Diss. ETH No. 24769

Computational Design Tools for Personalized Robotic Creatures and Animatronics

A dissertation submitted to
ETH Zurich

for the Degree of
Doctor of Sciences
(Dr. sc. ETH Zurich)

presented by
Vittorio Megaro
MSc in Computer Science, ETH Zurich, Switzerland
born November 9, 1989
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Bernhard Thomaszewski, co-examiner
Prof. Dr. Stelian Coros, co-examiner
Dr. Moritz Bächer, co-examiner

2017

Abstract

Mechanical creatures are becoming more and more ubiquitous in our society. While traditionally confined to industrial settings, robotic creatures and animatronics have arrived at the consumer-level in the form of electro-mechanical toys or robotic companions. Moreover, recently advanced technology such as 3D-printers, off-the-shelf servo motors or easy to program microcontrollers opened the door to a new generation of animated physical characters which can be designed according to preferences and needs of human individuals. However, even given the machinery required for the manufacturing, the design itself remains a major challenge. Unlike for digital animation, not all motions are feasible in physical reality. Therefore, this work aims at developing methods and algorithms to simplify the design of personalized robots and animatronics for both expert engineers and casual users. A guiding principle is to automate the tedious parts of the design process while providing the users with sufficient freedom to achieve their creative and functional goals. In this thesis, the problem of creating design tools that enable average users to create compelling 3D-Printed mechanical characters is divided into three sub-projects: a tool for authoring the structure and motion of rigidly articulated robots, a tool for creating 3D-printed compliant mechanisms and a tool for designing cable-driven kinematic chains and trees.

We start by introducing the fundamental models, simulation techniques and optimization methods on which the core contributions of this thesis are built. We describe how rigid bodies can be combined to produce deformable articulated characters and how to treat the different degrees of freedom in a unified manner during simulation. Moreover, we explain how to couple the unknown degrees of freedom of a simulation with the unknown design parameters, while optimizing for some user-specified goals.

The following part of the thesis introduces an interactive design system that allows casual users to quickly create 3D-printable robotic creatures. We show how our approach can automate the tedious parts of the design process while providing ample room for customization of morphology, proportions, gait and motion style.

We then extend our framework by developing a computational tool for designing compliant mechanisms. The proposed method takes as input a conventional,

rigidly-articulated mechanism defining the topology of the compliant design. Afterwards we automatically replace the different traditional joint types with parameterized flexures, to obtain a compliant replica of the input mechanism.

In the last part of the thesis, we present an optimization-based approach for the design of cable-driven kinematic chains and trees. Our system takes as input a hierarchical assembly consisting of rigid links connected together with hinges. The user also defines a set of target poses or keyframes using inverse kinematics. Our approach places torsional springs at the joints and computes a cable network that allows us to reproduce the specified target poses.

Sommario

Le creature meccaniche stanno diventando sempre più presenti nella nostra società. Tradizionalmente limitate all' industria, le creature robotiche e animatroniche sono arrivate al livello dei consumatori sotto forma di giocattoli elettromeccanici o compagni robotici. Inoltre, alcune tecnologie avanzate recentemente, come stampanti 3D, servomotori pronti all'uso o microcontrollori di facile programmazione hanno aperto le porte ad una nuova generazione di personaggi fisici e animati che possono essere progettati secondo le preferenze e le esigenze degli individui. Tuttavia, anche considerando la disponibilità dei dispositivi necessari per la produzione, la progettazione stessa rimane una grande sfida. A differenza dell'animazione digitale, non tutti i moti sono realizzabili nella realtà fisica. Pertanto, questo lavoro mira a sviluppare metodi e algoritmi per semplificare la progettazione di robot personalizzati e animatronici, sia per esperti che per principianti. Un principio guida è quello di automatizzare le parti ardue del processo di progettazione, fornendo agli utenti la libertà sufficiente per raggiungere i loro obiettivi creativi e funzionali. In questa tesi, il problema della creazione di strumenti di progettazione è suddiviso in tre sottoprogetti: uno strumento per la creazione delle strutture e dei movimenti di robot rigidamente articolati, uno strumento per la creazione di meccanismi flessibili e uno strumento per la progettazione di catene e alberi cinematici guidati da cavi.

La prima parte della tesi introduce i modelli fondamentali, le tecniche di simulazione e i metodi di ottimizzazione su cui vengono basati i contributi di questa tesi. Descriviamo come i corpi rigidi possono essere combinati per produrre personaggi articolati deformabili e come trattare i diversi gradi di libertà in modo unificato durante la simulazione. Inoltre, spieghiamo come combinare i gradi di libertà di una simulazione con i parametri di progettazione, per raggiungere gli obiettivi specificati dall'utente.

La parte successiva della tesi introduce un sistema di progettazione interattiva che consente agli utenti di creare rapidamente creature robotizzate stampabili in 3D. Mostriamo come il nostro approccio possa automatizzare le parti ardue del processo di progettazione, offrendo ampie possibilità di personalizzazione della morfologia, delle proporzioni, dell'andamento e dello stile di movimento.

Successivamente estendiamo il nostro programma informatico sviluppando uno strumento computazionale per la progettazione di meccanismi flessibili. Il metodo proposto prende come input un meccanismo convenzionale e rigidamente articolato che definisce la topologia del meccanismo. Successivamente sostituiamo automaticamente i diversi tipi di giunti tradizionali con flessioni parametrizzate, per ottenere una replica flessibile del meccanismo dato come input.

Nell'ultima parte della tesi, presentiamo un approccio per la progettazione di catene e alberi cinematici guidati da cavi. Il nostro sistema assume come input un gruppo gerarchico costituito da collegamenti rigidi collegati con cardini. L'utente definisce anche un insieme di pose o di fotogrammi utilizzando la cinematica inversa. Il nostro approccio introduce delle molle torsionali locate nelle giunture e calcola una rete di cavi che ci permette di riprodurre le posizioni specificate in precedenza.

Acknowledgments

I am deeply grateful to my advisor, Prof. Dr. Markus Gross for having given me the opportunity of being part of the Computer Graphics Laboratory, and for having provided such amazing and unique working conditions at both ETH and Disney Research Zurich. His guidance and his constructive research suggestions were extremely helpful during the great journey of my Ph.D. studies.

