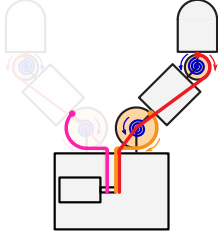
Most existing tendon-driven underactuated hands use the active tendons as agonists and passive springs as antagonists [32, 34, 35, 36, 11, 37, 38]. Furthermore, the most common underactuation scheme is to use a single tendon per motor, routed through multiple joints. While these paradigms are undoubtedly useful, in this study we wish to also explore their alternatives: where springs are used as agonists, and where multiple tendons are connected to the same motor shaft. Combining the two design dimensions mentioned above, we formalize a design matrix for tendon-driven underactuated hands. On one dimension, we have *Tendons as Agonists* (TA) versus *Springs as Agonists* (SA). On the other dimension, we have *Multiple Joints per Tendon* (MJT) versus *single joint per tendon, but Multiple Tendons per motor Shaft* (MTS).

We discuss this design matrix in detail as follows.

The first dimension of this design matrix is *the choice of agonist and antagonist*. In the most common way (e.g. [32, 34, 35, 36, 11, 37, 38]), active tendons are routed in the agonistic direction to drive the joint, while springs are used to restore the joints positions (“tendons as agonists (TA)”). The motor collects the tendons to close the fingers by overcoming the spring forces during pre-contact motion, and provides net grasping forces on top of spring forces after contacts are established. This choice can however be reversed: the springs can be used as agonists, or the prime mover for grasping, and active tendons can be used as antagonists (“springs as agonists (SA)”). In this case, springs are loaded and tendons are tensioned in one extreme of its range of motion, and the motor releases the tendon to let the springs drive the fingers.

The second dimension of this design matrix is *the coupling scheme across different joints*, i.e. the scheme of underactuation. One common paradigm is to have “Multiple Joints per Tendon (MJT)” (e.g. [32, 34, 35, 36, 11, 37, 38]). In each joint, the tendon goes around an idler pulley, or simply slides along fixed routing points, creating a torque that can drive the link to move. The alternative is to have a single joint per tendon, but connect “Multiple Tendons per motor Shaft (MTS)”, thus allowing a single motor to drive multiple joints.

Table 5.1: Design matrix of tendon transmission in underactuated hands.

| | Tendons as Agonists (TA) | Springs as Agonists (SA) |
|---|--|--|
| Multiple Joints per Tendon (MJT) | <p>TA+MJT</p> <ul style="list-style-type: none"> • Soft joint position coupling before contact [due to MJT] • Joint position differential, breakaway effect (i.e. distal joints can move if proximals are blocked) [due to MJT] • Joint torque coupling [due to MJT] • Grasping force regulation via motor torque [due to TA]  | <p>SA+MJT</p> <ul style="list-style-type: none"> • Soft joint position coupling before contact [due to MJT] • Joint position differential, breakaway effect (i.e. distal joints can move if proximals are blocked) [due to both MJT and SA] • Joint torque coupling [due to MJT] • Grasping force regulation is difficult [due to SA]  |
| Single joint per tendon w. Multiple Tendons per motor Shaft (MTS) | <p>TA+MTS</p> <ul style="list-style-type: none"> • Precise joint position coupling [due to MTS] • No joint position differential (breakaway) [due to MTS] • No joint torque coupling, torque differential [due to MTS] • Grasping force regulation via motor torque [due to TA]  | <p>SA+MTS</p> <ul style="list-style-type: none"> • Precise joint position coupling [due to MTS] • Joint position differential, breakaway effect (i.e. some joints can move if others are blocked) [due to SA] • No joint torque coupling, torque differential [due to MTS] • Grasping force regulation is difficult [due to SA]  |

Shown in Table 5.1, these two dimensions result in four combinations:

- **TA+MJT.** This is the most commonly used paradigm in underactuated hand design, and its advantages are well known. First of all, MJT provides a position differential (breakaway behavior) between joints: whenever a proximal link is blocked by objects, the motor can continue to drive the distal links to close, enabling adaptive grasping. It is important to look in more details at the problem of joint position coupling *before contact*. Based only on the tendon, this coupling is ill-defined: for a given tendon length, and without external forces, there are multiple possible combinations of acceptable joint positions. However, when antagonist springs are added, the problem becomes well-posed: for a given tendon length, the hand will take deterministic joint positions based on spring forces as well as other effects such as joint friction, tendon friction, etc. Thus, there is a "soft" position coupling between joints, one that can be exploited but also one that is sensitive to internal and external disturbances. Finally, since the tendon force is shared by multiple joints, the ratio of joint

torques from the tendon is always equal to the ratio of tendon moment arms (the spring also affects the overall joint torque, but as shown in 3.3, the net torque for grasping is only determined by the moment arm). This feature is a double-edged sword: we can carefully design this ratio and make use of it to make stable grasps, but the constraint that torques must follow this ratio can also be problematic in some applications (we further explain this in our design example in the following subsection).

- **SA+MJT.** This paradigm is more rarely used in robotics compared to the one above. Compared to TA+MJT, it has the additional disadvantage that grasping force regulation is difficult: grasping forces are determined by springs when the contacts are made and tendons get slack, and thus can not be controlled by the motor(s). However, it does present the advantage of being able to maintain a grasp in the absence of motor torques [118]. Due to the same reason, this paradigm is more well-known and used in the field of human-powered hand prosthetics [119].
- **TA+MTS.** This design is rare. The major limitation is that there is no joint position differential: whenever one joint is blocked by the object, the motor and thus all joints must stop. This results in the incompetence of adapting to object shapes. However, the MTS design still have several notable features: first, it provides precise joint position coupling because the kinematic relationship between joints are well-defined; second, the joint torques are not coupled (i.e. it has torque differential) since joints are actuated by independent tendons and each tendon can exert an arbitrary tension. These features can be favorable or undesired depending on the application, but the MTS design does provide a possibility of position coupling *without* torque coupling.
- **SA+MTS.** This scheme is also heavily underexplored but of particular interest. Compared to TA+MTS which does not have position differential, the SA design in this scheme is advantageous since can provide *position differential via tendon slack*: as the antagonistic tendon releases to enable joint motion by the spring, whenever the joint is stopped by the object, the

tendon will get slack, which breaks the transmission from the motor and allows the motor (and thus other joints) to continue. This feature enables adaptive finger motion similar to the MJT paradigm. In addition, this design also has the precise position coupling, and also allow the decoupled joint torques, provided by MTS paradigm as explained above.

5.2 Unmet Design Requirements for the Applications in the International Space Station

Though two functional prototypes are built in the previous chapter, there are a few application-specific requirements for the International Space Station (ISS) that are not satisfied by these designs:

- The overall dimension needs to be further reduced. This requirement rules out the possibility of Design Case II which needs two motors. Therefore we are looking at a single-motor design.
- The hand behavior needs to be robust to external disturbances — specifically, it should not be affected by gravity in order to be tested on earth. This is a major issue of Design Case I: this design forced the roll (abduction-adduction) torques and proximal and distal joint torques to be coupled, which is problematic: when the grasp is established and loaded, the proximal and distal joint should have a certain amount of torques, but adduction torques are also undesirably increased. As a compromise (discovered by the optimization algorithm), we ended up with a tiny pulley for roll joints, which, on the other hand, made the behavior sensitive to external disturbances (e.g. gravity, friction) and manufacturing errors.
- The hand needs to keep its pose after power-off. One major use of this hand is to grab and perch the *Astrobee* robot on a handrail in ISS, and for energy concerns, the hand should keep its grasp after power-off. However, neither of Design Case I and II satisfy this requirement, because the restoring springs in joints will drive the hand to the open pose.
- The hand also must be backdrivable to allow human intervention after power-off. This is

based on safety concerns and the ease of human-robot interaction. Neither of Design Case I and II satisfies this requirement.

5.3 Tendon Transmission Combining Different Paradigms

In the new design, we still use a single-motor three-finger eight-joint kinematic scheme (similar to Design Case I). Our *key innovations* in this design for the tendon transmission are as follows (illustrated in Fig.5.1):

- For all finger proximal and distal joints, we choose the TA+MJT paradigm, in which one tendon goes through an idler in the proximal joint and wrap around and fixed to the distal joint pulley.
- For the two roll (abduction-adduction) joints, we choose the SA+MTS paradigm, in which each joint connects to the motor via an independent tendon, and these joints are actuated by springs agonists.
- On the motor shaft, the tendons for finger flexion (proximal and distal joints) and the tendons for rolling (abduction-adduction) are winded in opposite directions, and when one set of tendons are collected, the others are released.

This design can result in the following favorable features regarding underactuation and grasping behavior:

- The *position differential (breakaway)* for different joints in a finger is achieved thus fingers can adapt to object shape mechanically — both the TA+MJT in flexion (proximal and distal) joints and SA+MTS in roll (abduction-adduction) joints have position differential. Here we elaborate more on the breakaway phenomenon by the SA+MTS design in roll (abduction-adduction) joints: whenever the adduction is blocked by objects, the tendons connected to these joints will get slack, while the proximal and distal joints can continue to flex the fingers.

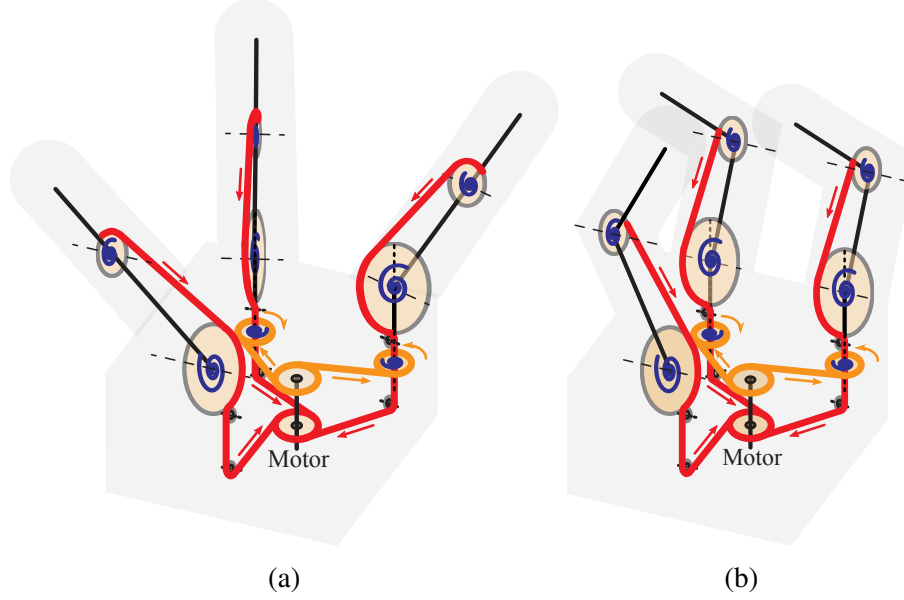


Figure 5.1: The tendon routing design of the proposed underactuated hand: (a) fully open, (b) partially closed. Red tendons are for finger flexion and orange tendons are for finger abduction-adduction. Please pay attention to the tendon winding directions and the finger abduction-adduction motion.

- The *position coupling* of all joints during free-motion is achieved thus all joints can follow a pre-designed trajectory driven by one motor. We note that, for SA+MJT joints, the position coupling is soft, while for SA+MTS this coupling is strict. Moreover, this design takes advantage of MTS paradigm to couple the roll (abduction-adduction) joints on different kinematic chains, illustrated in the second row of Table 5.1.
- Finger *roll (abduction-adduction) torques are not coupled with the flexion joints*, realized by the MTS design in roll (abduction-adduction) DoFs. The reason why this is essential is explained right before this subsection.
- The *grasping force regulation* via motor is achieved using the TA+MJT design across the flexion joints.

5.4 Spring Cancellation

In addition to the desired underactuation properties and grasping behavior, this design has another advantage: the restoring springs in flexion joints and the rolling (abduction-adduction)

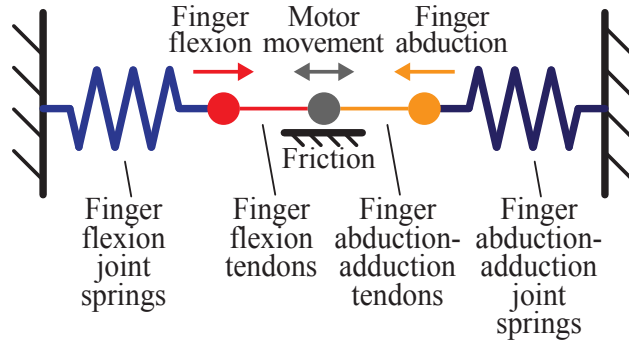


Figure 5.2: One-dimensional analogy of the tendon-spring system to illustrate the spring force cancellation.

joints are counter-acting on each other via the tendons. Therefore, with a proper choice of spring stiffnesses and preloads, there is a possibility that most of the spring forces can be canceled out and the remaining can be balanced by motor gearbox stiction. Fig. 5.1 shows the actual tendon routing (please note the opposite tendon directions on the motor pulley) and Fig. 5.2 is an one-dimensional illustration of this phenomenon.

The spring cancellation can bring the benefits of both power-off pose-keeping and backdrivability. When the motor is off, if the external forces are not strong enough to break the gearbox stiction (e.g. when perching on a handrail), the hand is neutral by itself and will keep the formed grasp; if the external forces exceed some thresholds, for example if a human is backdriving the hand, the stiction of motor gearbox can break and the hand pose can be changed.

Based on the availability of springs from the manufacturers, we select the adduction springs and design their preload angles to ensure that the gearbox stiction is larger than the difference between torques from restoring springs in flexion-extension joints and the rolling (abduction-adduction) joints, over the entire range of motion. In Fig. 5.3, we plot the torque from flexion-extension tendons (red), the torque from abduction-adduction tendons (orange) and the resultant torque (gray) (all converted to the motor shaft). Considering motor gearbox stiction in both motor driving directions, we mark the “qualified zone” with its upper and lower bound values being maximum gearbox stiction, shown as the gray highlight area in Fig.5.3. Note that the gray line is well enclosed by the "qualified zone" over the entire motor travel range, through which spring force cancellation is guaranteed.

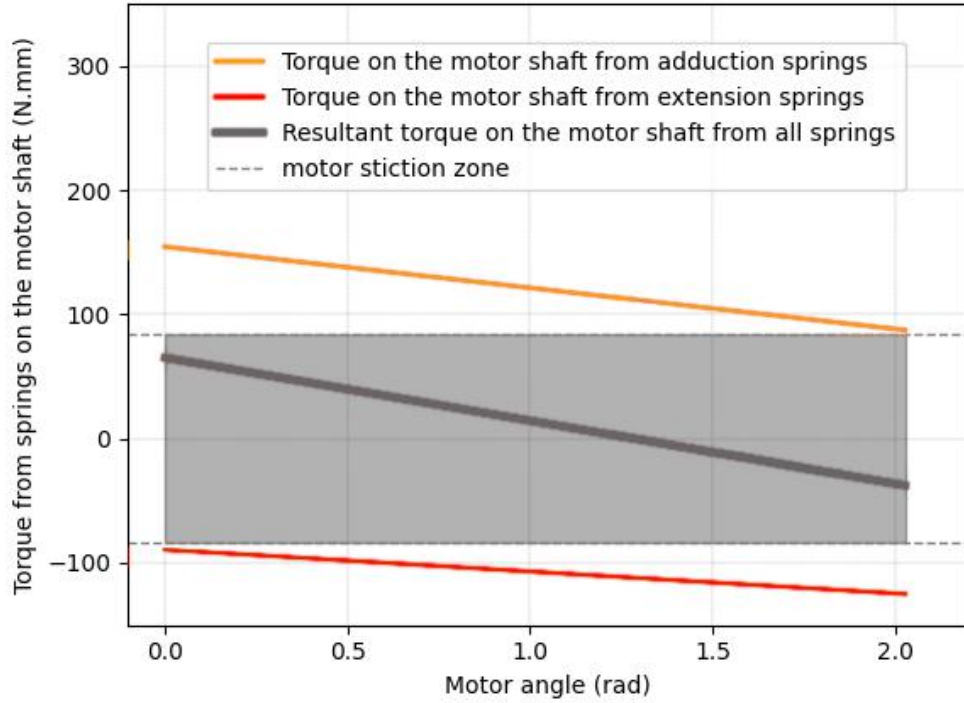


Figure 5.3: The plot of torques from springs on the motor shaft over the entire motor travel angles (motor angle 0: hand fully open; motor angle $2\pi/3$: hand fully closed).

5.5 Prototyping and Validation.

We construct a physical prototype with the proposed design, shown in Fig. 5.4. This hand is 3D-printed with resin (Formlabs Grey Pro), actuated by a single position-driven servo motor (Dynamixel XL430-W250-T), and controlled by an on-board microcontroller (PJRC Teensy 4.0) with custom circuits for controller-motor and controller-master communication.

Some example grasps are shown in Fig. 5.4 with a variety of objects, including both fingertip grasps and enveloping grasps.

Experiment data shows that the gearbox in the servo motor can resist maximum 84 Nmm torque with its stiction. We selected appropriate abduction-adduction springs based on this value. As shown in Fig. 5.5 and Fig. 5.6, the hand can effectively keep pose after power-off, by virtue of the spring force cancellation. Especially, we show the unpowered perching on an ISS handrail in Fig. 5.5.

As shown in Fig. 5.6, due to all-pose equilibrium after power-off, the hand can be manually



Figure 5.4: The example grasps with different objects.

shaped by humans to either make or release a grasp, providing operation convenience and human safety in the ISS application.

In collaboration with NASA, we were able to perform a preliminary integration test on an *Astrobee* Robot, as shown in Fig. 5.7. We successfully set up the mechanical and electrical connection as well as communication to the *Astrobee* robot. The hand prototype showed good performance in common object grasping.

5.6 Discussion

The design with the combination of TA+MJT and SA+MTS underactuation paradigms demonstrated stable grasps for a variety of objects, compact size, pose-keeping behavior and backdrivability to allow human intervention. These features satisfy the requirements of the ISS application.

Our experiments validate that the joint synergies are working as expected (as shown in Fig.5.4),



(a)



(b)

Figure 5.5: The power-off pose-keeping behavior in different hand configurations (note the power cable is unplugged). Particularly, (b) shows power-off perching on an ISS handrail.



(a)



(b)

Figure 5.6: Power-off human intervention: (a) manually close to grasp, (b) manually open.

while spring force cancellation is taking effect allowing energy saving and manual backdrivability when powered off (shown in Fig. 5.6). We also note that the hand can create a stable grasp on the ISS handrail when powered off (shown in Fig.5.5), enabling the power-off perching behavior similar to [118] while significantly improved the grasping capability.

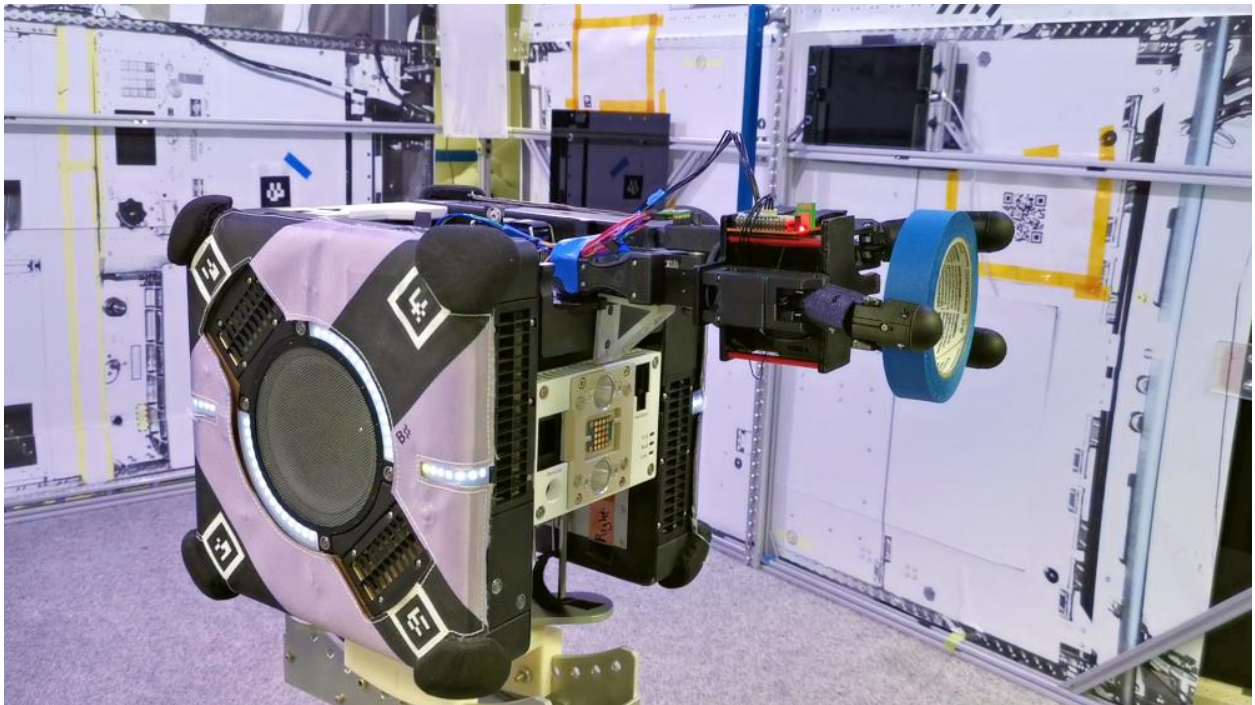


Figure 5.7: Preliminary integration test with the *Astrobees* free-flying robot.

Chapter 6: Hand Underactuation Design: Conclusion and Future Work

6.1 Contributions

First and foremost, we summarize the concrete contributions of Part I:

- To the best of our knowledge, *we are the first to propose a “mechanical dimensionality reduction” method — the “Mechanically Realizable Manifolds”* — to determine both
 - pre-contact postural synergies, and
 - post-contact joint torque coordination schemes,

which are *guaranteed to be mechanically realizable in tendon-driven underactuated hands*.

- We formalize a design matrix for tendon-driven underactuated hands, considering different tendon winding directions in each joint (TA vs. SA) and different tendon routing schemes across multiple joints (MJT vs. MTS). We show how different cells in this matrix have different properties, which can be leveraged to obtain desired behaviors.
- Taking ideas from this design matrix, *we are the first to present a design combining tendon agonists and spring agonists on different joints in one underactuated hand*. Specifically, we propose the use of TA+MJT for flexion/extension and SA+MTS for abduction-adduction, which facilitates the implementation of synergies in highly underactuated designs, and also allows for optimized spring cancellation for pose-keeping and human-intervention.
- We illustrate the concepts above on a physical prototype designed for a real use case: manipulation and perching for an assistive free-flyer robot in the International Space Station (ISS).

6.2 On-Going and Future Work

Potential Improvements

Automatic Grasp Generation. One of the on-going works to improve our method is the automatic desired grasp generation using GraspIt!. We already have a computational infrastructure that can automatically generate thousands of stable grasps using random search [120]. After that, we need to extract the useful matrices from this massive pool of grasps.

Dimension Optimization. During the generation of sample grasps, another improvement of the existing method is that we can search over the hand dimensions, such as link lengths and finger distances. These dimensions are human-specified in the existing work because it needs to fit the *Astrobee* payload bay, but for general-purpose hand design, dimension optimization is important.

Realization of Non-linear Manifolds Using Non-circular Pulleys In Design Case I, II and the final design, we used circular pulleys which can only express linear Mechanically Realizable Manifolds (both the position and torque of a joint are linear to those of other joints with a ratio of pulley radii). However, the linearity limited the expressive capability of such manifolds to fit arbitrary grasp data which is likely to be nonlinear. As shown in Fig. 6.1, a notable phenomenon is that the rolling (abduction-adduction) joints always turn in a uniform velocity during hand closing, while the desired behavior is that the rolling (abduction-adduction) joint first turns slowly to let the two fingers get close and make a good opposition to the thumb for pinching tiny objects and turns then quickly to let the fingers pass the thumb to make enveloping grasps.

We propose the idea to use non-circular pulleys to realize the non-linear manifolds and the non-uniform motion, parameterized by an array of radii. As shown in Fig. 6.2, we subdivide the useful arc of a pulley into several segments, and the radii of a point on the arc is a linear interpolation of the two adjacent anchor points.

This requires some changes in the mathematical formulations. For example, the torque generated by the tendon becomes the cross product of the radius vector and the tendon force vector

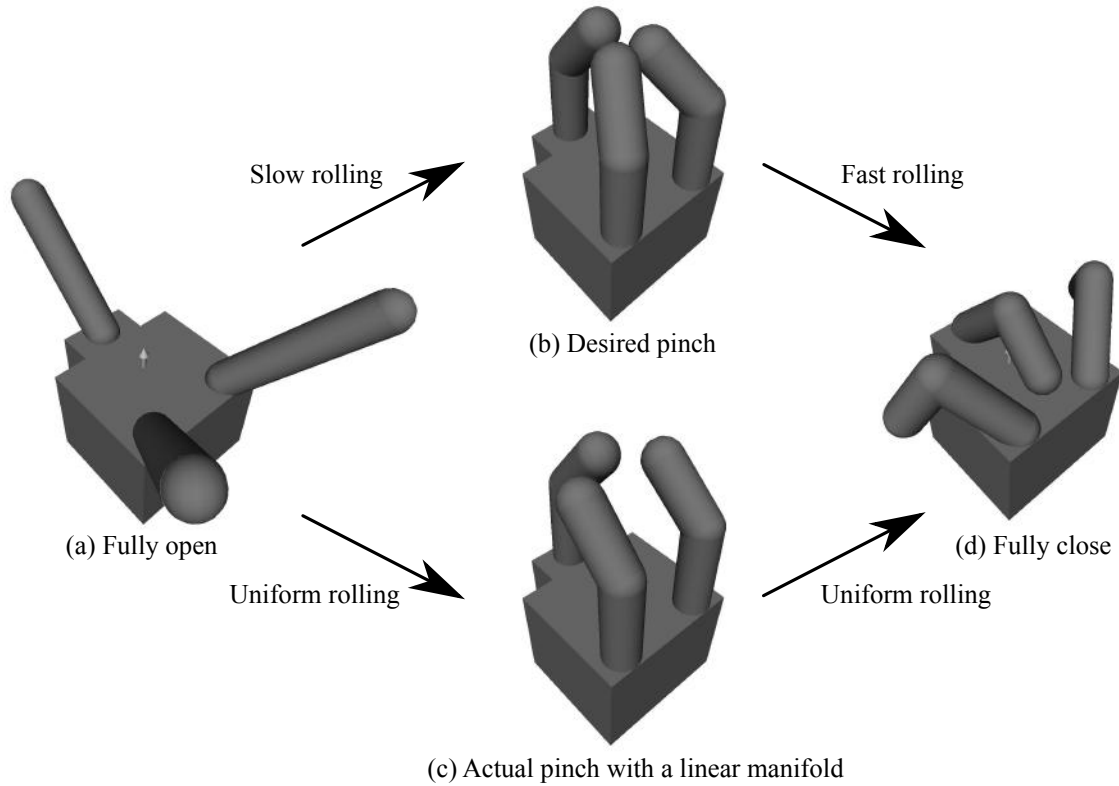


Figure 6.1: Desired (non-uniform) finger rolling (adduction) vs. uniform finger rolling using a linear manifold.

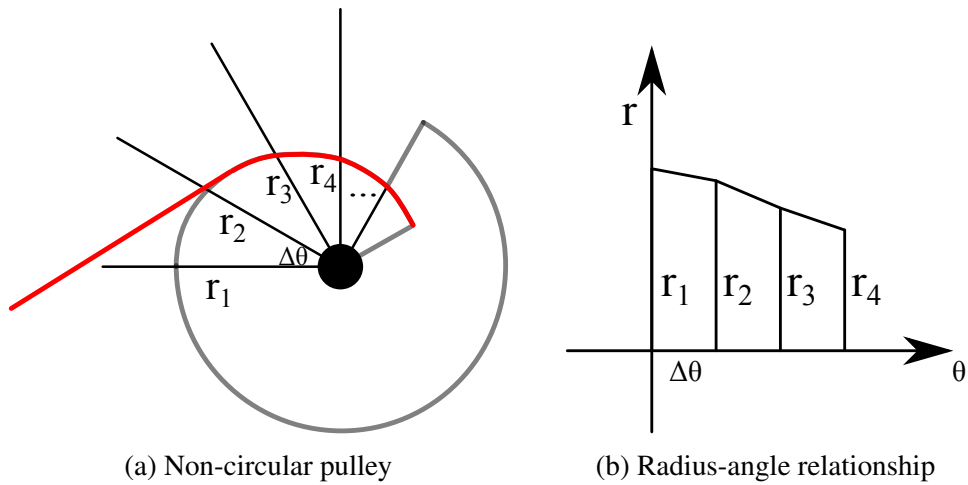


Figure 6.2: Non-circular pulley

(instead of a simple scalar product), the tendon travel becomes an integration (instead of a simple scalar product of angle and radius). However, the three-step optimization framework, the dual-layer structure and the optimization methods keep the same as the existing work.

Long-term vision

A long-term goal is to optimize the hand topology and tendon connectivity, such as the number of fingers, the number of links in each finger, and tendon connection pattern between motors and joints, etc. If we build this optimization on top of the existing work shown in this chapter, it means additional layers of hyperparameter search. Since these design decisions are discrete, evolutionary computation/Genetic Algorithms are good candidates.

However, formulating the hand topology optimization as additional layers of hyperparameter search will quickly fall into the “curse of dimensionality” (i.e. the computational load grows exponentially with the search dimensions). We aim to explore more efficient algorithms than hyperparameter search. In the Future Work section of the Chapter 14, we present some initial thoughts on graph-based search methods.

Part II

Compliant Force-Based Grasping

Chapter 7: A Series-Elastic-Actuated Hand with Mechanical and Computational Compliance¹

In addition to the mechanical coordination of different joints we discussed in the previous chapters, the compliance embodied in hands is another form of Mechanical Intelligence that helps the hand to grasp in an adaptive way.

Mechanical compliance is in nature connected to force/torque sensing and control, which is a kind of Computational Intelligence. In fact, deformation-based sensing is one of the major ways to sense the forces and torques. On the other hand, mechanical compliance is also one of the key factors of high-quality force/torque control.

The combination of passive compliance and active force/torque sensing and control is especially useful when considering blind grasping of unknown objects. Without any prior of the object exact location, shape, mass and inertia, friction coefficient, etc, we assume the only knowledge is that we know there is an object inside the graspable volume of the hand. This is to emulate the scenario where the vision gets blocked by the hand and arm after they are deployed. In this case, the only way to gather information and make a grasp is through compliant force-based interaction.

In this chapter, we present the hardware platform leveraging series-elastic actuation in hands, as well as the low-level controllers. In the next chapter, we will demonstrate the high-level control policy to perform compliant blind grasping.

7.1 Proprioception

For decades, people have been using vision extensively to guide robotic grasping, either in traditional ways that localize an object by vision, plan and execute the grasp, or in a learning fashion

¹The work presented in this and the next chapter is published as [121] in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

that takes raw images or point clouds as inputs. However, vision still has several limitations, for example, occlusion, sensitivity to lighting, transparency, glare, etc. Especially, after arm deployment, it is hard to use vision to guide the grasp because the object is usually heavily occluded by the hand and arm.

In contrast, our interest is to perform adaptive and reactive grasping using force/torque information. We believe force/torque information is a good complementary component to close the loop for vision, and it is even more interesting to study grasping with solely force/torque sensing. The ultimate goal is to understand the physical interaction with hands and objects, including touching exploration, object equilibrium reasoning, etc. Of course, the main focus will be multi-finger multi-joint hands instead of parallel hands for the sake of versatility and grasping stability.

We focus on proprioception. Human Proprioception refers to the ability to perceive the relative positioning of neighboring body parts as well as the muscle effort deployed to produce it. Together with vision and tactile sensing, it plays a unique and important role in the perception of human manipulation [122]. For robots, proprioception is translated as the combination of joint position and torque sensing (assuming a hand comprised exclusively of revolute joints).

Comparing with tactile sensing which is also a force-based sensing modality, joint torque sensing has the advantages of simpler hardware, negligible unsensed area/directions, higher refresh rate, off-the-shelf product availability, etc., while it has the disadvantage of a smaller amount of information.

Our hypothesis is that *hand proprioception alone can provide adequate feedback information to guide the physical finger-object interaction for grasping unknown objects*, under the assumption that arm and hand are already deployed close to the object(s) and the hand-object interaction is possible. We note that for multi-finger multi-joint hands, especially the ones with compliant and backdrivable joints (for safe interaction with unknown environments), the action of “grasping” is much more complicated than a one-DoF “closing” command for the parallel grippers, because the joint movements and torques need to be coordinated. Otherwise, the object will be squeezed out of the hand.

7.2 Mechanical and Computational Adaptation: Series Elastic Actuation

The simplest example of mechanical adaptation is through compliant designs. With appropriate compliance, the robot can interact with the environment robustly and safely, without a precise model of the environment, without precise position control, and even with unexpected disturbances. In the context of robotic hands, compliance is even more important: careful collision-free planning and control can enable robot arms to perform useful tasks without touching the environment, but for hands, this method defeats the purpose as hands are intended to physically explore and interact with the objects and obstacles.

In addition to the passive adaptation from mechanical compliance, we can also use active force/torque sensing and control to render an adaptive interface between the robot and the environment: it can not only render compliance, but also emulate adaptive underactuated mechanisms. In a word, active force/torque sensing, together with full actuation in all joints, can be a good alternative or an even more powerful counterpart of the passive mechanical adaptation scheme.

When these two types of adaptation patterns come together, new design ideas emerged. In the context of robotic hands (we only discuss hands in a traditional sense, not including soft pneumatic hands), there are two interesting paths to take: to put the compliance in the fingertip and use rigid transmissions, actuators and force sensors, or to put the compliance inside the transmission and actuation components. The former idea leads to the design of a category of force-sensed hands such as the Robonaut Hand [27], the DLR Hand II [70] and the *ROAM* hand we will introduce later; the latter leads to Series Elastic hands, which is the main topic of this chapter. As an interesting side note, the human hand is a combination of these two types.

Series Elastic Actuators (SEA) are first proposed in the 1990s [68], and is a design scheme known for high-fidelity torque control, shock protection, energy efficiency and human-safety. By nature, SEA is a type of proprioceptive actuators: it can give angle measurement by encoders, and torque/force reading by measuring the deflection of the elastic component. They are widely used in legged robots and rehabilitation systems, but rarely used in hands. We are interested in exploring

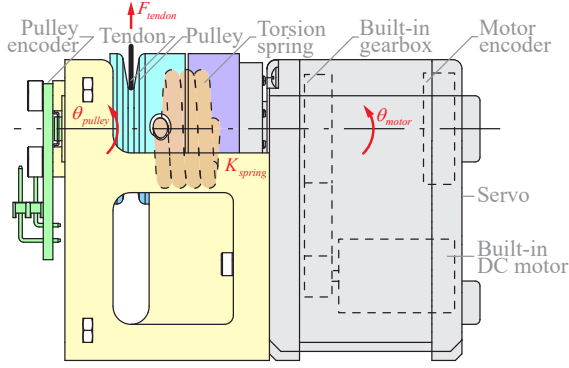


Figure 7.1: The SEA module, including a geared servo motor (in gray) with a built-in encoder, a torsion spring (in orange), an output shaft (in cyan), an output encoder (in green) and structural parts (in yellow).

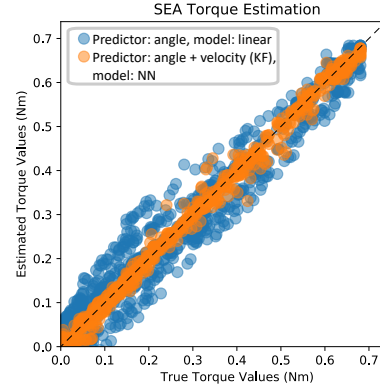


Figure 7.2: The prediction-vs-truth plot of SEA torque estimation, comparing a naive linear model (in blue) and a small neural network model (in orange).

using SEAs for grasping.

7.3 SEA Module Design

We first developed a simple and compact SEA module (Fig. 7.1). A position-driven servo (in gray, Dynamixel AX-12A) is used as the driving motor, which receives position commands and returns the current position (measured by the built-in potentiometer), i.e., θ_{motor} can be measured. A torsion spring (in orange, Lee Spring) is used as the elastic component, connecting the motor shaft (purple) and the output pulley shaft (cyan). A Spectra cable is tied on the pulley to transmit the force to the finger joint. An absolute magnetic encoder (in green, AMS AS5600-SO- EK-AB) is mounted on the end of the pulley shaft to measure θ_{pulley} .

Ideally, in steady-state, the SEA torque can be calculated as the product of spring stiffness and deflection:

$$\tau_{SEA} = K_{spring} \cdot (\theta_{pulley} - \theta_{motor}) \quad (7.1)$$

However, we encountered a severe problem of nonlinearity, especially the hysteresis. We believe this is a complex combination of stiction and dynamic friction between spring coils as well as the hysteresis from supporting material. We hypothesize that the hysteresis is velocity-dependent, so

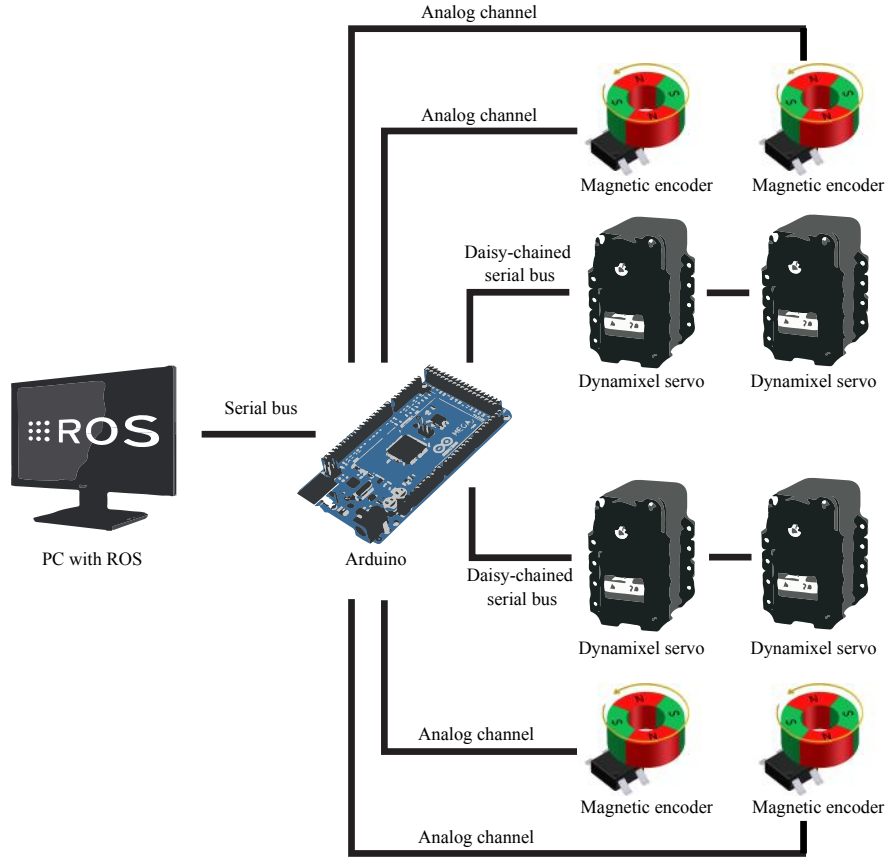


Figure 7.3: The electronics architecture.

we estimate the velocity by a Kalman Filter (using the naive dynamics — relating acceleration, velocity and position by integration). We collect random but smooth data, where angles are read from the SEA and the ground truth torques are read from a load cell. Then we send the deflection angle and velocity to a small Neural Network (NN) (two layers, 32 nodes in each layer). Our result shows this scheme provides better torque estimation, reducing the Mean Absolute Error (MAE) of torque by 74% comparing to the naive linear model. The prediction-vs-truth plot is shown in Fig. 7.2.

7.4 Electronics

In terms of the electronics architecture (shown in Fig. 7.3), the servo motors talk to the on-board microprocessor (Arduino Mega 2560) via daisy-chained TTL serial communication, and the magnetic encoders on the output pulleys of the SEAs send the signals via analog channels. A

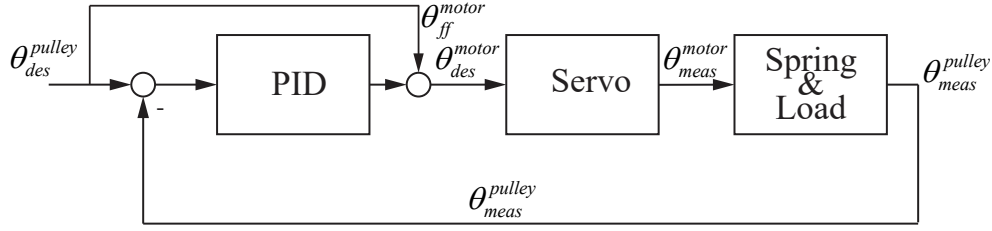


Figure 7.4: The low-level position control.

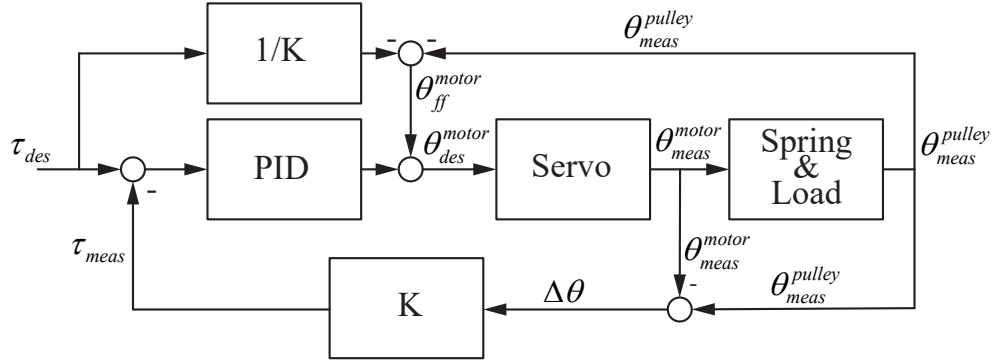


Figure 7.5: The low-level torque control.

custom shield board with necessary peripheral circuits was built for the TTL serial communication and the analog signals. The on-board electronics is in charge of all low-level processing and communicates with a Linux PC with Robot Operating System (ROS) via a serial bus.

7.5 Low-Level Control

There are three low-level control modes: motor position control, pulley position control and torque control, and they can be switched online. The motor position control is to send position command to the servo directly, making use of its built-in position controller. The pulley position control is a proportional-integral-derivative (PID) loop with the desired pulley position as feedforward, shown in Fig. 7.4 (a). The torque control is also a PID loop with a feedforward term which is the ideal motor position calculated by the current pulley position, desired torque and spring stiffness, shown in Fig. 7.5 (b).

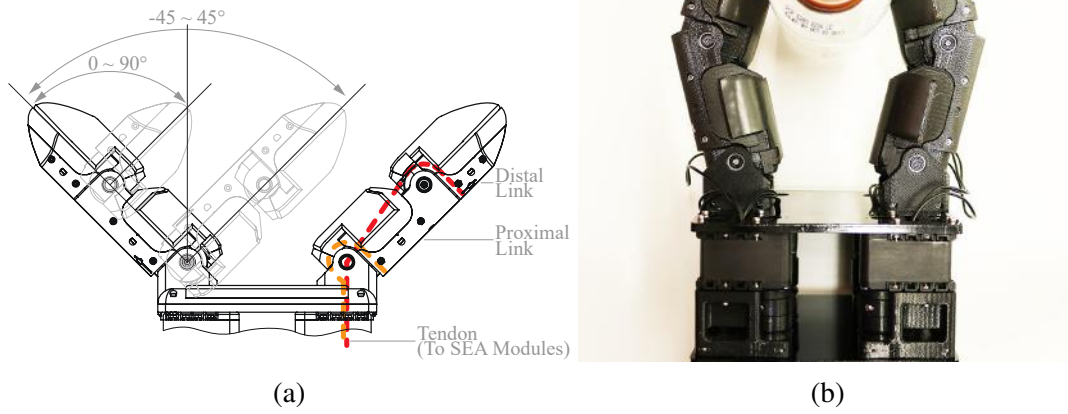


Figure 7.6: The schematic and photo of the SEA hand.

7.6 Finger and Hand Design

We developed a planar hand with two fingers and two links in each finger, shown in Fig. 7.6a. The zero position of the hand is defined as the configuration in which all links are perpendicular to the palm. The proximal and distal joints have a range of motion of -45 to 45 and 0 to 90 degrees respectively. In each joint, flexion is powered by the tendon (shown as colored dash lines in Fig. 7.6a) connected to the SEA pulley, and the extension is driven by a restoring spring. The tendon connected to the distal joint (the red dash line) goes right through the axis of the proximal joint so that the torques of proximal and distal joints are fully decoupled.

We note that we built joint-level or hand-level controllers on top of these low-level controllers. For example, joint position and torque control can be achieved by SEA pulley position and torque control with a simple linear conversion.

Chapter 8: Grasping with Series Elastic Actuation and Hand Proprioception

As discussed in the previous chapter, we focus on the problem of blind grasping using hand proprioception. In this chapter, we present the grasping controller using the aforementioned Series-Elastic-Actuated (SEA) hand. We show that passive compliance in the SEAs is crucial for force/torque control, and the active compliant behaviors using force/torque sensing and control can enable adaptive grasping with only hand proprioception.

Our hypothesis is that *hand proprioception alone can guide the physical finger-object interaction for different types of grasps with unknown objects*, under the assumption that arm and hand are already deployed close to the object(s) and the hand-object interaction is possible. By the “physical interaction” we refer to the way to apply grasping forces, the way to reason and keep grasp stability, etc. Besides, “different types of grasps” include the fingertip grasps (precision grasps) and enveloping grasps (power grasps). Also, “unknown objects” means there is no prior information about the object’s exact location (assuming it is inside the touchable range), shape, size, weight, inertia, friction coefficient, etc. This is to emulate the scenario where a simple vision system estimates a rough object location, the arm reaches out but blocked the view of the object – in this case, only force information can be acquired via interacting with the object.