I will always be grateful to Prof. Dr. Bernhard Thomaszewski for everything I learned under his supervision and for always having believed in me. The priceless advice and countless discussions with him contributed to increase my knowledge, my confidence, and ultimately, to the success of my projects.

I also want to express my sincere gratitude to Prof. Dr. Stelian Coros, who accompanied me through my Ph.D. studies since the beginning. His expertise and innovative ideas were crucial for my research progress and my achievements.

I additionally want to warmly thank Dr. Moritz Bächer for always having been available to help me and clear my doubts. His precious support and advice assisted me through the last part of my studies, and eventually led me to the successful realization of my projects.

I would also like to thank my close collaborators, Jonas Zehnder and Espen Knoop, who never refused to help me in case of need. I am also very grateful to Maurizio Nitti and Alessia Marra, who always provided me with great help for the many artistic tasks of this thesis. Many thanks also to Jan Wezel, who always offered his help and fabrication expertise when approaching deadlines.

Many thanks go to all my additional collaborators, who contributed to the work presented in this thesis: Otmar Hilliges, Andrew Spielberg, David Levin, and Wojciech Matusik.

I want to thank my family and friends for their unconditional support and for always have believed in my success.

Last but not least, I want to express my deep gratitude to my girlfriend, Martina Haefeli. Her love, understanding, patience, and support, pushed me through the many days and nights preceding deadlines. The accomplishment of this thesis would not have been possible without her.

Contents

Abstract iii

Sommario v

Acknowledgements vii

Contents ix

List of Figures xiii

List of Algorithms xvii

List of Tables xviii

Introduction 1
1.1 Contributions 5
1.2 Publications 6

Related Work 7
2.1 Robotics 7
2.1.1 Design-Specific Robotics 8
2.1.2 Morphology Exploration 9
2.1.3 Digitally Fabricated Robots 10
2.2 Motion Planning 10
2.3 Compliant Mechanism Synthesis 12
2.4 Cable-Driven Mechanisms 13
2.5 Fabrication-Oriented Design 14
2.5.1 Conventional Mechanism and Physical Character Design . . 14

Mathematical Framework 17
3.1 Physics Models 17
3.1.1 Rigid Body Model 17
3.1.2 Rigidly Articulated Rigid Bodies 20
3.1.3 Compliant Connectors 23

3.2 Physics Simulation . . . 26
3.2.1 Newton-Raphson Method . . . 29
3.2.2 Quadratic Penalty Method . . . 30
3.2.3 Sequential Quadratic Programming . . . 31
3.3 The Inverse Design Problem . . . 33
3.3.1 Sensitivity Analysis . . . 35
3.3.2 The Broyden–Fletcher–Goldfarb–Shanno Method . . . 37
3.4 Conclusion . . . 39

Interactive Design of 3D-Printable Robotic Creatures 41
4.1 Introduction . . . 41
4.2 Overview . . . 43
4.3 Motion Plan Generation . . . 46
4.3.1 Constraints . . . 47
4.3.2 Motion Style Objectives . . . 49
4.3.3 Optimization . . . 50
4.4 Generation of 3D Printable Mechanical Structures . . . 51
4.5 Results and Discussion . . . 54
4.5.1 Design Interface & Workflow . . . 54
4.5.2 Validation . . . 57
4.6 Limitations and Future Work . . . 58

A Computational Design Tool for Compliant Mechanisms 61
5.1 Introduction . . . 61
5.2 Computational Model . . . 63
5.2.1 Simulating Compliant Mechanisms . . . 63
5.2.2 Parameterizing Compliant Joints . . . 64
5.2.3 Generating Link Geometry . . . 66
5.3 Design Optimization . . . 67
5.3.1 Motion Tracking . . . 67
5.3.2 Lateral Stability . . . 67
5.3.3 Actuation Requirements . . . 68
5.3.4 Avoiding Collisions . . . 68
5.3.5 Preventing Material Failure . . . 70
5.3.6 Optimization . . . 74
5.4 Results . . . 75
5.5 Conclusions . . . 82

Designing Cable-Driven Actuation Networks for Kinematic Chains and Trees 85
6.1 Introduction . . . 86
6.2 Observations . . . 87

6.3 Simulating Cable-Driven Trees 89
6.4 Placing and Sizing Cable Networks 92
6.4.1 Identifying the Network Topology 93
6.4.2 Refining the Cable Network 94
6.5 Fabrication Considerations 96
6.6 Results 100
6.7 Conclusion 103

Conclusion **105**
7.1 Future Directions 108
7.1.1 Project-Related Research Directions 109
7.1.2 General Research Directions 111
7.1.3 Final Considerations 114

References **115**

List of Figures

3.1 Transformations for global and local coordinate systems. 18
3.2 Four types of connectors. 20
3.3 Rod centerline and its degrees of freedom. 24
3.4 Simple example showing six rigid bodies, connected through two hinge joints and four torsional springs. Gravitational forces are acting on the rigid bodies which are free to move. 27
3.5 Target configuration, with rb_3 being horizontal and at a hight of $\tilde{y}$. In transparency, the rest configuration before applying gravity. . . . 34

4.1 Digital designs (left) and physical prototypes (right) for our *Ranger* (top), *Bobby* (middle) and *Predator* (bottom) designs, fabricated using 3D-printing and off-the-shelf servo motors. 42
4.2 The footfall pattern indicates which leg is in stance (white) or in swing (red) mode. In this example, the user interactively adjusts the footfall pattern such that three legs are always in stance mode. . 43
4.3 Snapshot of the design interface. *Left*: the design viewport with the footfall pattern graph. *Right*: the preview window showing the center of pressure of the robot (green) and the support polygon (red). 44
4.4 The motion plans generated by our framework consist of trajectories for the center of mass (green), feet (blue), and corresponding full-body poses for the robotic creature (left). An inverted pendulum model is employed to obtain a relationship between the center of pressure and the center of mass (right). 47
4.5 Convergence plot showing the value of the objective function while optimizing all parameters at once (blue) versus a two-step optimization scheme (red). 52
4.6 Generating 3D-printable geometry. 53
4.7 Six robotic creatures designed with our interactive system: one biped, four quadrupeds and one five-legged robot. 55
4.8 *Salamander*: our framework automatically generates swaying tail and spine motion, leading to a natural-looking gait. 56