8.1 The Proportional-Integral (PI) Network Grasping Controller

Problem Statement

The key desired feature of the proposed algorithm is to increase the torques applied at the joints (and implicitly the contact forces) in a stable fashion after making initial contacts. In other words, after initial contact, we need to find an increase in joint torques that produces no net wrench on the object. For a hand with multi-link fingers, this is not straightforward given that the joint torques

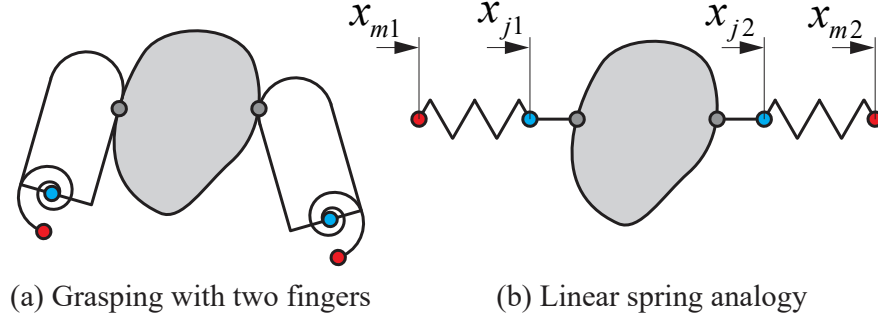


Figure 8.1: Illustration and analogy of grasping with SEA.

need to be coordinated in the absence of information on object shape and contact locations. It is necessary to note that we are not solving the contact planning problem, and we wish to develop a reactive control strategy that does not require pre-planning.

Our insight is that proprioception alone can characterize grasp stability. For an SEA-powered proprioceptive hand, an unbalanced net wrench will produce object movement against the compliant elements, which we can measure. Therefore, *we formulate the goal of grasp stability as the one of minimizing object movement while applying forces*. Since the servo motors only have position control, introducing the series elasticity becomes crucial. Also, since we do not have a direct measure of object movement in Cartesian coordinates by using only proprioception, we use the change of joint position in joint space (measured by SEA) during grasping as a proxy for object movement.

For intuition, consider the illustrations shown in Fig. 8.1: subfigure (a) shows a (simplified) scenario in which the motors (red dots) are driving the torsion springs, and the springs are pushing the fingers (blue dots) to make a grasp (gray dots). Subfigure (b) shows a one-dimensional illustration using linear springs, with similar color-coding as in (a). Here, we actively control the positions x_{m1} and x_{m2} (which translate to motor positions θ_{motor} on the real hand) to apply forces, and measure x_{j1} and x_{j2} (which translate to pulley positions θ_{pulley} linearly mapped to joint positions). We aim to keep x_{j1} and x_{j2} constant as we squeeze.

Controller Design

Our key insight is that the problem of grasping unknown objects can be solved even in the absence of contact information, by a multi-input-multi-output (MIMO) control with a network of proportional-integral (PI) controllers. Without knowledge of object geometry, contact locations and contact states, it is impossible to fully model the dynamic system analytically. However, a proprioceptive platform still provides sensory access to the variables that characterize the grasp stability, and the PI control framework provides ways to regulate these variables even though the analytical relationship is not constructed. We thus aim to use a feedback scheme operating exclusively in the sensory space of the robot, without explicitly modeling the physics of the hand and the object.

To enable the squeezing scheme that minimizes the object movement, we perform an initial light touch first, and then execute the PI Network Grasping Controller to load the grasp.

The first step — closing to create a light touch — is to put finger joints in a torque control mode with a very low torque reference (we use 0.05 Nm). In this way, the fingers will stop when touching the object. We wait for some time (we use 3 seconds) and then enter the second step — the PI Network Grasping Control.

Fig. 8.2 shows the block diagram of the PI Network Grasping Control loop. Here, the controller is constructed on top of the low-level sensing, so we consider the joint angles and torques are already obtained from SEA measurements. The reference \mathbf{u} consists of desired joint angle values $(\theta_{des}^{p1}, \theta_{des}^{d1}, \theta_{des}^{p2}, \theta_{des}^{d2})$ where the superscripts p represent proximal and d represent distal joints) and reference joint torques $(\tau_{des}^{p1}, \tau_{des}^{d1}, \tau_{des}^{p2}, \tau_{des}^{d2})$. We will discuss the choice of these reference values in the next paragraph. The feedback \mathbf{y} has the same structure, but contains actual measured values. The desired joint angles (first half of the reference vector \mathbf{u}) are used as the heuristic-based feedforward term. The error between the reference \mathbf{u} and feedback \mathbf{y} is fed into a network where each edge is either a proportional (P) or proportional-integral (PI) controller. The outputs of the network controller block and the feedforward term are summed up as motor position command \mathbf{c} and sent to the motors. After that, the actual motor position vector \mathbf{d} goes to the black-box system

of the hand and the unknown object.

$$\mathbf{c}(t) = \mathbf{F}\mathbf{u}(t) + \mathbf{K}_p\mathbf{e}(t) + \mathbf{K}_i \int_0^t \mathbf{e}(\tau) d\tau \quad (8.1)$$

Here, $\mathbf{F} = \begin{bmatrix} \mathbf{I}_{4 \times 4} & \mathbf{O}_{4 \times 4} \end{bmatrix}$ is the feedforward matrix, \mathbf{K}_p (4×8 matrix with all entries being non-zero) and $\mathbf{K}_i = \begin{bmatrix} \mathbf{K}_{4 \times 4} & \mathbf{O}_{4 \times 4} \end{bmatrix}$ (where \mathbf{K} is a matrix with all entries being non-zero) are proportional and integral gain matrices, and \mathbf{e} is the 8×1 error vector between \mathbf{y} and \mathbf{u} . After that, the actual motor position vector \mathbf{d} goes to the black-box system of the hand and the unknown object.

In the reference \mathbf{u} , the desired joint angle values $(\theta_{des}^{p1}, \theta_{des}^{d1}, \theta_{des}^{p2}, \theta_{des}^{d2})$ are equal to those in the initial touch configuration, while the reference torques $(\tau_{des}^{p1}, \tau_{des}^{d1}, \tau_{des}^{p2}, \tau_{des}^{d2})$ are chosen as the maximum motor torques. A special design of this controller is that we require the joint angles to be regulated exactly to the setpoints, but do *not* require the torques to be so. *We allow and make use of the steady-state error of pure proportional control.* In this way, the reference torques do not need to be a legal set of torques that result in equilibrium — actually, we are not able to design such a legal set of torques due to the absence of contact or object information. We let the law of dynamics decide the steady-state values for torques, and let the system balance itself automatically.

As for the tuning of the control gains, it is possible to set up a learning or optimization process, but we found that even hand tuning works well enough. The tuning is not as complicated as it seems. First, due to hand symmetry, the number of independent parameters is cut by half. Second, the P and I gains have the same effect as the conventional single-input-single-output systems, and tuning heuristics also apply here, for example, we can take spirits from the Ziegler-Nichols method [123]. Third, the parameters can be tuned in groups, and we formulate every gain as a product of a baseline value and a weight coefficient. The baseline values are set to be the same if the input entries corresponding to those gains have the same physical dimensionality, and the weight coefficients are tuned based on their relative importance. Last and most importantly, as we found in practice, the controller performance is not sensitive to the parameters as long as they are within a reasonable range.

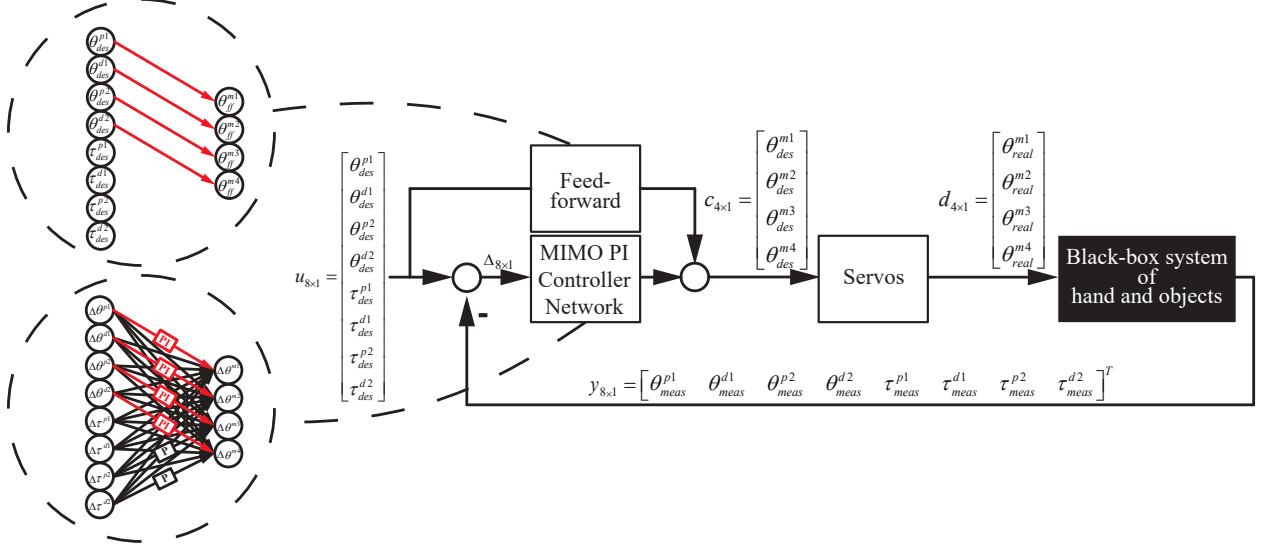


Figure 8.2: The PI Network Grasping Controller. Subscripts: des—desired value, meas—measured value, real—real value. Superscripts: m—motor, p—proximal link, d—distal link.

The structures of the gain matrices are as follows:

$$K_p = \begin{bmatrix} w_1 b_1 & w_2 b_2 & w_2 b_1 & w_2 b_2 & w_3 b_3 & w_4 b_4 & w_4 b_3 & w_4 b_4 \\ w_2 b_1 & w_1 b_2 & w_2 b_1 & w_2 b_2 & w_4 b_3 & w_3 b_4 & w_4 b_3 & w_4 b_4 \\ w_2 b_1 & w_2 b_2 & w_1 b_1 & w_2 b_2 & w_4 b_3 & w_4 b_4 & w_3 b_3 & w_4 b_4 \\ w_2 b_1 & w_2 b_2 & w_2 b_1 & w_1 b_2 & w_4 b_3 & w_4 b_4 & w_4 b_3 & w_3 b_4 \end{bmatrix} \quad (8.2)$$

$$K_i = \begin{bmatrix} w_1 b_5 & w_2 b_6 & w_2 b_5 & w_2 b_6 & 0 & 0 & 0 & 0 \\ w_2 b_5 & w_1 b_6 & w_2 b_5 & w_2 b_6 & 0 & 0 & 0 & 0 \\ w_2 b_5 & w_2 b_6 & w_1 b_5 & w_2 b_6 & 0 & 0 & 0 & 0 \\ w_2 b_5 & w_2 b_6 & w_2 b_5 & w_1 b_6 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (8.3)$$

We pick $b_1 = 0.2$, $b_2 = 0.5$, $b_3 = 4.0$, $b_4 = 8.0$, $b_5 = 1.0$, $b_6 = 1.0$, $w_1 = 1.0$, $w_2 = 0.3$, $w_3 = 1.0$, $w_4 = 0.5$ for our hardware. As a side note, we learned from practice that the performance is not sensitive to this set of parameters, and there is a reasonably large range of feasible parameters.

8.2 Experiments and Results

In this section, we validated the capability of this SEA hand of performing fingertip grasps, enveloping grasps, and the transition from the former to the latter. We note that in this study we only consider a two-dimensional scenario, in which the objects are confined to move only in the plane of the fingers.

Fingertip Grasps

The goal of this experiment is to test the hypothesis that the PI Network Grasping Controller is effective in fingertip grasping for unknown objects. We compare against our baseline: an emulated underactuated hand running a Fixed Torque Ratio Controller.

We emulate a common type of underactuated hands (tendon-pulley-driven, without special designs such as stoppers or clutches) by the Fixed Torque Ratio Controller, where the torques applied to proximal and distal joints always follow a certain ratio. When this kind of hand makes grasps, the configuration-dependent torques from the extension springs can be ignored, as they are usually much smaller than the flexing torques from the tendons. Therefore, the net joint torques in the proximal and distal joint have a configuration-independent and design-time-fixed ratio, which is the ratio between the joint pulley radii.

Experiment Protocol. Our experiment proceeds as follows. We execute the grasping in two phases. In the first phase (approaching and touching), the fingers are set in torque control mode with very low reference torques so that they stop when they touch the object. In the second phase(squeezing), the hand executes the PI Network Grasping Controller or the emulated underactuated hand in the fully-actuated testbed for comparison.

There are a lot of factors that may influence performance. To have a well-rounded comparison, we compare the performance of the PI Network Grasping Controller with the emulated underactuated hand, covering different object shape primitives, different object locations, different initial hand poses, and different friction coefficients. Please see Appendix B for the details.

We use two metrics to evaluate the performance:

- *Success rate.* The success rate is our primary performance metric. We define a “success” if the hand finally settles down in equilibrium with the object in hand after squeezing. This definition includes three scenarios: (i) the hand keeps the object in fingertips near initial touch pose, without converting to enveloping grasp or reaching joint limits, (ii) the hand holds the object but re-configures to an enveloping grasp, and (iii) the hand keeps the object

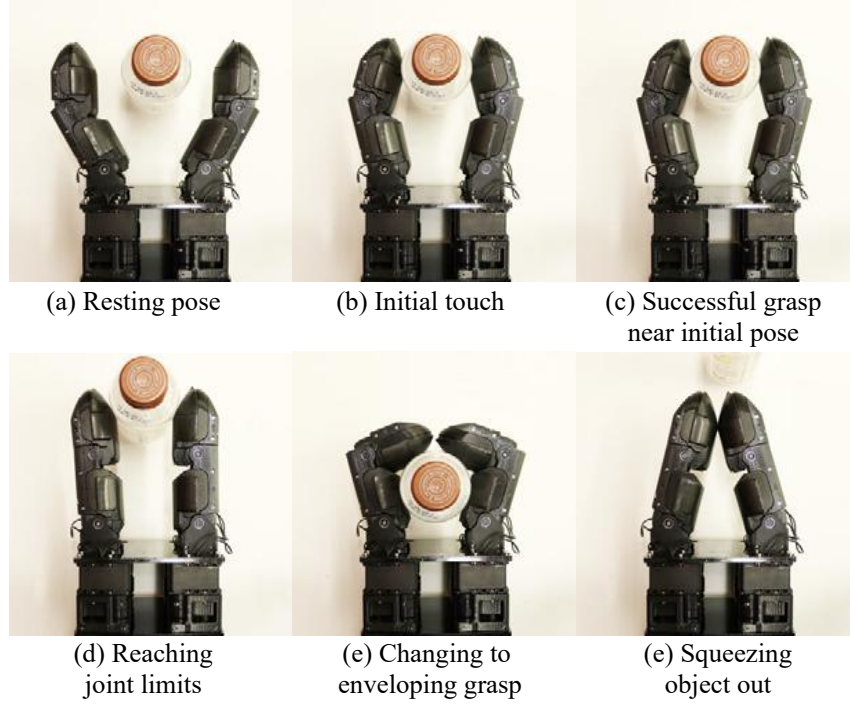


Figure 8.3: Typical scenarios in the fingertip grasp experiments.

in fingertips but reaches a mechanical joint limit (thus the joint torque ratio changes). These cases are all considered successful but still need to be distinguished. Case (i) is the most desirable, while (ii) and (iii) mean the grasp is not stable at the initial pose and relies on reconfiguration to be balanced.

- *Hand pose change.* We believe it is also useful to keep the object in the same pose as when the first contact is made. We thus use a secondary performance metric that evaluates how much the object moves in the hand during the squeezing process. Without access to object pose in Cartesian space, we measure this as the change in hand pose between initial touch and final grasp (Euclidian distance in four-dimensional joint space). This metric is only calculated and averaged for the successful cases.

Results. Fig. 8.3 shows the photos of some typical scenarios in the fingertip grasping experiments. (a) and (b) shows the resting pose and the initial touch. (c) shows the successful grasp near the initial touch pose. The other images show the cases in which the object was kept in fingertip

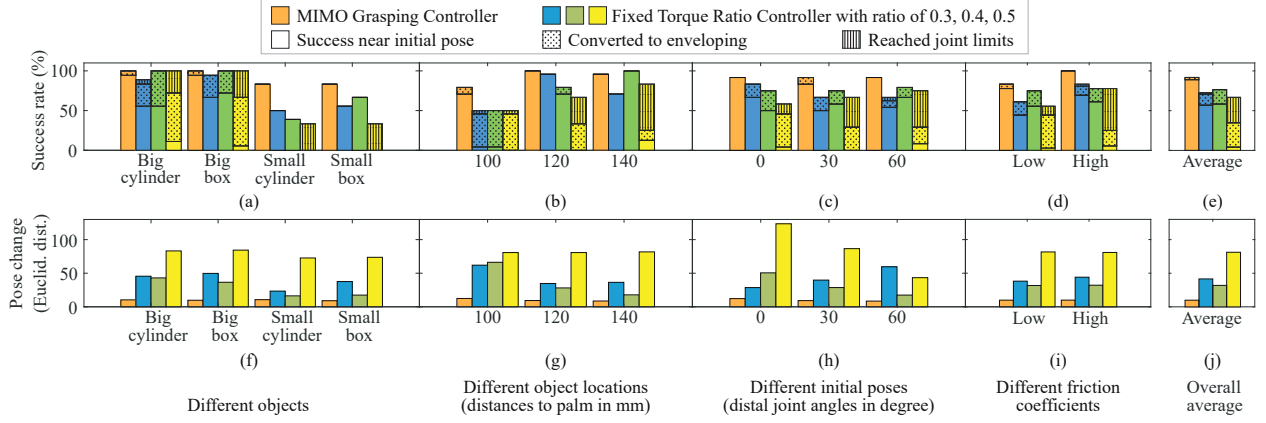


Figure 8.4: Experiment results of fingertip grasping, covering the proposed controller and the base-lines with Fixed Torque Ratio Controller, different objects, object locations, initial finger poses, and friction coefficients. First row: success rate, second row: pose change.

grasp but a joint limit was reached (d), the grasp was transformed into an enveloping one (e), and the object was squeezed out of the hand (f).

The results of the experiments comparing against emulated underactuated hand are visualized as multiple bar charts in Fig. 8.4. In each plot, the bar colors show four different controllers, the vertical axis is one of the performance metrics, and the horizontal axis represents another dimension which is different in each plot (from (a) (f) to (d) (i): different objects, object locations, initial poses, and friction coefficients). In each bar in the first row showing the success rate, the pure-color area, the dotted area, and the line-shaded area represent, respectively, successful fingertip grasp without reaching joint limits, successful grasps but converted to enveloping, as well as successful fingertip grasp but joint angles reached limits.

As shown in Fig. 8.4 (e)(j), the overall success rates are 91.67% for PI Network Grasping Controller, and 72.22%, 76.39%, 66.67% for emulated underactuated hand with torque ratio of 0.3, 0.4, 0.5, respectively. Even when the overall success rates are close, the types of the resulting grasps are significantly different. Besides, the hand pose change metric for the PI Network Grasping Controller is 9.96, compared to 41.55, 31.93, and 81.14 (degrees) respectively for the emulated underactuated hand.

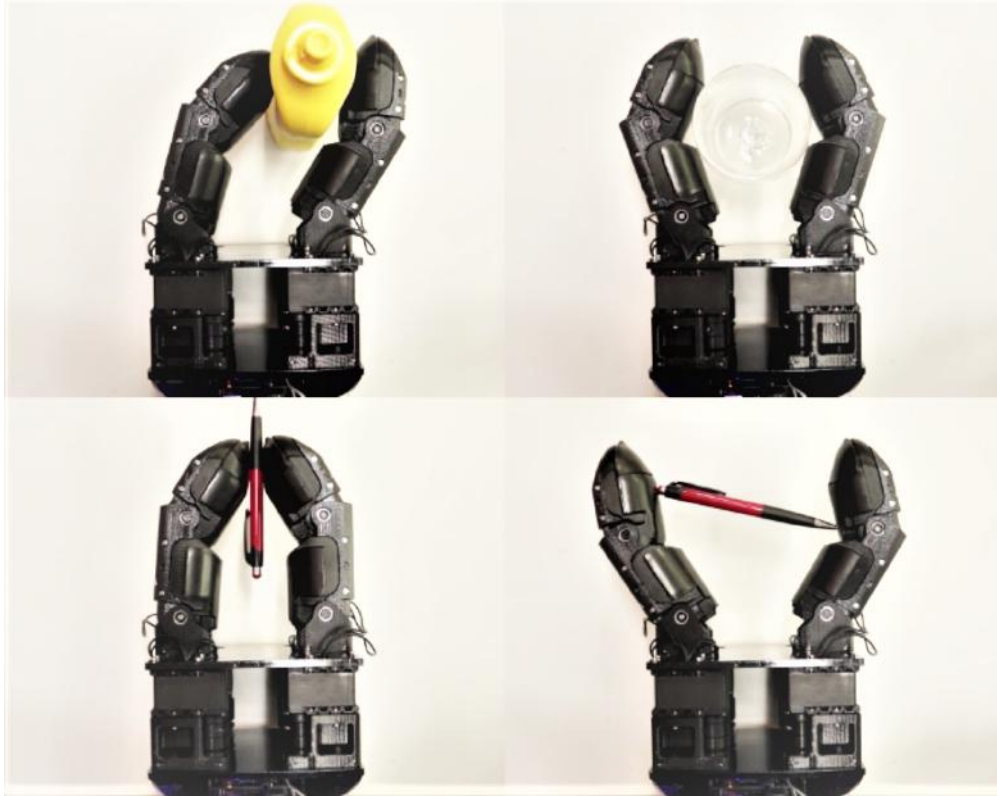


Figure 8.5: Grasps of different objects with random initial touch poses.

Enveloping Grasps

If the initial touch is an enveloping grasp (both proximal and distal links make contacts), the same grasping controller can be used to load such grasps. We performed experiments to show that this controller can perform enveloping grasps. We tested on two objects and two object locations. The success rate is 100%.

Fig. 8.5 shows some fingertip and enveloping grasps using PI Network Grasping Controller on some random objects with random initial touch poses.

In-Hand Manipulation

The last experiment is to show the performance of the in-hand manipulation using this controller. We note that this controller is not originally designed to perform in-hand manipulation — it has no sense of the exact object trajectory — but it can conduct some in-hand manipulation tasks

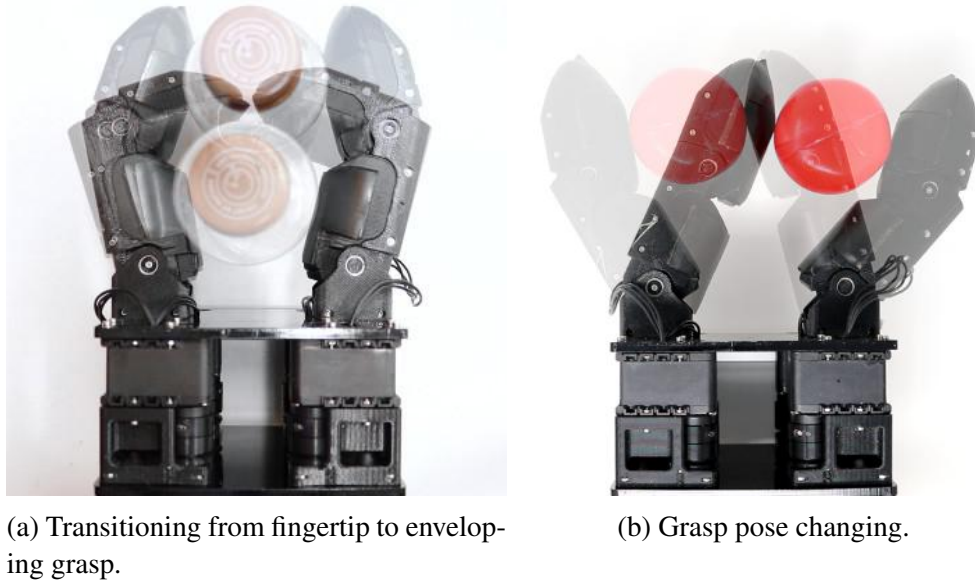


Figure 8.6: In-hand manipulation tasks.

such as transitioning grasp types, changing grasping poses, etc.

The first in-hand manipulation task is the transition from fingertip to enveloping grasp. We first created fingertip grasps using the PI Network Grasping Controller, and then change the goal pose of the controller to be an enveloping pose. We tested on two objects, three object locations, three initial poses with the low friction fingertips. We found the success rate is 83.33%. The most common failure mode is the hand still makes a fingertip grasp (no contact on the proximal links) after the move.

The second in-hand manipulation task is to change fingertip grasping poses. Similarly, we first make a fingertip grasp using the PI Network Grasping Controller, and then given it a different goal pose. We tested on two objects, three object locations, three initial poses. The success rate is 61.11%. The most common failure mode is that the hand loses contact with the object and thus throw it out of reach during the manipulation process. Even so, we note that the passive compliance of SEA is playing an important role to keep the grasp, because we only set the start and end points and have no control to keep the grasp along the trajectory.

8.3 Discussion

Overall, the results of the previous section support our hypotheses: the proprioceptive hand running PI Network Grasping Controller is effective at executing stable grasps in a variety of situations. Furthermore, the proprioceptive hand exhibits versatility in being able to perform different types of grasps and also simple in-hand manipulation tasks.

Based on Fig. 8.4, the PI Network Grasping Controller outperforms the baselines in fingertip grasping, and usually succeeds without transforming to an enveloping grasp or reaching joint limits. In contrast, the emulated underactuated hand often transform into an enveloping grasp or reach joint limits, thus relying on hand reconfiguration. The second row of Fig. 8.4 ((f) to (j)) gives similar intuition: for the Fixed Torque Ratio Controller, most cases have a large pose change. While the end-result is stable, it is different from the originally intended grasp. This might be unimportant or detrimental depending on the application.

It is also interesting to notice that, in different conditions, the optimal torque ratio for the emulated underactuated hand is different. We take this to mean that there is no one clearly preferable pre-set torque ratio, which could be physically implemented in a mechanical design, in order to obtain ideal performance in all these cases. In contrast, the proprioception-enabled hand has the flexibility to alter torques at run-time.

Looking at how specific factors affect performance we can gain additional insights. From Fig. 8.4 (a) and (b) we can see that the success rates for emulated underactuated hands are low if the objects are small and close to the palm. This is because the emulated underactuated hand tends to transform the initial fingertip grasps to enveloping when contacts are close to distal joints, but cannot cage the object if it is small because the distal links are fighting against each other — a common issue for underactuated hands. In contrast, the PI Network Grasping Controller does not suffer from this because it does not perform the conversion.

Another subtle but interesting point is that the controller network is not sensitive to parameter values. Due to the nature of the feedback architecture, sub-optimal hand-tuned parameters may

result in the desired behavior, and there is a large range of parameters that can provide similar performances. This is an example of an interesting point that a control policy with an appropriate architecture can work without finding the optimal parameters.

This work is also meaningful for studying grasping via Reinforcement Learning (RL). With an RL algorithm guiding the search for the initial grasping point, and a PI Network Grasping Controller loading the grasp, the search for good grasping policies may be much more efficient than pure stochastic approaches.

Chapter 9: The *ROAM* Hand: a Highly Sensorized and Fully-Actuated Hand Designed for Learning

After exploring the two-dimensional grasping with the Series-Elastic hand, we aimed to study force-based grasping in three-dimensional space, which requires a more dexterous hand with more fingers. Besides, instead of the hand-engineered controller shown in the previous chapter, we aim to tackle this problem by learning.

Taking the lessons learned from the SEA hand and compliant grasping, we designed another hardware platform: the *ROAM* Hand (shown in Fig. 9.5), which is a hardware designed for learning.

We believe good hardware platforms that are suitable for learning tasks are in great demand. Many existing hardware platforms do not meet the requirements for learning research in terms of sensing capability and robustness. For example, researchers from *OpenAI* complained that hardware breakage was one of the key challenges in their work [44].

The human body is an example of good hardware for learning: it is sensor-rich to receive a variety of signals in the environment, compliant to interact with the physical world safely, robust and failure-tolerant for long-term learning, and self-protected when there are potential hazards. Even though it is impossible for the robot hardware to reach human-level performance, we still take inspiration from the aforementioned points. We developed the *ROAM* Hand, which is a hardware platform specifically designed for learning motor skills.

In this chapter, we present the mechanical design of this hand. The use of this hand for learning purposes is the planned future work, which will be briefly discussed in the next chapter.

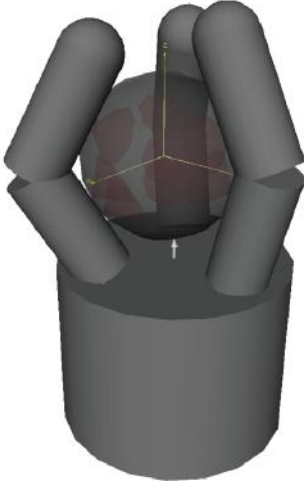


Figure 9.1: The *ROAM* Hand kinematics optimized using GraspIt! simulator.

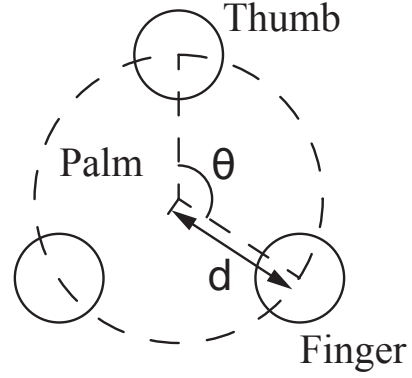


Figure 9.2: The parameterization of the *ROAM* Hand finger locations.

9.1 Kinematic Optimization

We used a large-scale grasp search infrastructure using GraspIt! simulator developed in our lab [120] to choose the best kinematic configuration of this hand. We searched over a variety of finger and palm dimensions in the outer loop, with a random-search-based grasp planning as the inner loop. We evaluated one grasp by the minimum Ferrari-Canny metric [116] of several disturbed grasps around the nominal configurations, and we evaluated the hand using the Ferrari-Canny of multiple objects.

Finally, we picked a design with three fingers, two links per finger, and 72mm in length and 38mm in diameter for each link. The thumb has two DoFs (proximal and distal) and the opposing fingers have three DoFs (the additional one is for rolling (abduction-adduction)). The finger relative locations on the palm are parameterized in Fig. 9.2, and we choose $\theta = 100^\circ$ and $d = 45mm$.

9.2 Sensing

The greatest feature of this hand is that it is highly sensorized: it is equipped with joint angle and torque sensors that provide proprioception; it also has tactile-sensed finger links.

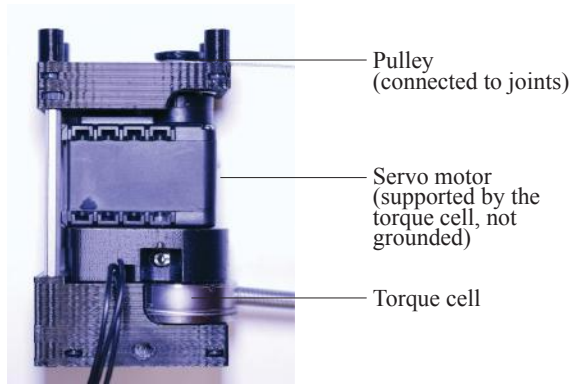


Figure 9.3: The *ROAM* Hand torque-sensed actuation unit.

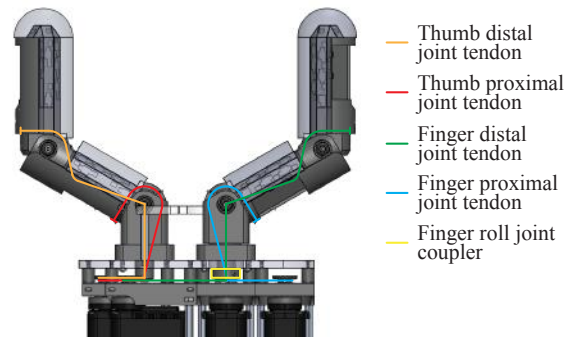


Figure 9.4: The tendon transmission of the *ROAM* Hand.

- *Proprioception.* In one actuation unit, the angle is measured by a built-in potentiometer in the servo motor, and the torque is sensed by a reaction torque cell. Unlike the SEA approach we took previously, we use the off-the-shelf torque sensors to get rid of the problem of hysteresis of SEA in this design. The motor is connected to the measuring-end of the grounded torque cell on one side and a bearing on the other side (the output), so the motor torque can be fully captured by the torque sensor.
- *Tactile sensing.* We use an optical-based approach developed in our lab [124] to sense touch. Each finger link is embedded in a layer of a transparent elastomer, with a reflective coating on top. The finger links are also equipped with a number of light emitters (LEDs) and receivers (photodiodes). The key principle is to have lights reflecting within the soft transparent layer on the finger, and detect and decode the change of light transmission when the material is deformed due to touch.

9.3 Actuation

There are eight joints in this hand: two flexing joints in the thumb, two flexing and one rolling (abduction-adduction) joints in each finger. We adopt a full actuation scheme, and use tendons to transmit torques from the motors in the palm to the joints. Without the SEA, this design matches the first paradigm we introduced in Section 7.2, where the compliance comes from the fingertips.

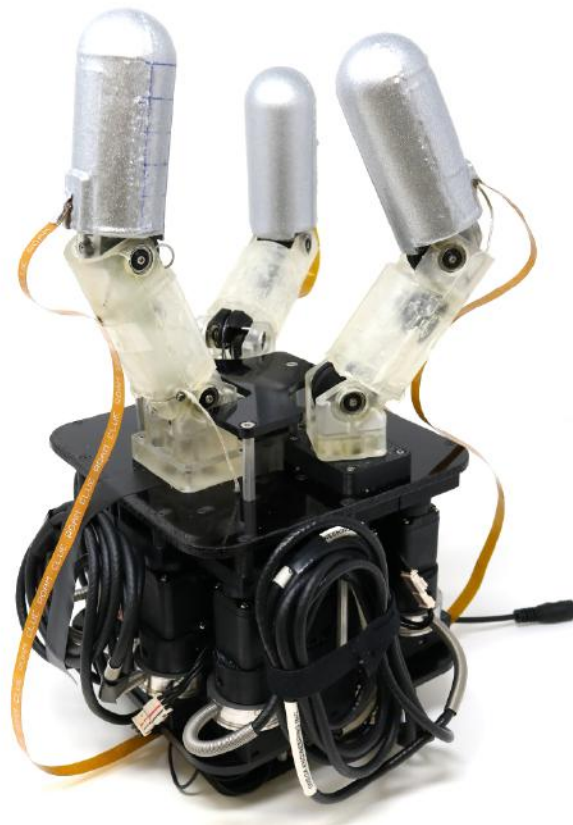


Figure 9.5: The *ROAM* Hand prototype.

Chapter 10: Compliant Force-Based Grasping: Conclusion and Future Work

10.1 Contributions

We summarize the contributions of this part here:

- To the best of our knowledge, *we are the first to show that, purely based on proprioceptive feedback on a compliant hand, a robot hand can perform the fingertip grasps, enveloping grasps, and some simple in-hand manipulation tasks* (e.g. transition from fingertip to enveloping grasp) *for unknown objects*, given that the object is inside the hand graspable range.
 - Specifically, we demonstrated a hand hardware with SEA which facilitates compliant grasping.
 - We also proposed a MIMO grasping control policy consisting of a PI controller network, which can leverage the embodied compliance in the hand.
- We presented the mechanical design of the *ROAM* Hand — a highly-sensorized and fully-actuated hand designed for learning-based tasks.

10.2 On-Going and Future Work

We believe the in-hand compliance and proprioception-based grasping are important paths to explore. In addition to the work presented before, there is an important on-going work to discuss — blind bin-picking using Reinforcement Learning with variable finger joint stiffnesses.

Blind Bin-Picking with Variable Finger Joint Stiffness using Reinforcement Learning

We aim to study blind grasping, where no vision information is provided and the robot needs to explore the object(s) by hand. The stiffness of finger joints plays an important role in the initial exploration and the grasping and manipulation after exploration:

- During exploration, the fingers should be as compliant as possible to act as a “sensor” — in this way, the hand can gather information of the object(s) and the environment without moving the object(s) in a premature fashion.
- After exploring the object(s) and making grasp decisions, the fingers should become stiffer to behave as a “mover” and impose its will on the object(s).

With the *ROAM* Hand, we can implement an impedance controller for each joint and actively change the stiffnesses.

We aim to focus on the task of blind bin-picking. The hand is expected to go down into a bin with a lot of different objects, explore the objects, and then isolate and pick (any) one out of the bin. This is an interesting application for the scenarios where the objects are transparent or flaring which causes troubles for vision systems. Besides, it is guaranteed that the hand can make contacts and get reward signals when it goes down, which can help the learning process.

We aim to learn the blind bin-picking behavior by deep Reinforcement Learning (RL). The observations are provided by the force-based sensing modalities in the hand (as a first step, we only use the proprioception, and we can try proprioception and tactile sensing together later). The actions are the setpoints of the palm position controller, as well as the neutral positions and stiffnesses of the finger joint impedance controllers. We aim to train the policy first inside MuJoCo [15] simulation (the environment is shown in Fig.10.1), and then transfer to the real robot with the help from Domain Randomization [18].



Figure 10.1: The MuJoCo simulation environment for the blind bin-picking task.

Part III

Mechanical-Computational Co-Optimization

Chapter 11: Hardware as Policy: Mechanical and Computational Co-Optimization using Deep Reinforcement Learning¹

In Chapter 3, we proposed a “mechanical dimensionality reduction” for hand underactuation design, based on prespecified grasp data. However, such grasp data are generated by human intuition. Therefore, grasp planning and control are disconnected from the mechanism design.

However, the grasping performance is dependent on both the underactuation schemes and the motor control decisions (e.g. where to put the hand, when to grasp, how to lift, etc.). Broadly speaking, for any robot, both the mechanisms and control contribute to the behavior and performance. We believe it is natural thinking to explore optimizing the mechanisms and control policies simultaneously (i.e. “co-optimization”). Ideally, the topology/morphology, transmissions, sensing, and control policies should be optimized together and for each other.

This is also inspired by nature: human “intelligence” resides in both the brain and the body, and our motor skills and body physical features co-evolve via natural selection. For example, it has been suggested that, as early hominids practiced throwing (fingertip or precision grasp) and clubbing (power grasp), hand morphology also changed accordingly, e.g. the thumb got longer to provide better opposition [126].

In robotics, the idea of jointly designing/optimizing mechanical and computational components has a long track record with remarkable advances. If the control policy and dynamics can both be modeled analytically, traditional optimization can be used. When such an approach is not feasible (e.g. due to complex policies or dynamics), evolutionary computation has been utilized instead. However, these methods still have difficulty learning sophisticated motor skills in com-

¹The work presented in this and the next two chapters is published as [125] in 2020 Conference on Robot Learning (CoRL).

plex environments (e.g. with partially observable states, dynamics with transient contacts), or are sample-inefficient.

We choose the approach of deep Reinforcement Learning (RL). Recent advances in Deep RL have shown great potentials for learning difficult motor skills despite having only partial information of complex, unstructured environments (e.g. [39, 43, 42, 44, 127]). We got aware that hardware mechanics is akin to control policies in the sense that both can be expressed as computational graphs (further explained in the next section). Therefore, we can leverage the capabilities of RL for learning the control policies and mechanism design concurrently.

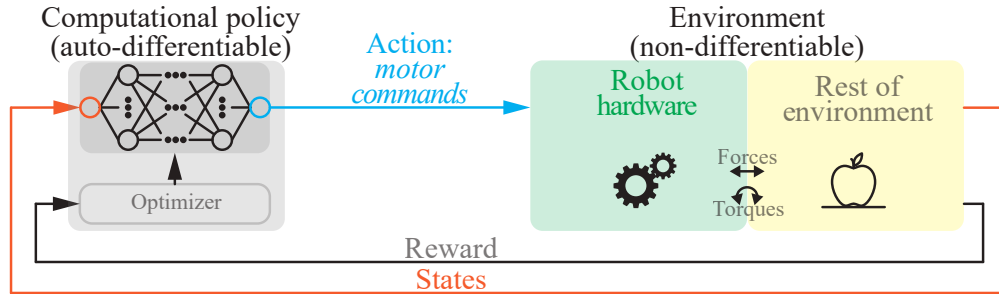
In this chapter, We first illustrate a different perspective of RL that enables the co-optimization, then explain it in detail. In the next two chapters, we will demonstrate the case studies and comparison with the baseline methods.

11.1 Traditional Perspective vs. Co-optimization Perspective of RL

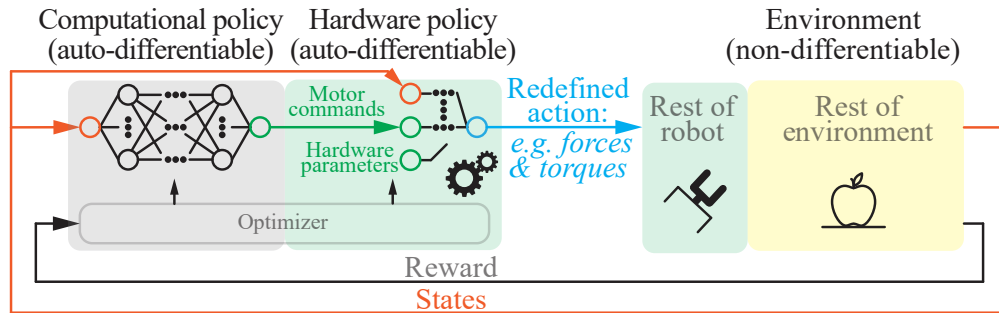
Traditionally, the output of an RL policy in robotics consists of motor commands, and the robot hardware converts these motor commands to effects on the external world (usually through forces and/or torques). In this conventional RL perspective, robot hardware is considered as given and immutable, essentially treated as part of the environment (Fig 11.1a).

In this work, we bring a different perspective for hardware in RL. Consider the concrete example of an underactuated robot hand. Motor angles and torques are converted into joint angles and torques by a transmission mechanism (gears, tendons or linkages). Through careful design of the hardware parameters, such a transmission can provide desired grasping behavior for a wide range of objects (e.g. [36, 128]). Such a transmission is conceptually akin to a policy, mapping an input (motor angles and torques) to an output (joint angles and torques) with carefully designed parameters leading to beneficial effects for overall performance.

Can we leverage the power of deep RL for co-optimization of the computational and mechanical components of a robot? High-fidelity physics simulation and effective sim-to-real transfer (where a policy is trained on a physics simulator and only then deployed on real hardware [18])



(a) Traditional perspective — Reinforcement Learning with a purely computational policy



(b) Hardware as Policy — computational graph implementation

Figure 11.1: Hardware as Policy overview. From the traditional perspective (a), all robot hardware is part of the simulated environment. In the proposed method (b), aspects of robot hardware are formulated as a “hardware policy” implemented as a computational graph, then optimized jointly with the computational policy.

provide such an opportunity, since it allows modifications of design parameters during training without incurring the prohibitive cost of re-building hardware. A straightforward option to optimize hardware parameters during training is to treat them as hyperparameters of the RL algorithm. However, this approach usually carries a massive computational cost.

In this study, we propose to consider *hardware as policy*, optimized jointly with the traditional computational policy. As is well known, a model-free Policy Optimization (e.g. [129, 130, 131]) or Actor-critic (e.g. [132]) algorithm can train using an auto-differentiable agent/policy and a non-differentiable black-box environment. The core idea we propose is to move part of the robot hardware from the non-differentiable environment into the auto-differentiable agent/policy (Fig 11.1b). In this way, hardware parameters² become parameters in the policy graph, analogous to the neural network weights and biases. Therefore, the optimization of hardware parameters can be directly incorporated into the existing RL framework, and can use existing learning algorithms with changes to the computational graphs as we will describe later.

11.2 Preliminaries and Nomenclature

We start from a standard RL formulation, where the problem of optimizing an agent to perform a certain task can be modeled as a Markov Decision Process (MDP), represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R})$, where \mathcal{S} is state space, \mathcal{A} is the action space, $\mathcal{R}(s, a)$ is the reward function, and $\mathcal{F}(s'|s, a)$ is the state transition model ($s, s' \in \mathcal{S}$, s' is for the next time step, $a \in \mathcal{A}$). Behavior is determined by a computational control policy $\pi_{\theta}^{comp}(a|s)$, where θ represents the parameters of the policy. Usually, π_{θ}^{comp} is represented as a deep neural network, with θ consisting of the network’s weights and biases. The goal of learning is to find the values of the policy parameters that maximize the expected return $\mathbb{E}[\sum_{t=0}^T \mathcal{R}(s_t, a_t)]$ where T is the length of episode. We note that here we use stochastic formulations of the policy and transition function, but our idea presented in this work can work with either stochastic or deterministic formulations.

²This work primarily focuses on the mechanical aspect of hardware, we use the terms “hardware” and “mechanics/mechanical” interchangeably. However, we believe that, in the future, the proposed idea could be extended to electrical or sensorial aspects of a physical device.

We start from the observation that, in robotics, in addition to the parameters θ of the computational policy, the design parameters of the hardware itself, denoted here by ϕ , play an equally important role for task outcomes. In particular, hardware parameters ϕ help determine the output (the effect on the outside world) that is produced by a given input to the hardware (motor commands). This is analogous to how computational parameters θ help determine the output of the computational policy (action \mathbf{a}) that is produced by a given input (state or observation \mathbf{s}).

Even though this analogy exists, traditionally, these two classes of parameters have been treated very differently in RL. Computational parameters can be optimized via gradient-based methods: for example, in Policy Optimization methods such as the Vanilla Policy Gradient (VPG) [129], Trust Region Policy Optimization (TRPO) [130] and Proximal Policy Optimization (PPO) [131], the parameters of the computational policy are optimized by computing and following the policy gradient:

$$\mathbf{g} = \mathbb{E}_{\tau \sim \pi_{\theta}^{comp}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}^{comp}(\mathbf{a}_t | \mathbf{s}_t) A_t(\mathbf{s}_t, \mathbf{a}_t) \right], \quad (11.1)$$

where A_t is the advantage function. Details on how to compute this gradient are beyond the scope of our work; we refer the interested reader to other studies [129, 130, 131, 133] for details.

In contrast, hardware is generally considered immutable, and modeled as part of the environment. Formally, this means that hardware parameters ϕ are considered as parameters of the transition function $\mathcal{F} = \mathcal{F}_{\phi}(\mathbf{s}' | \mathbf{s}, \mathbf{a})$ instead of the policy. This is the concept illustrated in Fig. 11.1a. Such a formulation is grounded in the most general RL framework, where \mathcal{F} is not modeled analytically, but only observed by execution on real hardware. In such a case, changing ϕ can only be done by building a new prototype, which is generally impractical.

However, in recent years, the robotics community has made great advances in training via a computational model of the transition function \mathcal{F} , often referred to as a physics simulator (e.g. [15]). The main drivers have been the need to train using many more samples than possible with real hardware, and ensure safety during training. Recent results have indeed shown that it is often possible to train exclusively using an imperfect analytical model of \mathcal{F} , and then transfer to the real

world [18].