5.1 **Conventional vs. compliant hinge.** We replace conventional joints (left) with a single or several flexures (right). 63

5.2 Parameterizing a compliant hinge with two offset flexures. 65
5.3 Compliant ball-and-socket (left) and universal joint (right). 67
5.4 A collision between a flexure and a link (*left*) is resolved by reshaping the corresponding geometry (*right*). 69
5.5 Bending induces stretching and compression away from the centerline. 71
5.6 Twist induces pure shear deformation about the rotation axis. . . . 73
5.7 **Chebyshev Linkage.** Simulated (top row) and fabricated (bottom row) compliant Chebyshev linkages that were optimized with different objective terms as indicated. Starting from the red trajectory after initialization, the trajectory (in blue) of a marker on the original assembly is recovered for all combinations of objectives. However, only with f_{fail} active (right most assembly), the compliant mechanism lasts. 75
5.8 **Jansen's Linkage.** After replacing conventional hinges with compliant flexures, the end effector trajectory (red) deviates significantly from the original trajectory (in black). After optimally placing, orienting, and sizing the flexures, we can recover the lower portion of the trajectory (in contact with the ground) to a large degree (in blue). 78
5.9 **Eye Mechanism.** Compliant eye mechanism at human-scale (top left), at Dime-scale (bottom left and top right), and a side-by-side comparison (bottom right). 79
5.10 **RC Car.** Remote controlled car featuring a fully compliant steering mechanism. In addition to preserving the steering functionality of the input mechanism, this design was optimized to sustain its own weight. 80
5.11 **Compliant Hand.** A compliant finger mechanism is replicated and assembled to create a fully operational hand. Side and bottom views are shown on the left and in the middle. On the right we show the hand performing a teleoperated grasping task using cables for actuation. Thanks to restoring forces from the compliant flexures, the hand can be actuated using a single cable per finger. . 81
5.12 **Dragon.** Full dragon (left) and close-up onto the wing mechanism (right). The wings of the dragon are two identical but mirrored versions of a spatial compliant mechanism. This example exhibits large deformations induced by twist, emphasizing the need for a twist-aware objective for preventing material failure. 82

6.1 Our computational tool for designing cable-driven kinematic chains and trees (left) enables artists and hobbyists to size and place a cable network (middle) in order to closely match a set of target poses or keyframes using co-optimized control forces (right). 87
6.2 The input to our system is a kinematic assembly consisting of rigid links, jointed together at their ends (left). The red link is fixed. A user first specifies target poses (dotted link contours) and adds torsional springs to the hinges (green circles). A set of cables is then optimized (middle) to hit the user-specified target poses as closely as possible when actuated (right). The cables are attached at one end (red circle) and are routed with pulleys (black circles) to the fixed link. Elastic springs are added to the hinges (black spirals) to define the rest configuration as the zero energy state. 88
6.3 The assembly is in equilibrium if the torque $\tau_s(\theta)$ of the torsional spring with joint angle θ equals the applied torque hf with signed moment arm $h = \frac{\det[\mathbf{u}-\mathbf{x},\mathbf{v}-\mathbf{x}]}{\|\mathbf{u}-\mathbf{v}\|}$. 89
6.4 Torques $h_i f_i$ from several cables can affect the position of a single joint (left) and a single cable can affect the positions of several joints (right). 90
6.5 To describe the kinematics of our hierarchical input, we use a recursive definition similar to the one used for rigs in character animation (left): this example consists of two hinges (at nodes 1 and 3) and three components (fixed link in red, link consisting of two segments in light grey, "leaf" link in dark grey). The topology is uniquely defined by the function $r = \{(0,o),(1,0),(3,1),(4,3),(2,1)\}$. We express routing points in local, per-segment frames $\left[\mathbf{d}^\perp, \mathbf{d}\right]$ with coordinates $(p^\perp, p)$ (right). . . 92
6.6 Dependent on the sequence of actuation, the symmetric single-joint assembly with two cables (left) ends up in two completely different configurations: the force f is first applied to the right, then to the left cable (top row). If this sequence is reversed (bottom row), the assembly tilts to the left instead. 94
6.7 We use two instead of a single routing point per cable and link (top row). While this adds to the complexity of the manual assembly task (the pulleys almost double per cable), it has a an important advantage when generating geometry for the final assemblies: we can rotate oriented triangles $(\mathbf{x}, \mathbf{u}, \mathbf{v})$ about the hinge axis without changing the torque equilibrium in any given pose. 96
6.8 A characterization experiment reveals a linear but asymmetric behavior with k_{pos} in blue and k_{neg} in red. 97

6.9 We compensate for gravity at a particular joint $\mathbf{x}_j$ by holding all other torsional springs fixed (left), adding up gravitational torques of rigid links (right, top), torsional springs (right, middle), and pulleys (right, bottom) on the path from j to the leaves. 99
6.10 We account for the finite dimension of pulleys by offsetting all moment arms h_j and h_{j+1} by the constant pulley radius r_p. To ensure that cables do not detach during actuation, we wrap them once around the pulleys (see cable in blue). 100
6.11 We validate our cable network optimization on the lower body of our Fighter character on two target poses (bottom, top) by controlling the cable forces with standard Newton meters. The simulated poses (left) match the physical poses (right) well. 101
6.12 Our gripper is optimized to pick up two T-shapes when one cable is actuated, and a heart-shape if only the other is actuated. 102
6.13 Our animatronic hand can perform a wide range of gestures. Its thumb is operating in a different plane than the remaining fingers and the wrist. 103

List of Algorithms

List of Tables

4.1 An overview of the complexity of robot designs we created with our system. 58

5.1 Computation times for optimizing the Chebyshev linkage using different objective terms as indicated in the top row. Simulation is the dominant part in both cases, whereas the evaluation of the objective terms is negligible in comparison. 78

5.2 **Statistics.** The columns (from left to right) list the number of degrees of freedom, the number of iterations required for convergence, the average iteration cost, as well as the mean and maximum error for the motion tracking objective. 83

6.1 We summarize the complexity (number of links, joints, specified targets, cables, and pulleys) of our three example assemblies. Our two-step optimization reduces the initial number of cables x to y after the first, then to z after the second step, formated in column "cables" using z (x, y). 100

CHAPTER 1

Introduction

Crafting and animating compelling digital characters is the vision of the modern computer graphics field. Fueled by the demand of ever more visually appealing and realistic characters in movies and games, a large number of tools to design, edit, animate, and render virtual characters were developed in the past decades. Undoubtedly, the presence of convincing digital characters in the movie and game industry is important. However, many applications also require the physical realization of such characters. Be it as attractions in amusement parks, complex electro-mechanical toys, or robotic companions; mechanical characters have a broad variety of applications in entertainment and beyond.