In our context, training with such physics simulators opens new possibilities for hardware design: we can change the hardware parameters ϕ and test different hardware configurations on-the-fly inside the simulator, without incurring the cost of re-building a prototype.

11.3 Hardware as Policy

The Hardware as Policy method (HWasP) proposed here largely aims to perform a similar optimization for hardware parameters as we do for computational policy parameters, i.e. by computing and following the gradient of action w.r.t such parameters.

The core of the HWasP method is to model the effects of the robot hardware we aim to optimize separately from the rest of the environment. We refer to this component as a “hardware policy”, and denote it via $\pi_{\phi}^{hw}(\mathbf{a}^{new}|s, \mathbf{a})$. The input to the hardware policy consists of the action produced by the computational policy (i.e. a motor command) and other components of the state; the output is in a redefined action space \mathcal{A}^{new} further discussed below.

In the traditional formulation outlined so far, the “hardware policy” and its parameters ϕ are included in the transition function \mathcal{F}_{ϕ} . With HWasP, π_{ϕ}^{hw} becomes part of the agent. The new overall policy:

$$\pi_{\theta, \phi} = \pi_{\phi}^{hw}(\mathbf{a}^{new}|s, \mathbf{a})\pi_{\theta}^{comp}(\mathbf{a}|s) \quad (11.2)$$

comprises the composition of both computational and hardware policies (here it is the product of them since both are independent probabilities). The new transition probability $\mathcal{F}^{new}(s'|s, \mathbf{a}^{new})$ encapsulates the rest of the environment. In other words, we have split the simulation of the environment: one part consists of the hardware policy, now considered part of the agent, while the other simulates the rest of the robot, and the external environment. The reward function, $\mathcal{R}(s, \mathbf{a})$, is redefined to be associated with the new action space: $\mathcal{R}^{new}(s, \mathbf{a}^{new})$. Once this modification is performed, we run the original Policy Optimization algorithm on the new tuple $(\mathcal{S}, \mathcal{A}^{new}, \mathcal{F}^{new}, \mathcal{R}^{new})$

as redefined above. However, in order for this to be feasible, two conditions have to be met:

Condition 1: The redefined action vector \mathbf{a}^{new} must encapsulate the interactions between the hardware policy and the rest of the environment. In other words, this new action interface must comprise all the ways in which the hardware we are optimizing effects change on the rest of the environment. Furthermore, the redefined action vector must be low-dimensional enough to allow for efficient optimization. Such an interface is problem-specific. Forces/torques between the robot and the environment make good candidates, as we will exemplify in the following chapters.

Condition 2: To use Policy Optimization algorithms, we need to efficiently compute the gradient of the redefined action probability w.r.t. hardware parameters. We further discuss this condition next.

Computational Graph Implementation (HWasP). In order to meet Condition 2 above, *we propose to simulate the part of hardware we care to optimize as an auto-differentiable computational graph*. In this way, gradients can be computed by auto-differentiation and can flow or back-propagate through the hardware policy. Similar to the computational policy, the gradient of log-likelihood of actions w.r.t mechanical parameters $\phi — \nabla_{\phi} \log \pi_{\phi}^{hw}(\mathbf{a}^{new}|\mathbf{a}, s) —$ can be computed in a straightforward way.

Critically, since the computational policy is also expressed as a computational graph, when we connect hardware and computational policies, the gradient can back-propagate through both, and the hardware and computational parameters are jointly optimized. This general idea is illustrated in Fig. 11.1b.

However, this approach is predicated on being able to simulate the effects of the hardware being optimized as a computational graph. Once again, the exact form of this simulation is problem-specific, and can be considered as a key part of the algorithm. In the following chapters, we illustrate how this can be done both for a toy problem, and for a real-world design problem, and regard these implementations as an intrinsic part of the contribution of this work.

A subtle side note is that the interface between the computational policy and hardware pol-

icy needs to allow back-propagation of gradients. If there is a sampling procedure in the interface (i.e. the original computational policy is stochastic), or if the outputs of the original computational policy are discrete values, the “re-parameterization tricks” (for example, the Gaussian re-parameterization used in Variational Autoencoders [134], and the Gumbel-Softmax re-parameterization [135]) can be used.

Minimal Implementation (HWasP-Minimal). In the general case, where should the split between the (differentiable) hardware policy and the (non-differentiable) rest of the environment simulation be performed? In particular, what if the hardware we care to optimize does not lend itself to a differentiable simulation using existing methods?

Even in such a case, we argue that a “minimal” hardware policy is always possible: we can simply put the hardware parameters into the output layer of the original computational policy. In this case, $\mathbf{a}^{new} = [\mathbf{a}, \boldsymbol{\phi}]^T$. Here, the policy gradient with respect to the hardware parameters is trivial but can be still useful to guide the update of parameters. When this case is implemented in practice, the transition function $\mathcal{F}(s'|s, \mathbf{a}^{new})$ typically operates in two steps: first, it sets the new values of the hardware parameters to the underlying simulator, then advances the simulation to the next step.

We illustrate HWasP-Minimal in Fig. 11.2, which can be compared to HWasP as illustrated in Fig. 11.1b. HWasP-Minimal is simple to implement since it does not require a physics-based auto-differentiable hardware policy. As shown in our results in the next chapters, we found that HWasP-Minimal performs at least as well as or better than our baselines, but still below HWasP.

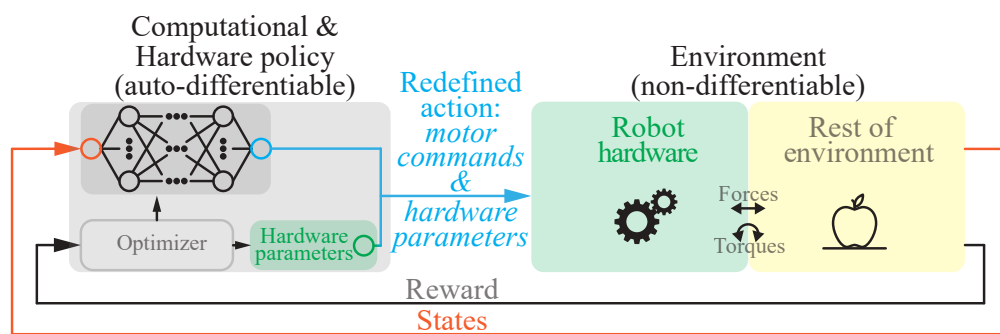


Figure 11.2: Hardware as Policy-Minimal Implementation.

Chapter 12: Mass-Spring Toy Problems

In this chapter, we illustrate the “Hardware as Policy”(HWasP) method in two toy problems of mass-spring systems. We also demonstrate the performance comparison with the baseline methods we compete against.

12.1 Comparison Baselines

We compare HWasP and HWasP-Minimal with the following baselines:

- Hyperparameter search:
 - *CMA-ES with RL inner loop*. We treat hardware parameters as hyperparameters, optimized in an outer loop using an evolutionary algorithm while the computational policy is learned (by RL algorithms such as PPO or TRPO) in an inner loop, for each set of hardware parameters. In this baseline we use Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [49].
 - *ARS with RL inner loop*. This baseline is very similar to the above except that the evolutionary algorithm is switched to Augmented Random Search (ARS) [136]. While ARS was originally tested on linear models, it can also be applied to non-linear systems (like our computational and hardware policies) where it still represents a useful baseline.
- Evolutionary computation:
 - *Pure CMA-ES*. We use CMA-ES as gradient-free evolutionary strategies to directly learn both computational policy and hardware parameters, without a separate inner loop.
 - *Pure ARS*. Similar to above, but the optimizer is switched to ARS.

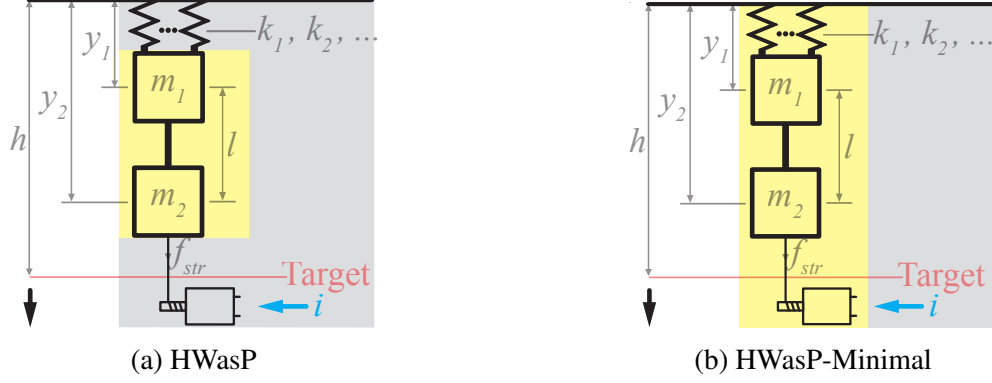


Figure 12.1: The mass-spring system — optimizing spring stiffnesses and computational policy (yellow: environment, gray: agent/policy).

12.2 Co-Optimization of Spring Stiffnesses and Computational Policy

We present a one-dimensional example of the mass-spring system in Fig. 12.1. Two point masses, connected by a massless bar, are hanging under n parallel springs with stiffnesses k_1, \dots, k_n . A motor can pull the lower mass by a string. The behavior is governed by a computational policy regulating the motor current, and by the hardware parameters (spring stiffnesses). We note that only the sum of spring stiffnesses matters since they are in parallel, but we still consider each stiffness as an individual parameter to test how our methods scale up for higher-dimensional problems.

Problem Formulation. This toy problem is formalized as follows:

- The *observations* we can measure are the mass position y_1 and velocity \dot{y}_1 .
- The *input* to the hardware policy is the motor current i , which is the result of the computational policy.
- The *variables* to optimize are the weights and biases in the computational policy neural network, as well as all the spring stiffnesses k_1, k_2, \dots, k_n .
- The *goal* is to make the mass m_2 go to the red target line in Fig 12.1 and stay there, with the minimum input effort. We designed a two-stage reward function that rewards smaller position and velocity error when the mass m_2 is far from the goal or moving fast, and in

addition rewards less input current when the mass is close to the goal and almost still.

$$R = \begin{cases} -\alpha|y_2 - h| - \beta|\dot{y}_2| - \gamma|i_{max}|, & \text{if } \alpha|y_2 - h| + \beta|\dot{y}_2| > \epsilon \\ -\alpha|y_2 - h| - \beta|\dot{y}_2| - \gamma|i|, & \text{if } \alpha|y_2 - h| + \beta|\dot{y}_2| < \epsilon \end{cases} \quad (12.1)$$

where the α , β , and γ are the weighting coefficients, i_{max} is the upper bound of the motor current, and ϵ is a hand-tuned threshold.

Hardware as Policy. In this case, we include the effect of the parallel springs in the hardware policy.

We use the Hooke’s Law

$$f_{spr} = \sum_{i=1}^n k_i y_1 \quad (12.2)$$

and current-torque relationship for the motor (ignoring rotor inertia and friction)

$$f_{str} = \frac{k_T i}{r_{shaft}} \quad (12.3)$$

to model the mechanical part of our agent, where k_T is the torque constant of the motor and r_{shaft} is the shaft radius. We implement the computational graph of this hardware policy and combine it with a computational policy expressed as a neural network computational, as shown in Fig 12.2a. The redefined action \mathbf{a}^{new} consists of the total resultant force $f_{total} = f_{str} - f_{spr}$. The transition function \mathcal{F} (rest of the environment) implements Newton’s Law for the two masses, assuming f_{total} as an external force.

Hardware as Policy — Minimal. In this method, we simply re-define the action vector to also include spring stiffnesses: $\mathbf{a}^{new} = [i, k_1, \dots, k_n]^T$. The transition function \mathcal{F} is responsible for modeling the dynamics of the springs and the two masses.

Implementation Details. We implemented HWasP, HWasP-Minimal, as well as our four baselines: CMA-ES with RL inner loop, ARS with RL inner loop, CMA-ES, and ARS. In order to have a fair comparison between them, we intentionally made different cases share common as-

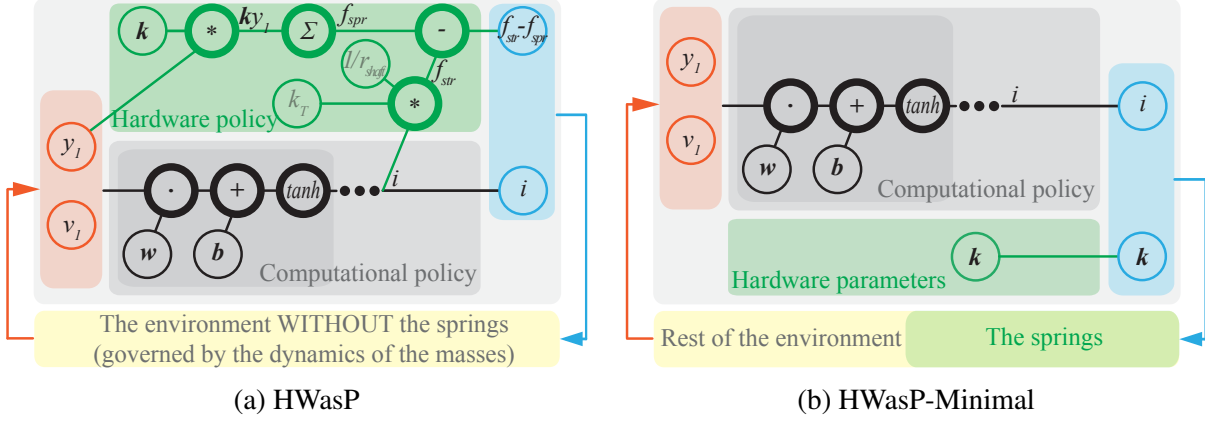


Figure 12.2: The (simplified) computational graphs for the toy problem — optimizing spring stiffnesses and computational policy. Bold circles represent tensor operations, light black or green circles with black fonts represent the variables to be optimized, and light circles with translucent fonts represent constants that are not part of the optimization.

pects wherever possible. The physics parameters not being optimized are the same for all cases: $m_1 = m_2 = 0.1\text{kg}$, $h = 0.2\text{m}$, $g = 9.8\text{m/s}^2$, $k_T = 0.001\text{Nm/A}$, $r_{shaft} = 0.001\text{m}$. The initial conditions are random within the feasible range. We used PPO and CMA-ES in the Garage package [137] for all cases and we adapted the ARS implementation from [136]. We implemented the computational graphs for HWasP in TensorFlow and the dynamics of the rest of the environment (non-differentiable) by ourselves using mid-point Euler integration. In the computational policies, the neural network sizes are set to be 2 layers and 32 nodes in each layer. The episode length is 1,000 environment steps and the total number of steps is 4×10^6 .

Numerical Results. If we ignore the transition phase of the task in this toy problem and make a quasi-static assumption, and assuming total spring stiffness equals the following:

$$k^* = \frac{(m_1 + m_2)g}{h-l} \quad (12.4)$$

then gravity will drag the mass m_2 exactly to the target, and the steady-state input current i can be zero, which minimizes the return on i in a long enough horizon. In the real world, the system has dynamic effects, but optimized total stiffness k should still be close to this value given a long enough episode. After training, we indeed find the optimized total stiffness close to k^* , as shown

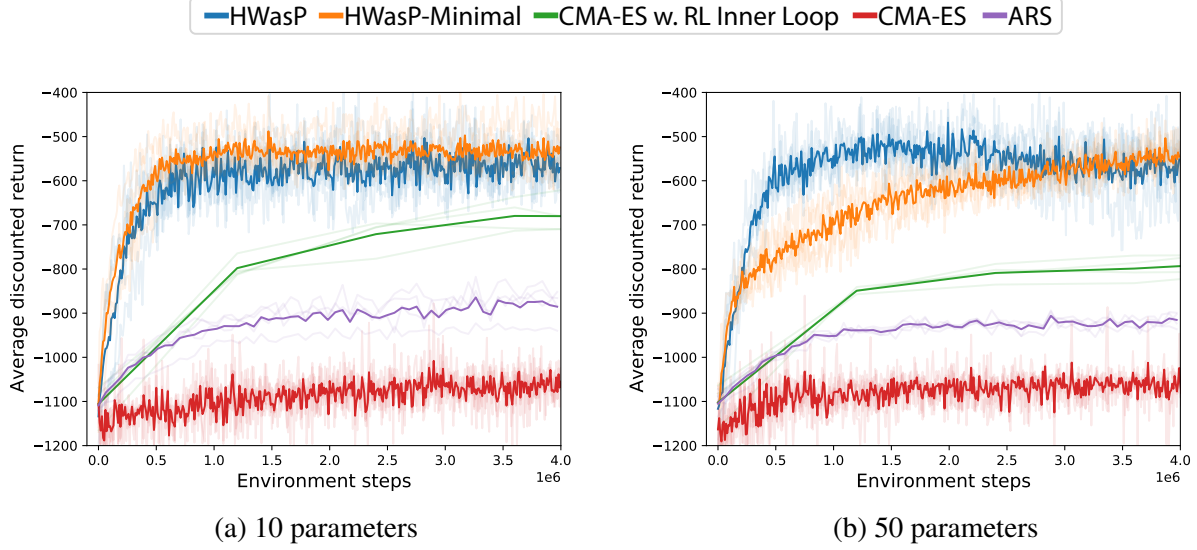


Figure 12.3: The training curves of the toy problem — optimizing spring stiffnesses and computational policy.

in Table 12.1.

Table 12.1: Optimization results of total stiffness [N/m]

| | $n = 10$ | $n = 50$ |
|---------------|----------|----------|
| k^* | 19.6 | |
| HWasP | 19.9 | 20.1 |
| HWasP-Minimal | 18.7 | 21.9 |

Training Curves. Fig. 12.3 shows the comparison of the training curves for both implementations of our method, as well as other baselines, for two cases: 10 and 50 parallel springs. In both cases, HWasP learns an effective joint policy that moves the lower mass to the target position. HWasP-Minimal works equally well for the smaller problem, but suffers a drop in performance as the number of hardware parameters increases. CMA-ES with RL inner loop also learns a joint policy to some extent, but learns slower than our method, especially for the larger problem. CMA-ES by itself does not exhibit any learning behavior over the number of samples tested. ARS achieves better performance than CMA-ES but is still worse than HWasP, HWasP-Minimal and CMA-ES with RL inner loop.

12.3 Co-Optimization of Bar Lengths and Computational Policy

We also performed co-optimization of the bar length between the two masses and the control policy to demonstrate our method can optimize geometric parameters. Unlike the previous case, the interface between the redefined policy and environment is no longer an element generating forces, but a “hard” geometric relationship. For HWasP, we propose to use springs and dampers to “soften” such interfaces, and still use the spring-damper forces as the redefined action. For HWasP-Minimal, the bar length can directly be used as a part of the action vector. Similar to the previous case, we subdivide the bar into multiple segments and treat the length of each segment as an individual parameter to test the scalability to higher-dimensional problems. The spring-mass system is illustrated in Fig.12.4.

Since we assume the bar is weightless, the dynamics is infinitely fast. Assuming all interfaces have the same k_{interf} and b_{interf} , we have:

$$\begin{aligned} f_{interf} &= k_{interf} \left(\left(\frac{1}{2}(y_1 + y_2) - \frac{1}{2} \sum_{i=1}^n l_i \right) - y_1 \right) + b_{interf} \left(\frac{1}{2}(\dot{y}_1 + \dot{y}_2) - \dot{y}_1 \right) \\ &= \frac{1}{2} k_{interf} (y_2 - y_1 - \sum_{i=1}^n l_i) + \frac{1}{2} b_{interf} (\dot{y}_2 - \dot{y}_1), \end{aligned} \quad (12.5)$$

$$f'_{interf} = -f_{interf}. \quad (12.6)$$

We implement this relationship as a computational graph, as shown in Fig.12.5a.

The details of the environment and training process are similar to the previous toy problem. The training curves are shown in Fig.12.6. We can see that the performance of both our methods and the baselines are very similar to the previous toy problem.

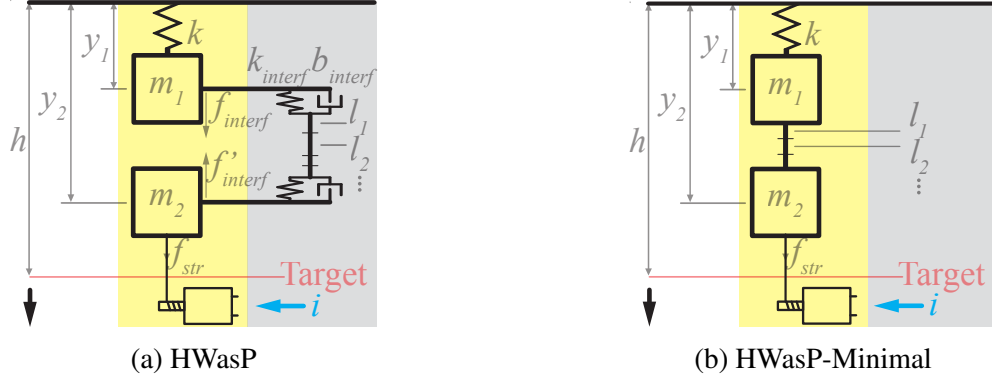


Figure 12.4: The mass-spring system — optimizing the bar lengths and computational policy (yellow: environment, gray: agent/policy).

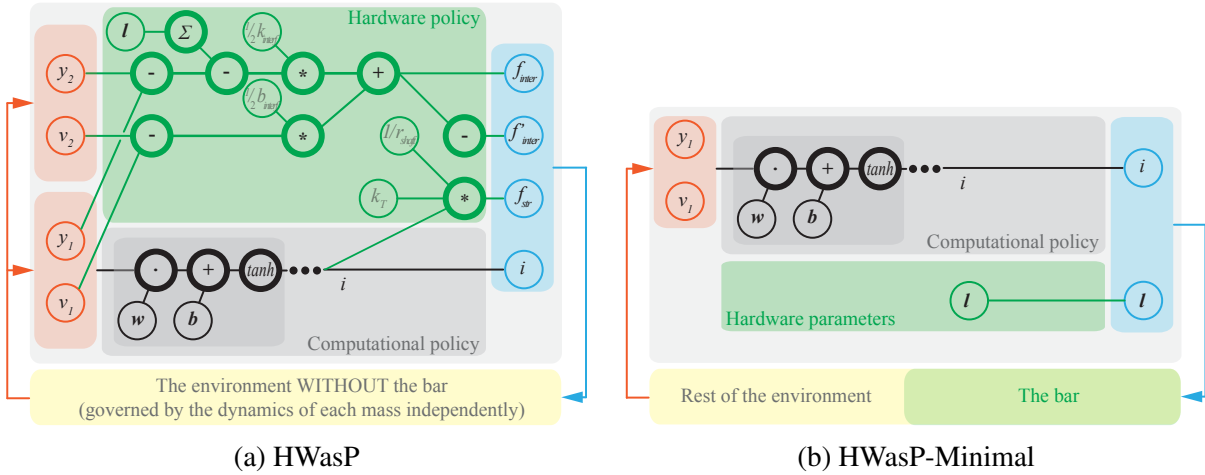


Figure 12.5: The (simplified) computational graphs for the mass-spring toy problem — optimizing bar lengths and computational policy.

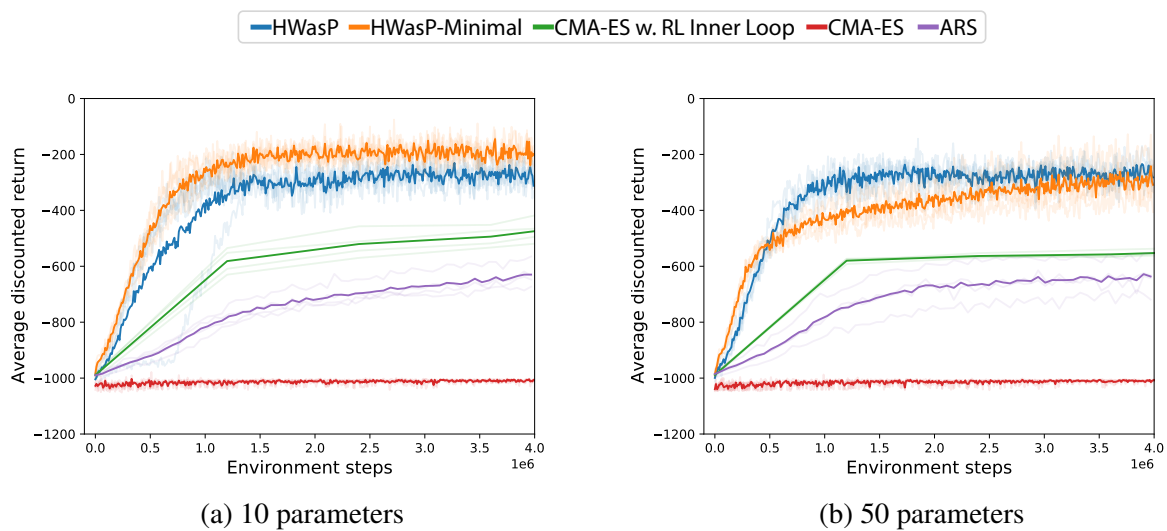


Figure 12.6: The training curves of the toy problem — optimizing bar lengths and computational policy.

Chapter 13: Mechanical-Computational Co-Design of an Underactuated Hand

The “Hardware as Policy” (HWasP) is a generally applicable method for the mechanical-computational co-design of any intelligence system. In this chapter, we give a concrete example of a real-world problem: optimizing the mechanism and the control policy of an underactuated robot hand.

The high-level design goal, inspired by our previous work [109], is to design a robot hand that is compact, but still versatile. To achieve the stated compactness goal, all joints are driven by a single motor, via an underactuated transmission mechanism: one motor actuates all joints by tendons in the flexion direction (see Fig 13.1). Finger extension is passive, via preloaded torsional springs. The mechanical parameters that govern the behavior of this mechanism consist of tendon pulley radii in each joint, as well as stiffness values and preload angles for restoring springs.

However, different from the previous work, here we look to simultaneously optimize the hardware parameters along with a computational policy that determines how to position the hand and use the hand motor. The input to the computational policy consists of palm and object positions, object size, and current motor travel and torque. Its output contains a relative position setpoint for hand motor travel as well as palm position commands.

From a hardware perspective, we aim to optimize all the underactuated transmission parameters listed above. We note that, in this work, we do not try to optimize the kinematic structure or topology for the hand. Unlike the underactuated transmission, these aspects do not lend themselves to parameterization and implementation as computational graphs, preventing the use of the HWasP method in its current form. While HWasP-Minimal could still be applied, we leave that for future investigations.

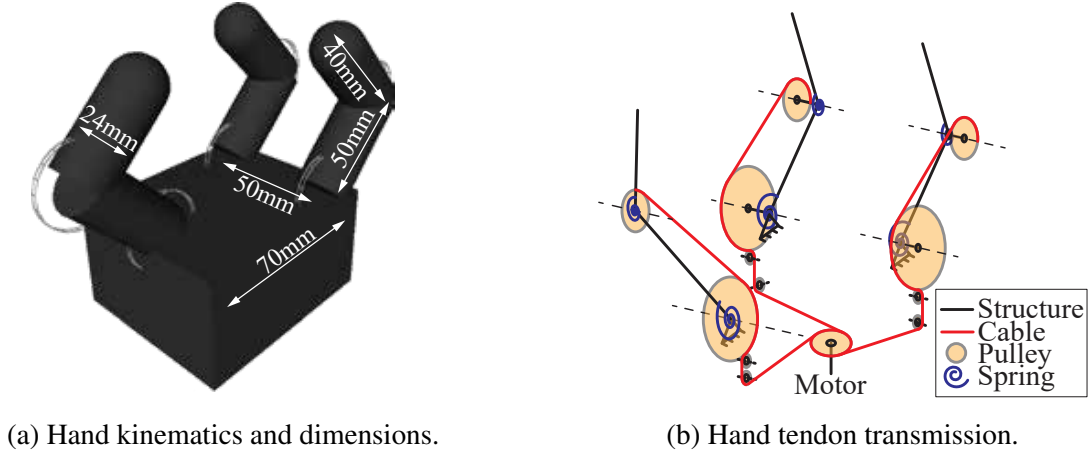


Figure 13.1: The Underactuated hand co-design optimization problem.

13.1 Tendon Underactuation Model

In a tendon-driven underactuated hand, the tendon mechanism acts as the transmission that converts motor forces to joint torques, based on the overall state of the system. In the model we constructed, it assumes an elastic tendon (with stiffness k_{tend}) goes through multiple revolute joints by wrapping around circular pulleys (radii r_{pul}), and each joint is closed by the tendon and opened by a restoring spring (stiffness k_{spr} and preload angle θ_{spr}^{pre}). In order to make this problem determinate, our model thus assumes a nominal (finite) tendon stiffness, takes in the commanded relative motor travel Δx_{mot} , the motor position reading x_{mot} and the joint angles θ_{joint} in the previous time step, and computes the joint torques τ_{joint} for the current time step. Such torques are then commanded to the joints in the physics simulation.

The tendon elongation can be calculated as:

$$\delta_{tend} = (x_{mot} + \Delta x_{mot}) + \mathbf{r}_{pul}^T \boldsymbol{\theta}_{joint}^{ref} - \mathbf{r}_{pul}^T \boldsymbol{\theta}_{joint} \quad (13.1)$$

where $\boldsymbol{\theta}_{joint}^{ref}$ is the joint angle when the motor is in zero-position, and usually we define it to be zero. Then the tendon force can be calculated as:

$$\mathbf{f}_{tend} = k_{tend} \delta_{tend}. \quad (13.2)$$

Hence, the torques applied to joints are:

$$\boldsymbol{\tau}_{joint} = f_{tend} \mathbf{r}_{pul} - \mathbf{k}_{spr} * (\boldsymbol{\theta}_{joint} + \boldsymbol{\theta}_{spr}^{pre}) \quad (13.3)$$

where $*$ means element-wise multiplication.

This model is built as an auto-differentiable computational graph in HWasP, and a non-differentiable model on top of the physics simulation in HWasP-Minimal and the baselines.

13.2 Problem Formulation

- The *observations* are the position vector of the palm \mathbf{p}_{palm} , the position vector of the object \mathbf{p}_{obj} , the size vector of the object bounding-box \mathbf{l}_{obj} , and the current hand motor travel x_{mot} and torque τ_{mot} . We can also send joint angles $\boldsymbol{\theta}_{joint}$ to the hardware policy (only used in training time), but not the computational policy, because there are no joint encoders in such an underactuated hand, and the computational policy (which will serve as a controller of the real robot in run time) does not have access to joint angles.
- The *input* to the hardware policy also includes the relative motor travel command Δx_{mot} produced by the computational policy. The output of the overall system includes joint torques as well as palm motion $\Delta \mathbf{p}_{palm}$. As mentioned above, we tested both “Z-Grasp” (top-down grasping with only z-axis motion) where $\Delta \mathbf{p}_{palm}$ is one-dimensional and “3D-Grasp” (top-down grasping with x,y,z motions) where $\Delta \mathbf{p}_{palm}$ is three-dimensional.
- The *variables* to optimize are the parameters in the computational policy neural network, and the hand underactuation parameters: pulley radii \mathbf{r}_{pul} , the joint restoring spring stiffnesses \mathbf{k}_{spr} and the joint restoring spring preload angles $\boldsymbol{\theta}_{spr}^{pre}$, where each vector has a dimension of four corresponding to the proximal and distal joints in the thumb and the opposing fingers (the two fingers share the same parameters).

- The *goal* is to grasp the object and lift it up. Formally, the reward function is:

$$R = \alpha \| \mathbf{p}_{palm} - \mathbf{p}_{obj} \| + \beta C + f(z_{obj}) \quad (13.4)$$

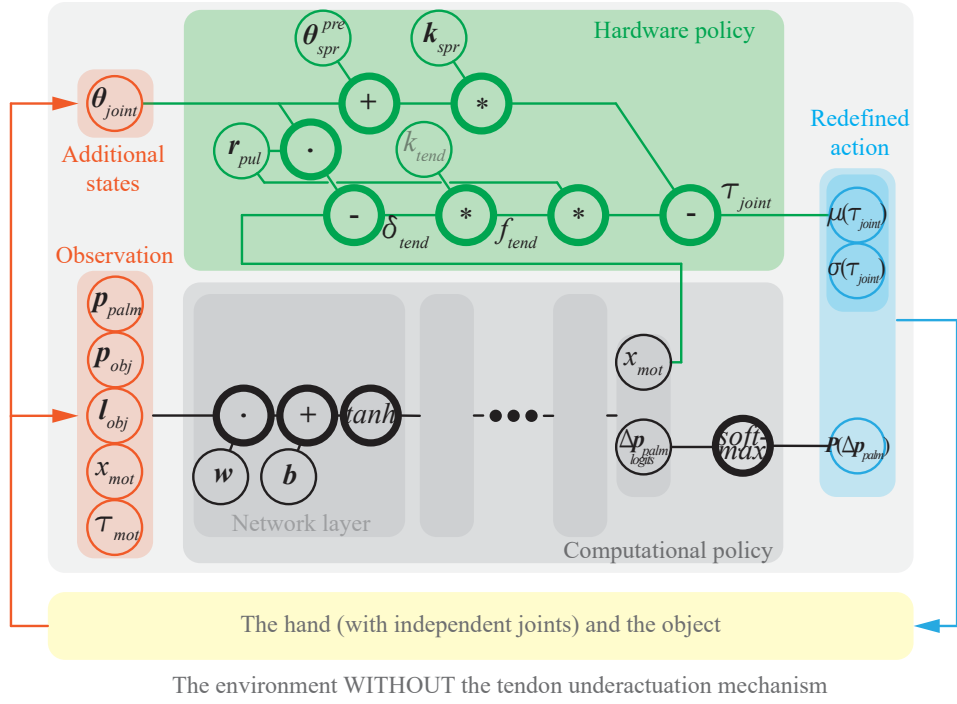
where the α , and β are the weighting coefficients, \mathbf{p}_{palm} and \mathbf{p}_{obj} are the positions of the palm and the object, C is the number of contacts between the distal links and the object, z_{obj} is the height of the object, and $f(z_{obj})$ is a hand-tuned non-decreasing piecewise-constant function of z_{obj} .

13.3 Implementation Details

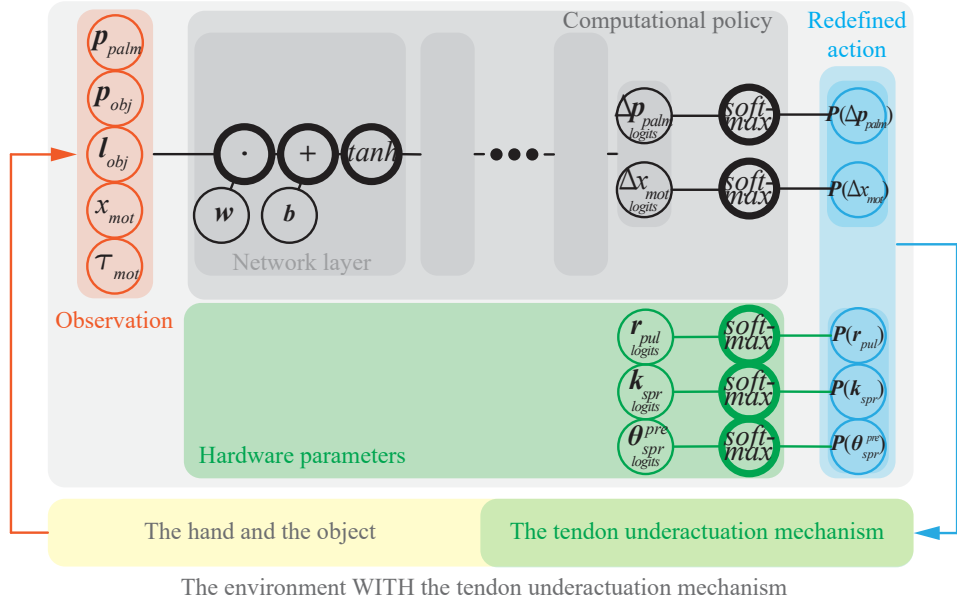
Similar to the toy problem, we used the Garage package [137] and TensorFlow for HWasP, HWasP-Minimal and the baselines in this design case. We use MuJoCo [15] for the physics simulation of the hand. The simulation time step is $0.001s$, the environment step is $0.01s$, and there are 500 environment steps per episode and 2×10^7 steps for the entire training. The computational policy is a fully-connected neural network with 2 layers and 128 hidden units on each layer.

Since hardware parameters can be large in scale compared to weights and biases in the neural network, a small change can lead to a large shift of the joint policy output distribution during training, which in turn can result in local optimum in the reward landscape. To alleviate this issue, We use TRPO [130] as the underlying RL algorithm as it allows for hard constraints on action distribution changes. In addition, we added scaling factors to mechanical parameters in different scales. For example, we scale all tendon stiffness and k_{spr} by 1.0×10^{-3} . Then, we multiply the torques by 1.0×10^3 when we apply them to the environment.

We also apply Domain Randomization [18] in the training to increase the chance of successful sim-to-real transfer. We randomized object shape, size, weight, friction coefficient and inertia, injected sensor and actuation noise, and applied random disturbance wrenches on the hand-object system.



(a) HWasP



(b) HWasP-Minimal

Figure 13.2: The (simplified) computational graphs for the hand mechanical-computational co-design problem. The illustration conventions are the same as the toy problem.

Table 13.1: The optimized pulley radii, joint spring stiffnesses and preload angles. In each cell, the first number is from HWasP, and the second number is from HWasP-Minimal.

| | Pulley radius [mm] | | Spring stiffness [Nmm/rad] | | Spring preload [rad] | |
|-----------------|--------------------|-----|----------------------------|-----|----------------------|-----|
| Thumb proximal | 10.0 | 8.1 | 6.2 | 7.0 | 2.0 | 3.2 |
| Thumb distal | 7.5 | 6.1 | 6.1 | 9.2 | 1.5 | 3.1 |
| Finger proximal | 3.0 | 5.0 | 6.1 | 8.2 | 1.9 | 3.4 |
| Finger distal | 2.6 | 4.0 | 5.9 | 9.2 | 1.1 | 3.1 |

Hardware as Policy. As shown in Fig. 13.2, we use the aforementioned tendon transmission model to build the computational graph of the hardware policy and connect it with the computational policy. The redefined action \mathbf{a}^{new} contains the palm position command output by the computational policy, and the joint torques produced by the hardware policy. The rest of the environment comprises the hand-object system without the tendon underactuation mechanisms, i.e. with independent joints.

Hardware as Policy — Minimal. In this case, all hardware parameters are simply appended to the output of the computational policy. The underactuated transmission model is part of the environment, along with the rest of the hand as well as the object.

13.4 Results

Numerical Results. Our results show that we can learn effective hardware parameters. The resulting pulley radii, spring stiffnesses and preload angles are shown in Table 13.1, using HWasP and HWasP-Minimal respectively. We note that the resulting parameters do not necessarily need to be identical: the optimal set of underactuation parameters is not unique in nature (for example, scaling pulley radii does not change the grasping behavior; for another example, higher spring stiffness and a higher spring preload angle have similar effects), the evaluation is also noisy since we intentionally injected noise, and the gradient-based training process may also settle in local optima in the optimization landscape. Even so, all these parameters are good enough for this tasks.

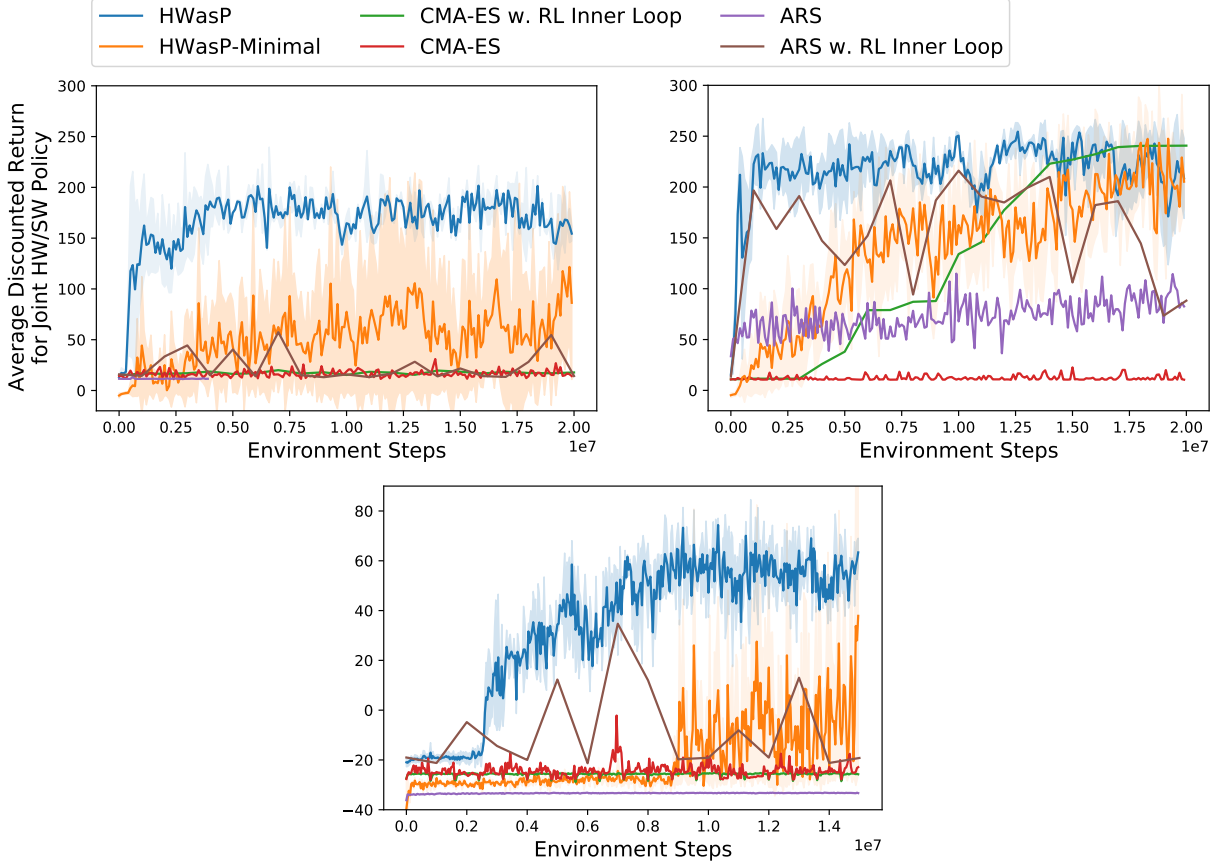


Figure 13.3: Training curves for the hand co-design problem. Top-left: Z-Grasp with a large hardware parameter search range. Top-right: Z-Grasp with a small hardware search range. Bottom: 3D-Grasp with a small search range.

Training Curves. Our results are shown in Fig. 13.3. The top left plot is the Z-Grasp with a large hardware parameter search range. In this case, HWasP learns an effective computational/hardware policy. HWasP-Minimal also learns, but at a slower pace. Neither hyperparameter search nor evolutionary strategies show any learning behavior over a similar number of training steps.

The top-right plot is the Z-Grasp with a small hardware parameter search range. We note that this is realistic because the choice of SI units makes the hardware parameters (e.g. pulley radii in meters) very small. Here, some of our baselines can also learn effective policies, particularly with an RL inner loop, but HWasP is still the most efficient.

Finally, we investigated performance for the more complex 3D-Grasp task. Here, HWasP learned an effective policy, and followed by HWasP-Minimal, while none of the baseline methods

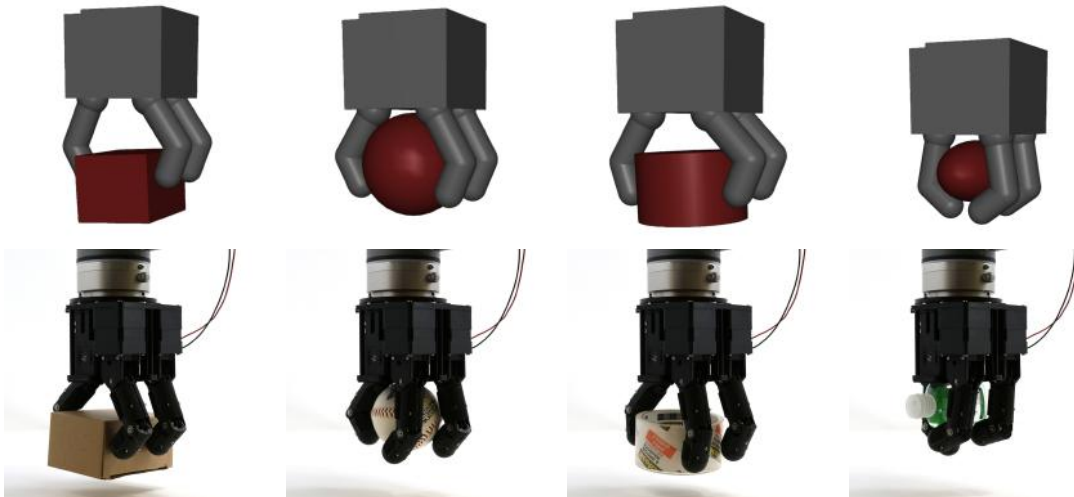


Figure 13.4: Successful grasps in simulation and on a physical hand.

displayed any learning behavior over a similar timescale.

13.5 Prototyping and Validation

To validate our results in the real world, we physically built the hand with the parameters resulted from the co-optimization. The hand is 3D printed, and actuated by a single position-controlled servo motor. Fig. 13.4 shows grasps obtained by this physical prototype, compared to their simulated counterparts. All shown grasps are stable and allow object lift and transport. We note that, as expected, the hand is highly versatile and can perform a wide range of both fingertip and enveloping grasps, for objects of varying shape and size. This shows that the optimized hardware policy is indeed effective in the real world.

Furthermore, we tested the combined hardware and computational policies for both Z-Grasp and 3D-Grasp on the real hand. Here, the computational policy determined how to position the hand (implemented in practice using a UR5 robot with position control) and also issued all commands to the hand servo. While driving the optimized hand, the computational policy achieved 100% success rate for Z-Grasp with boxes and spheres and 90% with cylinders. For 3D-Grasp, the success rate was 100% on boxes, 80% on cylinders and 50% on spheres. We note however that the computational policy expects object pose as input, which was not directly available on our experi-

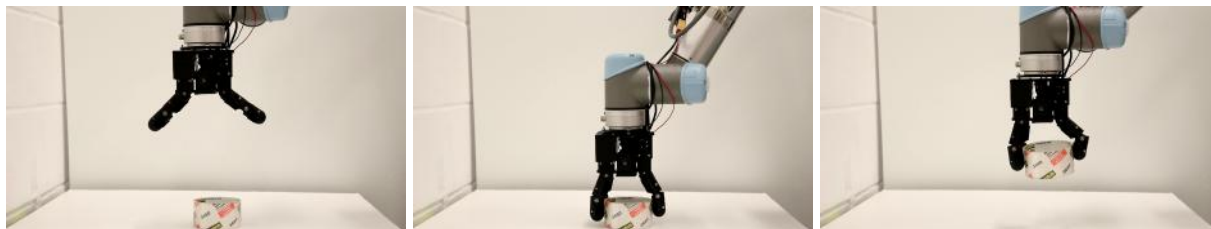


Figure 13.5: Policy deployment on a real robot.

mental setup; instead, we started each run with the object in a known location, which was provided as part of the observation. However, on the spherical object, premature contacts occasionally caused untracked object displacement, which lowered the success rate. While we hope to train and test more complex computational policies in the future, including with real sensor feedback, we believe these experiments underlined the effectiveness of the hardware policy, and its ability to operate together with the jointly optimized computational policy. Fig. 13.5 shows an example grasp in the 3D-Grasp case. Videos are available at <https://roamlab.github.io/hwasp/>.