The transition from the digital to the real world is a tedious process. The characters that once were digitally animated, suddenly start to have mass, and material properties, and their behavior is governed by complex physical laws. The same character, which was visually plausible, is not functional in the real world. An similar fate is also reserved for the established design tools, which have been developed for creating animations. These tools are not useful when dealing with tangible characters, where the artistic freedom is restricted by physical reality. Moreover, the complexity of the design process largely increases due to the high number of parameters, all of which can potentially affect the shape and behavior of the final result. The effect of a change in parameters is often unintuitive due to nonlinearities embedded in the underlying physics laws which govern the real world. The obstacles introduced by physical limitations strengthen the need for a dedicated set of tools, tailored for the the design of manufacturable characters.

There are many successful examples of functional robots and animatronics in industry and research, but their design, structure, and control policies are tightly coupled. A change in any of these designs, either in hard- or software, will inevitably propagate in a modification of the remaining components. A single robotic or animatronic design may require a joint effort of experts for many years, resulting in a costly product realization process in terms of both material and personnel. The reason behind the expensive process is a high number of time-consuming iterations required to converge towards a functional and efficient design. The iterations in the design process are based on a trial and error process, which most of the time requires a long implementation time when physical creatures are involved.

Some tools were developed with the intent of predicting the result of editing the design parameters, thus alleviating the tedious task of realizing a multitude of physical prototypes. Specifically, physics simulations can provide reliable results in a fraction of the time needed by manufacturing processes. Simulation is, therefore, well suited to be the back-bone for such tools. However, the design parameters of physical characters lie in a high dimensional space and simulation alone is not enough to guide novice and expert users towards their goals. Conversely, a more intuitive system would be one with the capability of letting the user directly express some high-level goals, leaving the task of finding the parameters that best satisfy them to the system. With this purpose in mind, this thesis proposes different tools which automatically search the design space to find a set of parameters leading to mechanical structures which meet a user's requirements.

The demand for such systems is not limited to a handful of experts. Our society is turning towards a direction of customization of different types of goods. From an individual's car or furniture to one's clothing or jewelry, personalization prooved to be an important need for a consumer, also demonstrated by the fast growth of craft stores and the 3D printing industry. The need of being able to realize the concepts a user has in mind, asks for intuitive tools with the ability to bring an individual's ideas to life. The inspiration for this thesis is based on the potential of personalization of mechanical creatures. We, therefore, aim at developing systems that provide different interfaces and intuitive tools for easy customization, while the system takes care of the functionality.

One of the goals for roboticists is to be able to realize robots and animatronics which have the same complexity and characteristics of living creatures. Complex tasks such as walking, galloping, swimming, flying besides grasping, jumping, and throwing, are achieved with grace and elegance in the animal kingdom. Bones, muscles, and tendons cooperate to perform a par-

ticular action, being the result of a harmonious combination of stiffness and elasticity. However, mechanical character realized by humans struggle in performing complex tasks as efficiently and smoothly as living creatures. One of the reasons being that when roboticists and engineers design robots or moving machines, they commonly use very stiff or rigid parts connected through hinges. Elastic deformation provides advantages such as adaptation to unpredicted environmental changes, shock absorption, and energy storage. However, in the history of robotics, piecewise rigid creatures are omnipresent while compliant designs are far rarer. The reason behind it is that, even if compliant structures can integrate different functions into fewer parts, they can be much harder to design. The coupling between kinematics and actuation torque has an unintuitive behaviour, and material failure can occur when stress is concentrated in a single point.

Thanks to the development of new materials and the drastic increase of computational capabilities, developers and researchers have increased our ability to design and analyze mechanisms that can combine rigidity and compliance. This thesis addresses the long-term goal of empowering experts and casual users to design, manufacture, and maintain robots and animatronics capable of performing complex tasks by mimicking nature. The obstacle in the way of this goal must be divided into subproblems that need to be treated individually, due to their complexity. This thesis is thus split into three different research directions: the personalization of stable walking motions for rigidly articulated robots, the creation of compliant mechanisms, and the design of cable-driven articulated structures. By solving these three problems, the gap between technology and makers can partly be filled.

We start by addressing the problem of generating stable locomotion for legged robotic creatures. Different techniques can be used to design a system that allows the user to specify high-level motion goals for walking creatures. The property shared among them is that they try to predict which sequence of actions would lead to a specific target goal. One of the distinctions between the different strategies is how far in time they try to foresee a certain event to happen. One can choose between a global and local overview of the motion plan. By using a global overview (long-horizon), the system can optimize the entire action plan simultaneously. By opting for a local plan (short-horizon), the system optimizes for the immediate actions to take. Both come with advantages and disadvantages. A long-horizon optimization enables early decisions making since the full plan is available. However, this comes with the drawback that every optimization step takes a long time to compute, due to a large number of space-time variables that need to be taken into account. A short-horizon control system needs to predict fewer time-steps, leading to an improvement in computational performance. However,

such a system has the incapability of finding solutions which are globally optimal. By analyzing the problem and the requirements of an easy-to-use interface that gives control to the user, we propose to use a long-horizon method. Such a decision allows us to automatically compensate for design choices which can bring a creature to fall. Since inexperienced users regularly make design choices which might be poor in terms of locomotion stability, we need to safeguard against them. In Chapter 4 we describe how to use a long-horizon motion planner, hence avoiding imbalance, while keeping computational performances high and accomplishing interactivity.

Afterwards, we focus on the challenges posed by the design of compliant mechanisms. Methods to develop compliant mechanisms have been studied for a long time in the field of mechanical engineering. However, such methods assume small deformations and focus on specific functionalities, making them unsuited for our task. When the goal is to create compliant components for a walking robot, propulsion and stability need to be considered. Therefore, the mechanisms we design need to reach complex motions while taking into account the structure's relevant properties. Such properties can be the work required to actuate a mechanism, the stiffness along and orthogonal to the direction of movement, or the material failure. We seek the development of an interface that allows casual users to design compliant mechanisms that exhibit a vast spectrum of motions and have properties that make them well-suited to be used as robotic or animatronic components. Our tool leverages previous research conducted in rigidly articulated characters, by taking as input a traditional assembly and convert it into a function-preserving compliant design. By proposing a conversion tool, we allow the use of previously-realized content available on-line.