Chapter 14: Hardware as Policy: Conclusion and Future Work

14.1 Contributions

We summarize the concrete novelty of this chapter as follows:

- To the best of our knowledge, *we are the first to express hardware and computation as a unified RL policy by leveraging auto-differentiable physics, which allows the algorithm to propagate gradients w.r.t both hardware and computational parameters simultaneously and optimize them in the same fashion.*
- We also present a minimal implementation of our method for scenarios where auto-differentiable physics modeling is non-trivial.
- Via case studies comprising both a toy problem and a real-world design challenge, we show that such gradient-based methods are superior to hyperparameter tuning as well as gradient-free evolutionary strategies for hardware-policy co-optimization.
- To the best of our knowledge, *we are the first to build a physical prototype and deploy the trained policy on it to validate a deep-RL-based hardware-policy co-optimization approach.*

14.2 Discussion

Our results show that the HWasP approach is able to learn combined computational and hardware policies. We attribute this performance to the fact that HWasP connects different hardware parameters via a computational graph based on the laws of physics, and can provide the physics-based gradient of the action probability w.r.t the hardware parameters. The HWasP-Minimal implementation does not provide such information, and the policy gradient can only be estimated

via sampling, which is usually less efficient, particularly for high-dimensional problems. In consequence, HWasP-Minimal also shows the ability to learn effective policies, but with reduced performance.

Compared to gradient-free evolutionary baselines for joint hardware-software co-optimization, HWasP always learns faster, while HWasP-Minimal is at least as effective as the best baseline algorithm. We note that combining an RL inner loop for the computational policy with a CMA-ES outer loop for hardware parameters proved more effective than directly using CMA-ES for the complete problem. Still, HWasP outperforms both methods.

The biggest advantage of HWasP-Minimal is that, like gradient-free methods, it does not depend on auto-differentiable physics, and is widely applicable with straightforward implementations to various problems using existing non-differentiable physics engines. We believe that our methods represent a step towards a framework where an algorithm designer can "tune the slider" to decide how much physics to include in the computational policy, based on the trade-offs between computation efficiency, ease of development, and the availability of auto-differentiable physics simulations.

In its current stage, our work still presents a number of limitations. The computational aspects of the policies we have explored so far are relatively simple (e.g. limiting hand motion to 1- or 3-DOF). We hope to explore more challenging robotic tasks, including 6-DOF hand positioning. Here, we also used HWasP with a single task, but believe it is possible to learn more robust hardware parameters by trying to generalize to multiple tasks. Finally, we aim to include additional hardware aspects in the optimization, such as mechanism kinematics, morphology, or link dimensions.

14.3 On-Going and Future Work

Potential Improvements

Adding More Degrees-of-Freedoms for the Hand Co-Design Case. The hand design study in the existing work has limited DoFs: the palm motion only has x, y, z translation (without rotations),

and the fingers are all moving in parallel planes (without adduction-abduction). These DoFs were initially removed to reduce the difficulty of learning, but it also got rid of some interesting motor behaviors: for example, the hand cannot align itself with the principal axis of a long object to make a better grasp.

We aim to articulate these previously fixed DoFs. For the palm rotations, it is straightforward since all DoFs are fully-actuated. For the adduction-abduction of fingers, we aim to use the same tendon routing scheme as the Fig.5.1 showed in Chapter 5, and we need to add the auto-differentiable tendon models for the adduction-abduction joints.

Mechanically-Constrained Neural Network Hardware Policy. As mentioned in Chapter 13, one major issue of the existing method is that, when the trainable hardware parameters (can be either very large or small) have a very different scale from the weights and biases in the neural network (usually small), the back-propagation results in gradients in very different scales, and thus the training tends to be unstable. We proposed a method of scaling these parameters to solve this issue, but this method requires the hardware policy computation has the distributive property, which is the case in our tendon model but not general.

An on-going work is to replace the original Hardware Policy with a neural network for training purposes (thus both Hardware Policy and Computational Policy are neural networks with small weights and biases in similar scales), but still keep the Hardware Policy network close to physically realizable hardware computational graph. We are working on several different methods to impose such constraints during training.

Long-Term Vision

Hand Topology Optimization. The existing HWasP work assumes a predefined robot kinematic structure. A long-term goal is to optimize the topology of a robot. In the hand case, this includes the number of fingers, number of links in each finger, tendon connectivity, etc.

The first step is to optimize the tendon connectivity patterns, e.g., which motor connects to

which finger joint, how many motors are necessary, etc. A promising path is to model the tendon connectivity by a graph and perform graph optimization, with some recently developed techniques in the Graph Neural Network (GNN) field. We aim to express both motors and joints as nodes in the graph, and the transmission as edges. The reward should provide an incentive to reduce the number of edges and nodes (i.e. reduce the number of motors, joints and transmissions) while generating good grasps at the same time. Related works in this path include [14] and [138].

HWasP on Sensing. The existing HWasP work only focused on the mechanical aspects. However, it is also an interesting idea to explore the sensing aspects. Take vision for example, it is possible to build auto-differentiable camera models, containing trainable parameters such as focal lengths and camera locations and orientations. Then we can train a vision-based policy network together with a camera model.

New Applications. In addition to grasping, we aim to apply the HWasP method to a broader range of robotic problems. *In-hand manipulation using underactuated hands* is another interesting use case. Design for in-hand manipulation is a heated topic by itself. Also, in in-hand manipulation tasks, physical interactions happen all the time, which can provide rich useful signals for RL, which helps training for both Computational Policy and Hardware Policy in HWasP.

Another interesting future direction is to explore *(semi-)passive bipedal locomotion* using HWasP. The (semi-)passive bipedal robot is a good example that requires both well-designed control and optimized mechanisms [6]. We expect that our method can co-design the mechanism and control policy of a bipedal walking robot that has minimal actuation and potentially underactuated legs but is optimized to walk stably on uneven terrains using minimal energy.

Part IV

Conclusion

Chapter 15: Conclusion

In this thesis, we presented three projects around the core idea of the integration and interplay of Mechanical Intelligence and Computational Intelligence. The success of the proposed methods all demonstrated the beauty and power of integrating mechanical and computational aspects. Especially, the advances in the design of simple but versatile hand hardware and the learning/optimization methods dance hand in hand with each other in an elegant fashion.

In this chapter, we first re-summarize the contributions, then we propose some high-level methodologies or suggestions to robotics researchers for the practice of integrating Mechanical and Computational Intelligence. Finally, we discuss the potential short-term improvements and long-term challenges and opportunities in this interdisciplinary area.

15.1 Contributions

In this section, we recap the major contributions of this thesis part by part.

Underactuation Design and Hand Synergies

Following the high-level idea to learn Mechanical Intelligence from grasp data with computational methods, we are the first to introduce the dimensionality reduction into hand underactuation design. Specifically, we first collected useful grasps with a given hand kinematic model a set of intended objects in simulation, *without* considering underactuation; then we found low-dimensional manifolds in the joint position and torque space to fit these data point as much as possible via optimization (quantified by physics-based metrics such as stability), with constraints that such manifolds must be physically realizable by underactuation mechanisms. We termed such manifolds “Mechanically Realizable Manifolds”. The objective values of the optimization can also be

used as “hand quality metrics” to compare different hand kinematics.

The aforementioned “mechanical dimensionality reduction” requires a prespecified underactuation transmission topology, specifically the tendon routing scheme for tendon-driven systems. We further investigated different tendon transmission paradigms, and formalized a design matrix with the choice of agonist and antagonist in one dimension, and the coupling scheme across different joints as the other dimension. We proposed an underactuation design combining two paradigms in this matrix and made it a physical prototype. This design not only exhibits the desired grasping behavior, but also enables the power-off pose-keeping and human-intervention, which are crucial to our ultimate application — for the *Astrobee* free-flyer in the International Space Station.

Compliant Grasping using Proprioception

This part is about Computational Intelligence which leverages mechanism features: we discussed mechanical compliance and its interplay with the control algorithms. We designed a Series-Elastic-Actuated hand equipped with both passive compliance and active adaptation from proprioception. This hardware platform provides a good position and torque control performance.

We also proposed a grasping controller leveraging the proprioception and the embodied compliance in the SEA. We first let the hand make an initial light touch and then use this controller to load the grasp, aiming at increasing the contact forces but not moving the object. We showed that this controller is effective in the fingertip and enveloping grasps, as well as some simple in-hand manipulation tasks.

RL-based Mechanical-Computational Co-Optimization

In the last part of this thesis, we demonstrate the most powerful technique for the interplay of Mechanical and Computation Intelligence — the concurrent design of both. We demonstrated the “Hardware as Policy” method, which is a co-optimization approach for both mechanisms and control policies leveraging the power of deep Reinforcement Learning (specifically Policy Optimization). This is the first time we can design the hardware parameters and computational policy

parameters in the same fashion, by expressing the hardware as auto-differentiable physics and connect it with the neural network control policy.

We illustrated this method in a real-world design case of the mechanism and control of an underactuated hand. We showed that our method outperforms our baselines — hyperparameter tuning and evolutionary computation — in training time, and is effective when deployed to real hardware.

15.2 Suggestions for Integrating Mechanical and Computational Intelligence

Here we summarize some of the high-level thinking or suggestions for robotics researchers for integrating Mechanical and Computational Intelligence.

Discover the Concept Correspondences and Borrow Ideas

The robot hardware and software fields are developed on top of different fundamental sciences and by different groups of people. However, there are concepts where the underlining mathematical principles are similar. When such correspondences exist, it is a good idea to take the advances from one domain to the other.

A good example is the “mechanical dimensionality reduction” introduced in Chapter 3: the dimensionality reduction or manifold learning is well-studied in the statistics and machine learning community; in the field of mechanism design, a corresponding concept is the underactuation. Taking the idea of dimensionality reduction (by optimizing certain metrics to make the low-dimension manifold as representative as possible) while ensuring the physical feasibility, we proposed the “Mechanically Realizable Manifolds” and the computational methods to obtain them.

Another example is the “Mechanical Policy” introduced in Chapter 11. We posit that mechanics is just akin to neural network policies in a way that both can be expressed as a computational graph. This leads to the idea to treat mechanics and neural network policies the same way, which allows efficient gradient computation and optimization.

Optimize with and for Each Other

This is the main idea of Chapter 11. We believe it is beneficial to design the mechanisms and policies simultaneously, by optimizing both parts with and for each other, since the performance of a robot is determined by both the mechanical structures and the controller. This is especially useful if the robot is designed for a certain task or category of tasks that have certain patterns, for example, grasping, walking, etc.

Look for Solutions from the Other Domain

This idea is already used practiced in the engineering design of a variety of robot systems. It is a good practice to always look at the solutions from the hardware and software domains and compare them. In certain cases, one may outperform the other.

For example, mechanical synergies and software synergies both provide joint coordination. In cases where only grasping is concerned and mechanical and control simplicity is crucial, we may choose mechanical synergies (underactuation). In cases where the hand also needs to perform tasks other than grasping (e.g. in-hand manipulation), software synergies can be used for grasping and can also be turned off for other tasks.

Another example is about introducing the Series Elastic Actuation (SEA) to realize compliance in hands in Chapter 7. Comparing to the software solution to render compliance using current-based Proprioceptive Actuators [139], SEAs (hardware compliance) can realize a higher torque density in highly constrained spaces in hands. Comparing to the torque-sensor-based software solutions, SEAs are much more cost-effective and are not limited by control bandwidth.

15.3 Limitations and Improvements

We have listed the limitations and to-be-refined items at the end of each chapter. Here, at a thesis level, we list some important limitations and potential improvements.

Hand Dimension Optimization

Regarding the hand design, the design cases used in the proposed methods, including “Mechanically Realizable Manifolds” and “Hardware as Policy”, require a hand model (without the underactuation mechanisms) for simulation, i.e. require prespecified hand dimensions (e.g. link lengths). We aim to explore the problem of optimizing the geometric parameters, either as an additional optimization layer (which is computationally expensive) or using “Hardware as Policy”.

Hand Topology Design

Even if we can include link dimensions as optimization variables, the aforementioned methods still need a prespecified hardware topology. A major improvement on those methods is to do hardware topology optimization (e.g. how many fingers are needed, how tendons and joints should be connected, etc.). It is possible that we can use the existing methods as the inner loop and the topology search as the outer loop, but such an exhaustive search will be very expensive. A promising technical path is through graph optimization — to express the topology as a graph and perform state-of-the-art learning/optimization techniques on it. Also, we hope to integrate the topology optimization with the optimization of the morphology or actuation parameters, instead of treating the topology optimization as an outer loop.

General Blind Grasping

The existing method introduced in Chapter 8 simplified the problem by grasping only in 2D and also with a limited number of objects. Also, the “blind exploration” phase is simplified which does not require palm relocation. We aim to explore a more general blind grasping using force-based feedback — for example, the hand should move in 3D, it should actively explore the objects, and the objects should be more representable of the real use case.

15.4 Long-Term Challenges and Opportunities

Looking forward, we believe there are still some major challenges and also opportunities in the practice of mechanical-computational fusion for robotics. We list them as follows.

Auto-differentiable Physics Engine

Auto-differentiable physics is very useful for optimization purposes — in addition to computing the next state by forward dynamics, it can also provide the gradient w.r.t the previous states or the physics parameters. Such gradients can be used for gradient-descent algorithms which are usually more efficient than gradient-free black-box optimization methods.

In Chapter 11, we proposed the idea to combine the hardware expressed as auto-differentiable physics and the software policy and optimize them together. However, we only presented examples with hand-coded Hardware Policies for specific problems. To make this method more widely applicable, we need a general-purpose auto-differentiable physics engine, which internally uses auto-differentiable computational graphs to perform the computation for physics. This is a research topic that draws more and more attention these years [103, 104, 105, 106, 107]. However, to the best of our knowledge, an engine that is fast, robust and general enough still does not exist yet. When such an auto-differentiable physics engine comes into being, we believe our method can play well with it.

Simulation Fidelity and Sim-to-Real Transfer

Techniques such as the ones showed in Chapter 3 and 11 require simulation for hardware optimization. Unlike software which can keep improving after deployed to real robots, the mechanisms are fixed when the design phase is done. This makes the simulation fidelity crucial.

The recent advances in physics simulations [15, 16, 17] made great achievements in terms of robustness and fidelity. However, the issue of the “reality gap” is still not solved. For example, these simulators are based on rigid-body assumptions, meaning they are not suitable to simulate

deformable bodies (e.g. rubbers, fabrics, tendons) and the interaction with fluids (e.g. suction cups). One step further, future multi-domain co-optimization may require the simulator to be able to capture multi-physics phenomena, including optics, thermal, or acoustics.

Sim-to-real transfer techniques, such as Domain Randomization [18], are well-studied for software policies. However, it is still unclear how and how much they can help in the transfer of robot hardware.

Generalization of Task-driven Design

The last challenge is the generalization of the hardware performance on environments it is not optimized for. On the machine learning side, generalization is well-studied. For robot hardware designed by task-driven learning/optimization methods, this is still an unanswered question. We can borrow a lot of ideas from the existing work on the generalization of machine learning, but new theories specific to the hardware design also need to come out.

15.5 Summary

In conclusion, we believe the idea of combining hardware-embedded and software-based intelligence is a promising path towards versatile robotic grasping and manipulation. Our works on the data-driven underactuated hand design, the compliant grasping with mechanical-computational adaptation, and the RL-based mechanical-computation co-optimization are meaningful attempts along this path. We aim to explore more possibilities in the intersection area between these two aspects. We believe when software better understands the physical world and when the hardware is better designed to enhance the algorithmic performance, we will get closer to the world described in science fiction where robots are everywhere, doing various tasks for humans every day.

References

- [1] S. Russell and P. Norvig, “Artificial intelligence: A modern approach,” 2002.
- [2] A. Kaplan and M. Haenlein, “Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence,” *Business Horizons*, vol. 62, no. 1, pp. 15–25, 2019.
- [3] N. T. Ulrich, “Grasping with mechanical intelligence,” Ph.D. dissertation, University of Pennsylvania, Philadelphia, Dec. 1989.
- [4] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, “Universal robotic gripper based on the jamming of granular material,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 44, pp. 18 809–18 814, 2010.
- [5] J. Franch, S. K. Agrawal, and V. Sangwan, “Differential flatness of a class of n -dof planar manipulators driven by 1 or 2 actuators,” *IEEE transactions on automatic control*, vol. 55, no. 2, pp. 548–554, 2010.
- [6] S. H. Collins and A. Ruina, “A bipedal walking robot with efficient and human-like gait,” in *IEEE international Conference on Robotics and Automation*, IEEE, 2005, pp. 1983–1988.
- [7] L. Birglen, T. Laliberté, and C. M. Gosselin, *Underactuated robotic hands*. Springer, 2007, vol. 40.
- [8] A. M. Dollar and R. D. Howe, “Towards grasping in unstructured environments: Grasper compliance and configuration optimization,” *Advanced Robotics*, vol. 19, no. 5, pp. 523–543, 2005.
- [9] M. Ciocarlie and P. Allen, “A constrained optimization framework for compliant underactuated grasping,” *Mechanical Sciences*, vol. 2, no. 1, pp. 17–26, 2011.
- [10] H. Dong, E. Asadi, C. Qiu, J. Dai, and I.-M. Chen, “Geometric design optimization of an under-actuated tendon-driven robotic gripper,” *Robotics and Computer-Integrated Manufacturing*, vol. 50, pp. 80–89, 2018.
- [11] M. Ciocarlie, F. M. Hicks, R. Holmberg, J. Hawke, M. Schlicht, J. Gee, S. Stanford, and R. Bahadur, “The velo gripper: A versatile single-actuator design for enveloping, parallel and fingertip grasps,” *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 753–767, 2014.

- [12] K. Sims, “Evolving virtual creatures,” in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 1994, pp. 15–22.
- [13] F. L. Hammond, J. Weisz, A. Andrés, P. K. Allen, and R. D. Howe, “Towards a design optimization method for reducing the mechanical complexity of underactuated robotic hands,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2843–2850.
- [14] T. Wang, Y. Zhou, S. Fidler, and J. Ba, “Neural graph evolution: Towards efficient automatic robot design,” *arXiv preprint arXiv:1906.05370*, 2019.
- [15] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.
- [16] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, <http://pybullet.org>, 2016–2019.
- [17] (2018). “Isaac platform for robotics,” NVIDIA.
- [18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2017, pp. 23–30.
- [19] (1954). “Unimate - robots: Your guide to the world of robotics,” General Motors.
- [20] (2018). “Onrobot two-fingered gripper,” OnRobot.
- [21] (2006). “PR2 - hardware and software platform for mobile manipulation R&D,” Willow Garage.
- [22] (2014). “Fetch mobile manipulator,” Fetch Robotics.
- [23] J. K. Salisbury and J. J. Craig, “Articulated hands: Force control and kinematic issues,” *The International journal of Robotics research*, vol. 1, no. 1, pp. 4–17, 1982.
- [24] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers, “Design of the utah/mit dextrous hand,” in *IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 1986, pp. 1520–1532.
- [25] C. Loucks, V. Johnson, P. Boissiere, G. Starr, and J. Steele, “Modeling and control of the stanford/jpl hand,” in *IEEE International Conference on Robotics and Automation*, IEEE, vol. 4, 1987, pp. 573–578.

- [26] L.-R. Lin and H.-P. Huang, "Ntu hand: A new design of dexterous hands," *Journal of Mechanical Design*, vol. 120, no. 2, pp. 282–292, 1998.
- [27] C. Lovchik and M. A. Diftler, "The robonaut hand: A dexterous robot hand for space," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 907–912.
- [28] (2005). "Shadow dexterous hand," Shadow Robot Company.
- [29] A. D. Deshpande, Z. Xu, M. J. V. Weghe, B. H. Brown, J. Ko, L. Y. Chang, D. D. Wilkinson, S. M. Bidic, and Y. Matsuoka, "Mechanisms of the anatomically correct testbed hand," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 1, pp. 238–250, 2011.
- [30] M. Quigley, C. Salisbury, A. Y. Ng, and J. K. Salisbury, "Mechatronic design of an integrated robotic hand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 706–720, 2014.
- [31] Z. Xu and E. Todorov, "Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration," in *IEEE International Conference on Robotics and Automation*, IEEE, 2016, pp. 3485–3492.
- [32] C. Gosselin, F. Pelletier, and T. Laliberte, "An anthropomorphic underactuated robotic hand with 15 dofs and a single actuator," in *IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 749–754.
- [33] (2008). "3-finger adaptive robot gripper," Robotiq.
- [34] A. M. Dollar and R. D. Howe, "The highly adaptive sdm hand: Design and performance evaluation," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 585–597, 2010.
- [35] L. Wang, J. DelPreto, S. Bhattacharyya, J. Weisz, and P. K. Allen, "A highly-underactuated robotic hand with force and joint angle sensors," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 1380–1385.
- [36] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, "A compliant, underactuated hand for robust manipulation," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 736–752, 2014.
- [37] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit soffthand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

- [38] H. Stuart, S. Wang, O. Khatib, and M. R. Cutkosky, “The ocean one hands: An adaptive design for robust marine manipulation,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 150–166, 2017.
- [39] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2016, pp. 3406–3413.
- [40] V. Kumar, E. Todorov, and S. Levine, “Optimal control with learned local models: Application to dexterous manipulation,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2016, pp. 378–383.
- [41] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” *arXiv preprint arXiv:1703.09312*, 2017.
- [42] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.
- [43] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations,” *arXiv preprint arXiv:1709.10087*, 2017.
- [44] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, “Learning dexterous in-hand manipulation,” *arXiv preprint arXiv:1808.00177*, 2018.
- [45] S. Yuan, L. Shao, C. L. Yako, A. Gruebele, and J. K. Salisbury, “Design and control of roller grasper v2 for in-hand manipulation,” *arXiv preprint arXiv:2004.08499*, 2020.
- [46] (2010). “How do hands work?” InformedHealth.org.
- [47] H. Liu, K. Wu, P. Meusel, N. Seitz, G. Hirzinger, M. Jin, Y. Liu, S. Fan, T. Lan, and Z. Chen, “Multisensory five-finger dexterous hand: The dlr/hit hand ii,” in *2008 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2008, pp. 3692–3697.
- [48] S. Hirose and Y. Umetani, “The development of soft gripper for the versatile robot hand,” *Mechanism and machine theory*, vol. 13, no. 3, pp. 351–359, 1978.
- [49] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [50] J. Mockus, *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media, 2012, vol. 37.