Finally, we transition to motion propagation using cables. Taking inspiration from the tendons found in animals, we investigate remote controlling using a network of cables. The system we propose takes as input a virtual character together with a corresponding animation. It, then, automatically computes the set of cables routings, together with the relative forces needed to reproduce the input animation on a physical character. The computed cable-network is optimized for reduced complexity and minimal torque required to animate the final physical character. The more cable introduced to the structure the more freedom and control the character has; a trade-off between motion fidelity and design complexity is left to the user. The resulting structure is built from rigid components representing the skeleton provided as input, where the virtual joints are replaced with torsional springs. Differently from traditional articulations, the added springs favor a rest configuration, which allow us to actuate the character using cables. We then route a cable-network through the character's structure with the use of pulleys

and remotely actuate the cables with either servos or levers. Compared to techniques which comprehend the use of one actuator per joint, our method enables the creation of physical animated characters which are compliant, lighter, and more energy efficient.

1.1 Contributions

This thesis makes the following main contributions:

- A fast optimization-based solution that generates stable motions for legged robots of arbitrary designs. To warrant feedback at interactive rates, we depart from conventional space-time approaches, leveraging a geometric view of trajectory optimization. Tightly integrated with the optimization, an intuitive set of interactive tools allows the user to design and edit the morphology, proportions, gait and motion style of a robotic creature. In order to translate the digital designs to the physical world, we automatically generate geometry for the mechanical structure of the robot such that it can be fabricated using 3D-printing and off-the-shelf servo motors.
- An assistive tool that enables non-expert users to leverage the advantages of compliant mechanisms while avoiding common pitfalls. Our method takes as input a conventional, rigidly-articulated mechanism defining the topology of the compliant design. This input can be both planar or spatial, and we support a number of common joint types which, whenever possible, are automatically replaced with flexures drawn from existing catalogs [Smith, 2000; Howell et al., 2013]. As the technical core of our approach, we describe a number of objectives that shape the design space in a meaningful way. The list of objectives includes trajectory matching, collision avoidance, lateral stability, resilience to failure, and minimizing motor torque. To represent our flexures, we use discrete elastic rods [Bergou et al., 2008; Bergou et al., 2010], extending the model to predict *volumetric* stresses. Optimal designs in this space are obtained as solutions to an equilibrium-constrained minimization problem that we solve using a variant of sensitivity analysis.
- The first algorithm to design artist-controlled, fabricable cable-driven kinematic chains and trees. The method uses a reduced coordinate formulation that couples actuation forces to joint angles and routing points, enabling quasi-static simulation and co-optimization

of routing points and control forces. A two-step optimization approach is put in place to size and place a complex cable network to match a set of user-specified target poses or keyframes in a least squares sense.

1.2 Publications

This thesis is based on the following peer-reviewed publications:

V. MEGARO, B. THOMASZEWSKI, M. NITTI, O. HILLIGES, M. GROSS, and S. COROS. Interactive Design of 3D-printable Robotic Creatures, *Proceedings of ACM SIGGRAPH Asia (Kobe, November 2 - November 5, 2015), ACM Transactions on Graphics*, vol. 34, no. 6, pp. 216:1–216:9.

V. MEGARO, E. KNOOP, A. SPIELBERG, D. LEVIN, W. MATUSIK, M. GROSS, B. THOMASZEWSKI and M. BÄCHER. Designing Cable-Driven Actuation Networks for Kinematic Chains and Trees, *Proceedings of the 2017 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (Los Angeles, United States, July 28-30, 2017)*, pp. 15:1–15:10.

V. MEGARO, J. ZEHNDER, M. BÄCHER, S. COROS, M. GROSS, and B. THOMASZEWSKI. A Computational Design Tool for Compliant Mechanisms, *Proceedings of ACM SIGGRAPH (Los Angeles, United States, July 30 - August 3, 2017), ACM Transactions on Graphics*, vol. 36, no. 4, pp. 82:1–82:12.

CHAPTER 2

Related Work

Lately, the field of Digital Fabrication and Computational Design has acquired a lot of attention by the Computer Graphics community. While being of recent interest, this area of research finds a place at the intersection of various, long studied, disciplines. Among others, we find for example computer graphics, mathematics, engineering, robotics, and mechanics. Tools oriented in the direction of computational design frameworks may borrow and extend discoveries and technologies from the above-listed disciplines. The employment of existing techniques can span from the representation of hierarchical articulated characters to physics simulation or the preservation of certain mechanical properties. This chapter provides an overview of the current work related to the research fields connected to this thesis.

2.1 Robotics

While the study of the various genres of robots started decades ago, the field of robotics remains a hot topic among both the academic community and industry. The creation of the first intelligent robot and the introduction of industrial robotic arms in the assembly lines dates back to the early seventies [Waseda University, 1972; KUKA, 1974]. From that point on, engineers and robotiscists tried to make robots more efficient [Makino, 1982; Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1989], capable of different functions [Massachusetts Institute of Technology, 1995], and able to mimic [Honda Motor Co., Ltd., 1993; Honda Motor Co., Ltd., 1996; Honda Motor Co., Ltd., 1997; Honda Motor

Co., Ltd., 2000; Honda Motor Co., Ltd., 2005] and interact [Sony Corporation, 1999; Sony Corporation, 2003] with humans.