- [51] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.
- [52] J. Rosell, R. Suárez, C. Rosales, and A. Pérez, "Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures," *Autonomous Robots*, vol. 31, no. 1, p. 87, 2011.
- [53] T. Wimböck, B. Jahn, and G. Hirzinger, "Synergy level impedance control for multifingered hands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 973–979.
- [54] G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo, "Mapping synergies from human to robotic hands with dissimilar kinematics: An approach in the object domain," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 825–837, 2013.
- [55] C. Meeker, T. Rasmussen, and M. Ciocarlie, "Intuitive hand teleoperation by novice operators using a continuous teleoperation subspace," in *IEEE International Conference on Robotics and Automation*, 2018.
- [56] G. C. Matrone, C. Cipriani, E. L. Secco, G. Magenes, and M. C. Carrozza, "Principal components analysis based control of a multi-dof underactuated prosthetic hand," *Journal of neuroengineering and rehabilitation*, vol. 7, no. 1, p. 16, 2010.
- [57] G. C. Matrone, C. Cipriani, M. C. Carrozza, and G. Magenes, "Real-time myoelectric control of a multi-fingered hand prosthesis using principal components analysis," *Journal of neuroengineering and rehabilitation*, vol. 9, no. 1, p. 40, 2012.
- [58] A. Tsoli and O. C. Jenkins, "Robot grasping for prosthetic applications," in *The International Journal of Robotics Research*, 2010, pp. 1–12.
- [59] C. Y. Brown and H. H. Asada, "Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2877–2882.
- [60] K. Xu, H. Liu, Y. Du, and X. Zhu, "Design of an underactuated anthropomorphic hand with mechanically implemented postural synergies," *Advanced Robotics*, vol. 28, no. 21, pp. 1459–1474, 2014.
- [61] K. Xu, Z. Liu, B. Zhao, H. Liu, and X. Zhu, "Composed continuum mechanism for compliant mechanical postural synergy: An anthropomorphic hand design example," *Mechanism and Machine Theory*, vol. 132, pp. 108–122, 2019.
- [62] S. Li, X. Sheng, H. Liu, and X. Zhu, "Design of a myoelectric prosthetic hand implementing postural synergy mechanically," *Industrial Robot*, vol. 41, no. 5, pp. 447–455, 2014.

- [63] W. Chen, C. Xiong, and S. Yue, “Mechanical implementation of kinematic synergy for continual grasping generation of anthropomorphic hand,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1249–1263, 2015.
- [64] C. Xiong, W. Chen, B. Sun, M. Liu, S. Yue, and W. Chen, “Design and implementation of an anthropomorphic hand for replicating human grasping functions,” *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 652–671, 2016.
- [65] M. Gabiccini, A. Bicchi, D. Prattichizzo, and M. Malvezzi, “On the role of hand synergies in the optimal choice of grasping forces,” *Autonomous Robots*, vol. 31, no. 2-3, p. 235, 2011.
- [66] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi, “On motion and force controllability of precision grasps with hands actuated by soft synergies,” *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1440–1456, 2013.
- [67] G. Grioli, M. Catalano, E. Silvestro, S. Tono, and A. Bicchi, “Adaptive synergies: An approach to the design of under-actuated robotic hands,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1251–1256.
- [68] G. A. Pratt and M. M. Williamson, “Series elastic actuators,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1995, pp. 399–406.
- [69] R. Ma and A. Dollar, “Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption,” *IEEE Robotics & Automation Magazine*, vol. 24, no. 1, pp. 32–40, 2017.
- [70] J. Butterfaß, M. Grebenstein, H. Liu, and G. Hirzinger, “Dlr-hand ii: Next generation of a dextrous robot hand,” in *IEEE International Conference on Robotics and Automation*, vol. 1, 2001, pp. 109–114.
- [71] A. Edsinger-Gonzales and J. Weber, “Domo: A force sensing humanoid robot for manipulation research,” in *4th IEEE/RAS International Conference on Humanoid Robots*, vol. 1, 2004, pp. 273–291.
- [72] E. Torres-Jara, “Obrero: A platform for sensitive manipulation,” in *5th IEEE/RAS International Conference on Humanoid Robots*, 2005, pp. 327–332.
- [73] T. Yoshikawa, “Control algorithm for grasping and manipulation by multifingered robot hands using virtual truss model representation of internal force,” in *IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 369–376.
- [74] S. Arimoto, P. T. A. Nguyen, H.-Y. Han, and Z. Doulgeri, “Dynamics and control of a set of dual fingers with soft tips,” *Robotica*, vol. 18, no. 01, pp. 71–80, 2000.

- [75] S. A. Schneider and R. H. Cannon, “Object impedance control for cooperative manipulation: Theory and experimental results,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 383–394, 1992.
- [76] S. Arimoto, R. Ozawa, and M. Yoshida, “Two-dimensional stable blind grasping under the gravity effect,” in *IEEE International Conference on Robotics and Automation*, 2005, pp. 1196–1202.
- [77] M. Yoshida, S. Arimoto, and J.-H. Bae, “Blind grasp and manipulation of a rigid object by a pair of robot fingers with soft tips,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 4707–4714.
- [78] D. Wang, B. T. Watson, and A. H. Fagg, “A switching control approach to haptic exploration for quality grasps,” *Robotic Science and Systems*, 2007.
- [79] R. Platt Jr, A. H. Fagg, and R. A. Grupen, “Null-space grasp control: Theory and experiments,” *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 282–295, 2010.
- [80] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, “Contact-reactive grasping of objects with partial shape information,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1228–1235.
- [81] J. Felip, J. Bernabé, and A. Morales, “Contact-based blind grasping of unknown objects,” in *IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2012, pp. 396–401.
- [82] H. Dang, J. Weisz, and P. K. Allen, “Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 5917–5922.
- [83] A. Murali, Y. Li, D. Gandhi, and A. Gupta, “Learning to grasp without seeing,” in *International Symposium on Experimental Robotics*, Springer, 2018, pp. 375–386.
- [84] B. Wu, I. Akinola, J. Varley, and P. Allen, “Mat: Multi-fingered adaptive tactile grasping via deep reinforcement learning,” *arXiv preprint arXiv:1909.04787*, 2019.
- [85] M. Huber and R. A. Grupen, “2-d contact detection and localization using proprioceptive information,” *IEEE Transactions on Robotics and Automation*, vol. 10, no. 1, pp. 23–33, 1994.
- [86] B. Belzile and L. Birglen, “Stiffness analysis of underactuated fingers and its application to proprioceptive tactile sensing,” *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 6, pp. 2672–2681, 2016.

- [87] S. J. Gordon and W. T. Townsend, “Integration of tactile force and joint torque information in a whole-arm manipulator,” in *IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 1989, pp. 464–465.
- [88] B. S. Eberman and J. K. Salisbury, “Determination of manipulator contact information from joint torque measurements,” in *Experimental Robotics I*, Springer, 1990, pp. 463–473.
- [89] L. Manuelli and R. Tedrake, “Localizing external contact using proprioceptive sensors: The contact particle filter,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2016, pp. 5062–5069.
- [90] J.-H. Park and H. Asada, “Concurrent design optimization of mechanical structure and control for high speed robots,” *Journal of dynamic systems, measurement, and control*, vol. 116, no. 3, pp. 344–356, 1994.
- [91] C. Paul and J. C. Bongard, “The road less travelled: Morphology in the optimization of biped robot locomotion,” in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, vol. 1, pp. 226–232.
- [92] T. Geijtenbeek, M. Van De Panne, and A. F. Van Der Stappen, “Flexible muscle-based locomotion for bipedal creatures,” *ACM Transactions on Graphics*, vol. 32, no. 6, p. 206, 2013.
- [93] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, “Computational co-optimization of design parameters and motion trajectories for robotic systems,” *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1521–1536, 2018.
- [94] T. Liao, G. Wang, B. Yang, R. Lee, K. Pister, S. Levine, and R. Calandra, “Data-efficient learning of morphology and controller for a microrobot,” in *2019 International Conference on Robotics and Automation*, IEEE, 2019, pp. 2488–2494.
- [95] H. Lipson and J. B. Pollack, “Automatic design and manufacture of robotic lifeforms,” *Nature*, vol. 406, no. 6799, p. 974, 2000.
- [96] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, “Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding,” *ACM SIGEVOlution*, vol. 7, no. 1, pp. 11–23, 2014.
- [97] N. Cheney and H. Lipson, “Topological evolution for embodied cellular automata,” *Theoretical Computer Science*, vol. 633, pp. 19–27, 2016.
- [98] T. F. Nygaard, C. P. Martin, E. Samuelsen, J. Torresen, and K. Glette, “Real-world evolution adapts robot morphology and control to hardware limitations,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 125–132.

- [99] D. Ha, “Reinforcement learning for improving agent design,” *arXiv preprint arXiv:1810.03779*, 2018.
- [100] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, “Jointly learning to construct and control agents using deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2019, pp. 9798–9805.
- [101] K. Vermeer, R. Kuppens, and J. Herder, “Kinematic synthesis using reinforcement learning,” in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, ASME, vol. 51753, 2018, V02AT03A009.
- [102] K. S. Luck, H. Ben Amor, and R. Calandra, “Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning,” in *Conf. on Robot Learning*, 2019.
- [103] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, “End-to-end differentiable physics for learning and control,” in *Advances in Neural Information Processing Systems*, 2018, pp. 7178–7189.
- [104] M. Gifftthaler, M. Neunert, M. Stäuble, and J. Buchli, “The control toolbox—an open-source c++ library for robotics, optimal and model predictive control,” in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, IEEE, 2018, pp. 123–129.
- [105] J. Degraeve, M. Hermans, J. Dambre, and F. Wyffels, “A differentiable physics engine for deep learning in robotics,” *Frontiers in neurorobotics*, vol. 13, p. 6, 2019.
- [106] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, “Chainqueen: A real-time differentiable physical simulator for soft robotics,” in *International Conference on Robotics and Automation*, IEEE, 2019, pp. 6265–6271.
- [107] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, “DiffTaichi: Differentiable programming for physical simulation,” *arXiv preprint arXiv:1910.00935*, 2019.
- [108] T. Chen, M. Haas-Heger, and M. Ciocarlie, “Underactuated hand design using mechanically realizable manifolds,” in *IEEE International Conference on Robotics and Automation*, 2018, pp. 7392–7398.
- [109] T. Chen, L. Wang, M. Haas-Heger, and M. Ciocarlie, “Underactuation design for tendon-driven hands via optimization of mechanically realizable manifolds in posture and torque spaces,” *IEEE Transactions on Robotics*, 2020.
- [110] M. Santello, M. Flanders, and J. F. Soechting, “Postural hand synergies for tool use,” *Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

- [111] A. M. Dollar and R. D. Howe, “Joint coupling design of underactuated hands for unstructured environments,” *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1157–1169, 2011.
- [112] D. Prattichizzo and J. C. Trinkle, “Grasping,” in *Springer handbook of robotics*, 2008, pp. 671–700.
- [113] N. Hansen, S. D. Müller, and P. Koumoutsakos, “Reducing the time complexity of the de-randomized evolution strategy with covariance matrix adaptation (cma-es),” *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.
- [114] M. Bualat, J. Barlow, T. Fong, C. Provencher, T. Smith, and A. Zuniga, “Astrobee: Developing a free-flying robot for the international space station,” in *AIAA SPACE 2014 Conference and Exposition*, vol. 4643, 2015.
- [115] A. T. Miller and P. K. Allen, “Graspit! a versatile simulator for robotic grasping,” *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.
- [116] C. Ferrari and J. Canny, “Planning optimal grasps,” in *IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.
- [117] T. Chen, T. Zhang, and M. Ciocarlie, “Design paradigms based on spring agonists for underactuated robot hands: Concepts and application,” *arXiv e-prints*, arXiv–2011, 2020.
- [118] I.-W. Park, T. Smith, H. Sanchez, S. W. Wong, P. Piacenza, and M. Ciocarlie, “Developing a 3-dof compliant perching arm for a free-flying robot on the international space station,” in *IEEE International Conference on Advanced Intelligent Mechatronics*, IEEE, 2017, pp. 1135–1141.
- [119] G. Smit, R. M. Bongers, C. K. Van der Sluis, and D. H. Plettenburg, “Efficiency of voluntary opening hand and hook prosthetic devices: 24 years of development,” *J Rehabil Res Dev*, vol. 49, no. 4, pp. 523–34, 2012.
- [120] C. Meeker, M. Haas-Heger, and M. Ciocarlie, “A continuous teleoperation subspace with empirical and algorithmic mapping algorithms for non-anthropomorphic hands,” *arXiv preprint arXiv:1911.09565*, 2019.
- [121] T. Chen and M. Ciocarlie, “Proprioception-based grasping for unknown objects using a series-elastic-actuated gripper,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2018, pp. 6675–6681.
- [122] C. Blanchard, R. Roll, J.-P. Roll, and A. Kavounoudias, “Differential contributions of vision, touch and muscle proprioception to the coding of hand movements,” *PloS one*, vol. 8, no. 4, e62475, 2013.

- [123] J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *trans. ASME*, vol. 64, no. 11, 1942.
- [124] P. Piacenza, W. Dang, E. Hannigan, J. Espinal, I. Hussain, I. Kyminsis, and M. Ciocarlie, “Accurate contact localization and indentation depth prediction with an optics-based tactile sensor,” in *IEEE International Conference on Robotics and Automation*, IEEE, 2017, pp. 959–965.
- [125] T. Chen, Z. He, and M. Ciocarlie, “Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning,” *arXiv preprint arXiv:2008.04460*, 2020.
- [126] R. W. Young, “Evolution of the human hand: The role of throwing and clubbing,” *Journal of Anatomy*, vol. 202, no. 1, pp. 165–174, 2003.
- [127] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, “Learning to walk via deep reinforcement learning,” *arXiv preprint arXiv:1812.11103*, 2018.
- [128] L. Birglen and C. M. Gosselin, “Kinetostatic analysis of underactuated fingers,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 211–221, 2004.
- [129] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, *et al.*, “Policy gradient methods for reinforcement learning with function approximation,” in *NIPS*, Citeseer, vol. 99, 1999, pp. 1057–1063.
- [130] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International Conference on machine learning*, 2015, pp. 1889–1897.
- [131] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [132] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [133] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” in *Proceedings of the International Conference on Learning Representations*, 2016.
- [134] D. P. Kingma and M. Welling, “An introduction to variational autoencoders,” *arXiv preprint arXiv:1906.02691*, 2019.
- [135] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.

- [136] H. Mania, A. Guy, and B. Recht, “Simple random search provides a competitive approach to reinforcement learning,” *arXiv preprint arXiv:1803.07055*, 2018.
- [137] T. garage contributors, *Garage: A toolkit for reproducible reinforcement learning research*, <https://github.com/rlworkgroup/garage>, 2019.
- [138] P. Velickovic, L. Buesing, M. C. Overlan, R. Pascanu, O. Vinyals, and C. Blundell, “Pointer graph networks,” *stat*, vol. 1050, p. 11, 2020.
- [139] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, “Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots,” *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, 2017.

Appendix A: Derivation of the Matrices of Design Case III

This is for the Design Case III in Chapter 4.

First, the inverse kinematics (i.e. calculating the tendon length from angles) for two-DoF proximal joint need to be solved. Fig. A.1 shows the two-DoF joint and the attached coordinate frames. The tendon lengths can be solved using coordinate transforms. For example, the length of the tendon connecting A_1 and B_1 can be calculated as:

$$\|{}^{O_A}\overrightarrow{B_1A_1}\| = \|{}^{O_A}\overrightarrow{OA_1} - {}^{O_A}R_O {}^OR_{O_B} {}^{O_B}\overrightarrow{OB_1}\| \quad (\text{A.1})$$

where the prescripts are the coordinates the vector is described in, and the rotation matrix ${}^{O_A}R_O$ represents the transform from coordinate $\{O_A\}$ to $\{O\}$.

Next, we show the derivation of the Actuation Matrix, which relates the tendon forces and the net joint torques. Here we use the torque of pitch DoF τ_p as an example:

$$\tau_p = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot ({}^{O_A}\overrightarrow{OA_1} \times \frac{{}^{O_A}\overrightarrow{B_1A_1}}{\|{}^{O_A}\overrightarrow{B_1A_1}\|} t_1 + {}^{O_A}\overrightarrow{OA_2} \times \frac{{}^{O_A}\overrightarrow{B_2A_2}}{\|{}^{O_A}\overrightarrow{B_2A_2}\|} t_2 + {}^{O_A}\overrightarrow{OA_3} \times \frac{{}^{O_A}\overrightarrow{B_3A_3}}{\|{}^{O_A}\overrightarrow{B_3A_3}\|} t_3) \quad (\text{A.2})$$

where t_1, t_2, t_3 are the magnitude of tendon forces.

We denote:

$${}^{O_A}\rho = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \left[{}^{O_A}\overrightarrow{OA_1} \times \frac{{}^{O_A}\overrightarrow{B_1A_1}}{\|{}^{O_A}\overrightarrow{B_1A_1}\|} \quad {}^{O_A}\overrightarrow{OA_2} \times \frac{{}^{O_A}\overrightarrow{B_2A_2}}{\|{}^{O_A}\overrightarrow{B_2A_2}\|} \quad {}^{O_A}\overrightarrow{OA_3} \times \frac{{}^{O_A}\overrightarrow{B_3A_3}}{\|{}^{O_A}\overrightarrow{B_3A_3}\|} \right] \quad (\text{A.3})$$

and then:

$$\tau_p = {}^{O_A}\rho \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = {}^{O_A}\rho_1 t_1 + {}^{O_A}\rho_2 t_2 + {}^{O_A}\rho_3 t_3 \quad (\text{A.4})$$

Following the requirement that the post-contact movement is negligible, and the reasoning shown in (3.1) (3.2) and (3.3), the net joint torque can be expressed as:

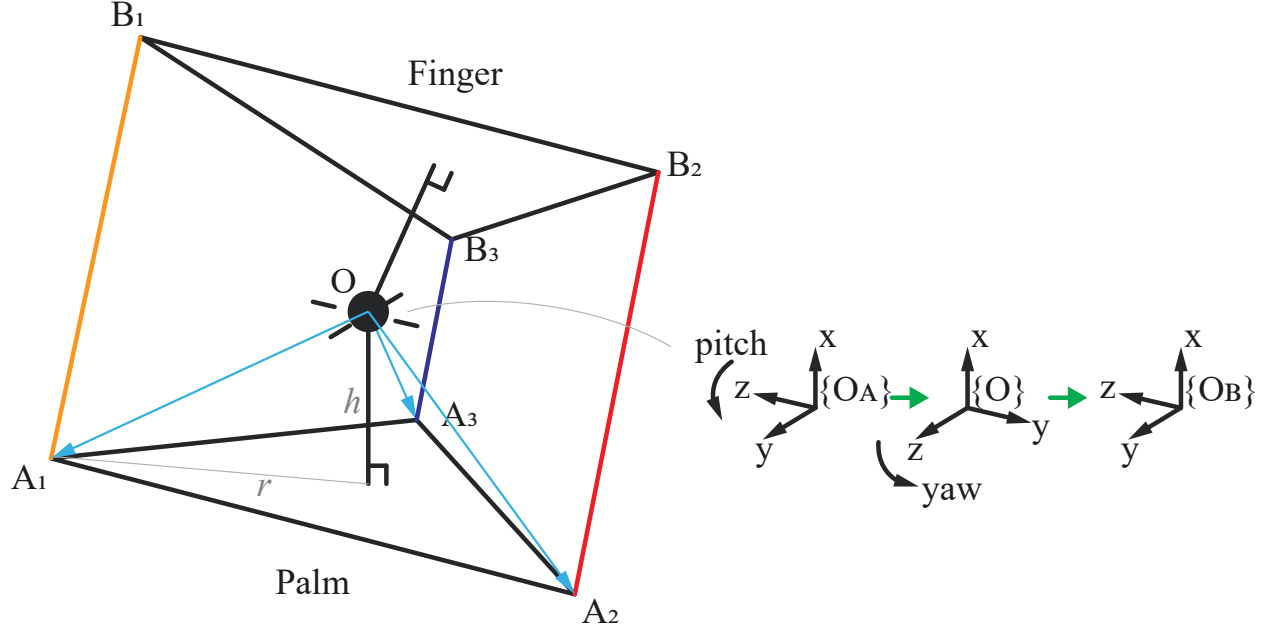


Figure A.1: The 2-DoF universal proximal joint in Design Case III, as well as the attached coordinate frames. The origin of all frames are located in O , and frame O_A , O_B are bonded to the lower and upper platform respectively.

$$\tau_{p,net} = {}^{O_A}\rho_1 t_{1,net} + {}^{O_A}\rho_2 t_{2,net} \quad (\text{A.5})$$

The moment arms ${}^{O_A}\rho_1$ and ${}^{O_A}\rho_2$ in (A.5) are the entries ρ_{fpp11} and ρ_{fpp12} of the Actuation Matrix A shown in (4.6). We can construct the entire Actuation Matrix in a similar way.

Finally, we give some details in the tendon shortening vector s in (4.8). The Δl 's are the tendon length changes between zero-configuration and grasp configuration. For example:

$$\Delta l_{fp11} = l_{1(zero)} - l_{1(grasp)} \quad (\text{A.6})$$

where $l_{1(grasp)}$ and $l_{1(zero)}$ can be solved by the aforementioned joint inverse kinematics using the pitch and yaw angles.

Appendix B: Details of the Experiment Protocols

Here we show the experiments details of the MIMO Grasping Controller in Chapter 8

There are a lot of factors that may influence performance. To have a well-rounded comparison, we swept the following dimensions:

- *Controllers.* The torque ratio is a key parameter for both the Fixed Torque Ratio Controller, and the physically underactuated gripper. We tested the MIMO Grasping Controller against the other two baselines with three different ratios between the distal and proximal joint: 0.3, 0.4 and 0.5.
- *Object shape primitives.* We selected four object shape primitives for the test: a big cylinder (diameter: 67mm), a big box (side length: 57mm), a small cylinder (diameter: 47mm) and a small box (side length: 39 mm). All objects have negligible friction with the table.
- *Object locations.* We swept three locations along the centerline of the gripper within the range of fingertip grasp: 100 mm, 120 mm and 140 mm from the palm.
- *Initial touch poses.* We tested three different distal joint angles for the initial touch: 0, 30 and 60 degrees. We note that we only include this dimension for MIMO Grasping Controller and Fixed Torque Ratio Controller test, and not for the physically underactuated hand because the distal joint angles of initial touch cannot be explicitly controlled in run time.
- *Friction coefficient.* We tested the controllers with two fingertip materials: rubber (high friction, $\mu = 1.2$) and vinyl plastic (low friction, $\mu = 0.4$).

To sum up, we swept all five dimensions and conducted 360 grasping experiments. Fig. B.1 shows the three object locations (shown as the cross-hairs), three initial touch poses (colored fingers), and the sizes of the objects relative to the gripper (orange shapes).

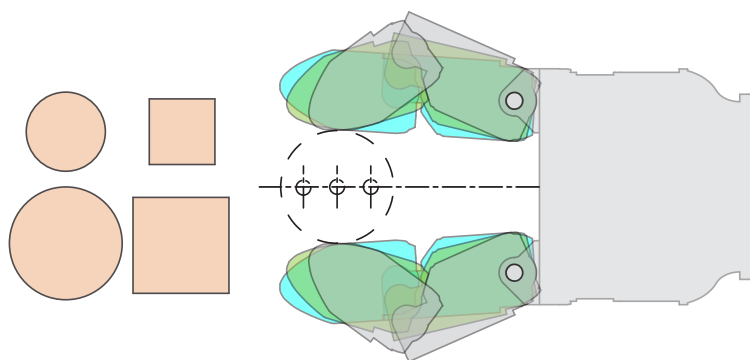


Figure B.1: Object sizes, object locations and initial touch poses.