2.1.1 Design-Specific Robotics

The ability to move is a desirable skill for robots. The capability of robots to transport themselves from place to place makes them more versatile, allowing a single robot to fulfill different tasks in different locations. While humans can locomote at ease in various environments and different contexts, design a control policy for legged robots is an arduous task. In recent years a lot of effort was put into the study of locomotion strategies for different types of robot structures and terrain features. The Massachusetts Institute of Technology has proposed a quadruped, the MIT Cheetah 1 [Seok et al., 2013], which was built with the goal of having a highly efficient legged robot with low cost of transport per unit of mass. To reach such a goal, Seok et al. introduced and implemented four design principles that minimize the energy loss with the environment: the use of high torque density motors, low impedance transmission, energy-regenerative electronics and design architecture that reduces leg inertia. The Cheetah 1 can perform fast running motions while keeping its energy consumption low. However, the robot comes with the limitation of being held in a plane, constraining it to the inability of full three-dimensional movements. The next-generation MIT robot, the Cheetah 2 [Park et al., 2014], is a full 3D robot, which employs actuator torques on the leg and shows major differences in the control algorithm. While the Cheetah 1 control policy is based on impedance and position tracking, its more recent model uses impulses to achieve the desired stance and total stride duration. The skills of the Cheetah 2 were enhanced by Park et al. [2015; 2015], where the control system components were designed to enable the quadruped to run at a wide range of speeds and autonomously jump over obstacles up to 40 cm in height. The Massachusetts Institute of Technology also introduced the MIT Super Mini Cheetah (SMC) [Bosworth et al., 2015], a quadruped below 10 kg in mass and below 10k dollars in cost. The small robot can control the force and impedance between each foot and the ground, allowing for performances such as walking, turning, pronking and jumping. The type of terrain on which the robot operates affects the ground impedance and surface friction, making the task of modeling the foot-ground interaction a priori challenging. Bosworth et al. [2016] present experimental data of the SMC hopping on hard and soft grounds. They show that controllers tuned for each surface perform better for each particular surface type.

Other quadrupedal robots were presented in the robotics community. Hutter et al.[2012] introduced StarlETH, a robot actuated with torque-controlled, highly compliant series of elastic actuators, designed to study fast, efficient and versatile locomotion. Gehring et al. [2013] presented a control framework for quadrupedal robots able to locomote using several gaits and tested the result on the StarlETH. Semini and colleagues [2011] developed a versatile hydraulically powered quadruped robot (HyQ). The robot was used to study motions such as running and jumping, but also careful navigation over rough terrains [Winkler et al., 2015]. At EPFL another Cheetah robot was developed [Rutishauser et al., 2008], featuring three-segment pantographic legs with passive compliant knee joints. The Cheetah legs [Sproewitz et al., 2009] are light-weight, retractable and actuated using RC servo motor mounted proximal where the force transmission is achieved using a Bowden cable system. Tuleu et al. [2011] present two model-free locomotion control approaches for the EPFL Cheetah: an open loop central pattern generators, and an open and closed-loop dynamical moment primitives. They show that thanks to the passive-compliance of the legs, the robot requires less control effort in the knee joints. The Cheetah-cub [Spröwitz et al., 2013] was also built upon passive compliant legs with a dedicated open-loop locomotion controller. The experiments show that both in simulation and reality the robot shows self-stabilizing properties.

2.1.2 Morphology Exploration

The problems addressed by this thesis are closely related to the field of robotics, where a long-standing goal is to automate the design of robotic and mechanical systems based on high-level functional specifications. While in the aforementioned sequence of work focuses on develop, study, and improve robots with specific designs and skills, our aim is to help casual and expert users creating creatures and characters with a variety of shapes and functionalities. Thanks to the capability of exploring the vast continuous and discrete space of robot morphologies, Sims' work [1994] on evolving virtual creatures, inspired the investigation of a variety of evolutionary methods that aim to co-design a robot's structure and control inputs [Leger, 1999; Lipson and Pollack, 2000; Auerbach et al., 2014; Methenitis et al., 2015], and this effort continues today [Ito et al., 2016; Konsella et al., 2017]. In Chapter 4, although having overlapping goals, rather than relying on stochastic algorithms, we focus on putting the user in the loop with a set of easy-to-use editing tools that provide control over the creative process.

2.1.3 Digitally Fabricated Robots

Digital fabrication and its new prototyping methods open the door for a new generation of robots which feature fast fabrication, assembly, and reconfiguration. Cutkosky et al. [2009] examine different materials and structures that contribute to the performance of running and climbing. Combining stiffness and compliance, they make the robots more robust and less demanding to control than traditionally designed robots. Researchers have investigated the venue to self-assembling and self-reconfiguring robots, driven by the advantages such as easier transportation and inexpensive deployment. Spröwitz et al. [2014] propose a modular robotic system that, employing physical latches, allows the robot to self-reconfigure and locomote. Instead, Sun et al. [2015] rely on the pressure change to actively alter the shape of planar self-assembling robots. Other methods that allow users to create origami-inspired, foldable robots have been proposed [Mehta and Rus, 2014; Mehta et al., 2014]. The design system introduced in Chapter 4 leverages similar digital fabrication techniques and easy-to-obtain hardware. However, we enable users to freely explore a vast space of legged robot designs: two or more legs, point or area feet and articulated spines are all handled in a unified manner by our framework.

2.2 Motion Planning

To generate motions for the physical creatures presented in Chapter 4 we draw inspiration from the rich literature on this topic from both computer animation and robotics. In particular, the motions we generate are similar to those used by sophisticated robots like Boston Dynamic's *Little Dog* [Neuhaus et al., 2011]. A property of the control solutions reported in the robotics literature are usually linked intimately to the robots that they are designed for. We depart from coupling and rely on a fully automated approach to generate optimal walking motions for creatures of arbitrary morphologies. The trajectory optimization method we use for this purpose is inspired by a well-known class of animation methods based on space-time constraints [Witkin and Kass, 1988] which continue to be actively investigated to date [Wampler and Popović, 2009a; Mordatch et al., 2012; Zimmermann et al., 2015; Park et al., 2016]. However, motivated by the need to develop a computationally efficient system that supports interactive design, as well as by our goal of creating robotic creatures capable of walking stably using only commodity hardware, in Chapter 4 we develop a model that presents important differences. Briefly, rather than modeling

full system dynamics and assuming idealized joint torque actuators, we directly use joint angles as control variables and explicitly enforce a dynamic stability objective. In order to achieve locomotion stability, we analyze the center of mass trajectory, and we constrain the position of the Zero Moment Point (ZMP) to be inside of the convex hull of the feet in contact with the ground. This technique to perform stable walking motion is not new to the robotics community, and it is identified as the *Zero Moment Point* stability criterion. Even though the concept of ZMP has been known for a long time [Vukobratović and Borovac, 2004], it is still widely-used, and other criteria based on it remain of crucial importance [Pratt et al., 2006; Pardo et al., 2016; Winkler et al., 2017].

Machine learning algorithms were conceived during the last years of the fifties [Samuel, 1959], but thanks to the recent advance of general purpose GPU and the resulting increase in parallel computational power, machine learning was applied to all sort of applications. During the past years, an increasing number of papers tackled motion planning and locomotion stability problems using machine learning techniques. Due to the high dimensionality of the parameter space in humanoid motion, Stulp et al. [2010] propose to use path integrals [van den Broek et al., 2008] to derive a probabilistic reinforcement learning approach which is more numerically robust and requires less state sampling during the high dimensional space exploration. Geijtenbeek et al. [2013] presented a muscle-based control method for simulated bipeds. In their work both the muscle routing and control parameters are optimized, which yields a generic method that supports a variety of creatures. Agrawal et al. [2013] propose an optimization framework to generate diverse variations of physics-based character motions. They describe actions to be underconstrained and optimize for diversity using the Covariance Matrix Adaptation (CMA) method. The transfer of human movements to humanoid robots was also investigated. To obtain stable motion transfer, Vuga et al. [2013] base their controller on an approximate model of the robot dynamics combined with a model-free reinforcement learning to improve accuracy and balance. Different methods to employ Neural Networks to learn control policies have been proposed to teach two-dimensional and three-dimensional quadrupeds to run and jump over flat and rough terrains [Gay et al., 2013; Peng et al., 2015; Peng et al., 2016; Peng et al., 2017a; Peng et al., 2017b]. The use of deep networks allows for model-free feedback controllers which once trained can respond with less delay between sensing and acting. In Chapter 4 we focus on interactivity, which is achieved without any prior knowledge of the character shape and morphology. Integrating a learning-based approach into our system would be non-trivial, due to the

computationally heavy training step that would be required every time the user changes the robot structure.

2.3 Compliant Mechanism Synthesis

Compliant mechanisms achieve motion through elastic deformation and have been the subject matter of a large body of prior art. An exhaustive review is beyond the scope of this thesis and we focus our discussion on mechanism synthesis, referring the interested reader to excellent books [Smith, 2000; Howell, 2001; Howell et al., 2013] and a recent survey [Albanesi et al., 2010] on the subject matter.

Early synthesis techniques use structural optimization [Kota and Ananthasuresh, 1995] that bear the advantage of supporting topological changes. While initially targeting planar mechanisms, follow-up work [Frecker et al., 1997] addresses the design of compliant spatial mechanisms. Wang et al. [2009] propose a stiffness matrix representation and target synthesis of planar mechanisms through topology optimization. Like these works, in Chapter 5, we account for kinematic and structural requirements in the design process. However, we base our formulation on elastic rods [Bergou et al., 2008; Bergou et al., 2010] instead of beam deflection theory [Kota and Ananthasuresh, 1995; Frecker et al., 1997] or small-displacement analysis [Wang and Chen, 2009], significantly increasing the prediction quality for large deformations and non-linear behaviors.

Related to our effort is pseudo-rigid-body replacement [Howell and Midha, 1994]. Like in Chapter 5, a designer starts with a conventional mechanism designed to accomplish a particular task. Conventional joints are then replaced with torsional springs to provide an expert designer with approximate performance estimates. Conventional joints are then replaced one-by-one, adjusting torsional spring constants of the pseudo-rigid-body model. In Chapter 5 we automate this process and avoid the tedious manual tuning of spring constants.

To provide the compliant mechanism designer with guidelines on the placement and orientation of flexures, Hopkins et al. [2010a; 2010b] developed the concept of Freedom and Constraint Topology (FACT) with theoretical underpinnings in screw theory [Huang et al., 2013]. We follow these guidelines when replacing conventional joints, favoring compliance in desired and stiffness in undesired directions.

Limaye et al. [2012] propose a kit with flexible beams and connectors together with an analysis and synthesis approach to design and hand-

assemble compliant mechanisms. While restricted to the plane, their synthesis enables rapid design iterations of monolithic compliant mechanisms. The design space, however, is restricted, supporting only beams of discrete length and a single connector type.

2.4 Cable-Driven Mechanisms

Given the desire to produce natural looking motions, computer graphics has actively explored the efficient simulation of cable-driven (also referred to as tendon-driven) systems [Sueda et al., 2008; Sueda et al., 2011]. Furthermore, biomechanics literature has done extensive work in the efficient simulation of tendon-driven biomechanics (for instance OpenSIM [Delp et al., 2007]). There has recently been work on generating key-framed animations by applying both black-box and white-box control schemes to these systems [Sachdeva et al., 2015]. The design system described in Chapter 6 features a simulator for cable-driven mechanisms, but rather than previous fully dynamic simulations [Sueda et al., 2008; Sueda et al., 2011; Sachdeva et al., 2015], we rely on a quasistatic assumption, allowing us to avoid costly time integration. While there has been work on fabricating cable-actuated folding surfaces [Kilian et al., 2017], this prior work focuses on folding origami shapes between open and closed positions, not the co-optimization of control and design for the motion of cable-driven linkages that we attack in Chapter 6.

Finally, cable-driven mechanisms and biomechanical modeling have received much attention in robotics. However, many of the works in this field are targeted towards manually designing or learning controllers for specific mechanical designs such as spines [Mizuuchi et al., 2002], tentacles [Camarillo et al., 2008; Rucker and III, 2011], arms, hands and fingers [Rooks, 2006; Ozawa et al., 2009; Borghesan et al., 2010; Sawada and Ozawa, 2012; Ma et al., 2013; Mitsui et al., 2013; Roy et al., 2013; Grebenstein, 2014; Whitney et al., 2014] or parallel manipulators [Fang et al., 2004; Behzadipour, 2005]. Surgical robots can also benefit from cable-driven actuation due to the fact that their end-effectors consist of small, surgical apparatuses [Hannaford et al., 2013]. The idea of a fixed design and optimizing for control parameters also extends to more esoteric designs such as continuum manipulators [Camarillo et al., 2008]. Control strategies have even been developed for full-body, cable-driven robots such as the ECCEROBOT [Potkonjak et al., 2011], RoBoy [Rob, 2016] and Sparky [Spa, 2016]. More recent work uses genetic algorithms to optimize cable tensions and cable angle to generate

single, periodic trajectories for fixed, small DOF (2-3) designs [Bryson et al., 2016].

Unlike previous work, in Chapter 6 we focus on designing cable-driven linkages that artists can control. To this end, we provide an algorithm that co-optimizes both control parameters (cable forces) and design parameters (number of cables, cable routing points) to match a number of artist-supplied key frames or target poses for kinematic chains. Crucial for animation, our system allows an artist to control the pose of the entire linkage, not just end effector position. This differentiates it from the control only, end effector-driven approaches prevalent in robotics.

2.5 Fabrication-Oriented Design

Driven by technological advances in digital fabrication, the graphics community has devoted a significant effort on the computational design of physical artifacts, characters and structures. Proposed techniques have targeted plush toys [Mori and Igarashi, 2007] besides structurally-sound [Stava et al., 2012; Umetani and Schmidt, 2013; Zhou et al., 2013; Lu et al., 2014], stably standing [Prévost et al., 2013], spinning [Bächer et al., 2014], or floating [Musialski et al., 2015] 3D printable models. Other examples include design systems for physically-valid furniture pieces [Lau et al., 2011; Umetani et al., 2012], free-form gliders with optimized aerodynamic properties [Umetani et al., 2014] and prototypes that test and validate the functionality of finished products [Koo et al., 2014]. Others have tackled the problem of creating composition of wire meshes [Garg et al., 2014], thermoforming of shapes [Schüller et al., 2016], and wind instruments [Umetani et al., 2016] have been tackled. Similar to the flexures described in Chapter 5, fexible rod [Pérez et al., 2015], filigree [Chen et al., 2016], ornamental curve [Zehnder et al., 2016] and stable rod [Miguel et al., 2016] structures are designed to last when bent. This thesis shares the same high-level goal as these works: empowering casual users in creating complex physical artifacts without requiring domain-specific knowledge.

2.5.1 Conventional Mechanism and Physical Character Design

Several techniques that aid the non-expert with the understanding [Mitra et al., 2010], design [Zhu et al., 2012; Coros et al., 2013; Ceylan et al., 2013; Thomaszewski et al., 2014; Megaro et al., 2014; Hergel and Lefebvre, 2015; Lin et al., 2016], editing [Bächer et al., 2015], and recovery [Koo et al., 2014] of

complex mechanical assemblies have recently been proposed. These methods focus on the design of passive mechanical structures, such as linkages and gears, that propagate the motion of one input motor to the entire system. In Chapter 5 we leverage these existing approaches, where the mechanical assemblies output by these work serve as input to our framework. Other methods were proposed to create physical characters, for instance, the design systems proposed by Bächer et al. [2012] and Calì et al. [2012] can be used to design 3D printable characters with functional joints. Skouras et al. [2013] proposed a method to automatically optimize the internal distribution of material parameters in order to control the ways in which 3D printed characters deform when subjected to external forces.

CHAPTER 3

Mathematical Framework

In this chapter, we describe the fundamentals concepts to model, simulate and optimize the components constituting the assemblies and characters presented in the rest of the thesis. We first describe how to model degrees of freedom of rigid bodies, joint-connectors, servos, springs, and elastic rods (Section 3.1). We then discuss generic formulations to simulate physical systems and several techniques to numerically minimize energy functions (Section 3.2). Finally, we conclude the chapter with some procedures to solve what is known as the *inverse design problem* (Section 3.3). The latter is the problem that requires finding the design parameters for which, after a *forward* simulation, the state of the system will minimize some problem-specific objectives. Even though most of the ground concepts are shared between forward and inverse problems, the latter is typically harder to solve. A dedicated discussion about the topic can help the reader in the understanding of the design methods proposed in this thesis.

3.1 Physics Models

3.1.1 Rigid Body Model

Rigid bodies, as the name suggests, are assumed to not deform under external forces. Thanks to this property, we can represent their state using a translation vector $\mathbf{t} \in \mathbb{R}^{3x1}$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3x3}$. Typically $\mathbf{t}$ denotes the displacement of the center of mass from the origin O but, in general, can indicate the shift of any arbitrary point of the rigid body. Instead, $\mathbf{R}$ encapsulates the information of the orientation of the rigid body with respect to

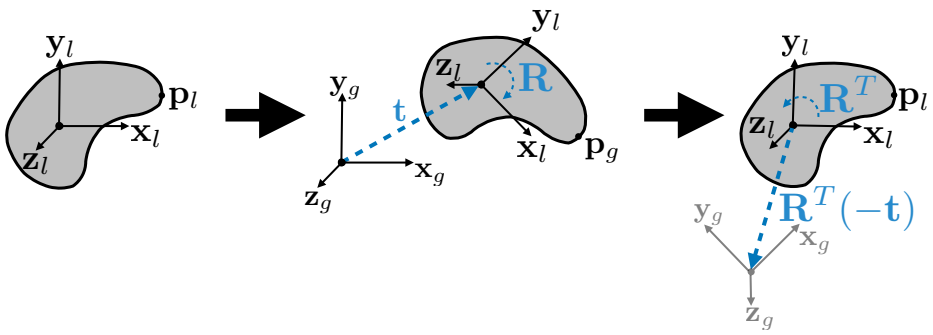


Figure 3.1: *Transformations for global and local coordinate systems.*

the *global coordinate frame*. To describe a rigid body with respect to the global coordinate frame, we can transform any *local coordinate frame* point $\mathbf{p}_l$ using the following operation

$$\mathbf{p}_g = \mathbf{R}\mathbf{p}_l + \mathbf{t}. \tag{3.1}$$

To compute the position of a global point $\mathbf{p}_g$, with respect to the rigid body coordinate frame, we just need to invert the previous operation

$$\mathbf{p}_l = \mathbf{R}^{-1}(\mathbf{p}_g - \mathbf{t}) = \mathbf{R}^T(\mathbf{p}_g - \mathbf{t}). \tag{3.2}$$

Figure 3.1 displays the situation of passing from local coordinates to global coordinates and vice versa.

Being a three-dimensional matrix, $\mathbf{R}$ has nine elements, but rotations can theoretically be expressed using only three parameters. By defining a unitary rotation axis (two parameters) and an angle of rotation (one parameter), we can represent all possible rotations in three dimensions. A more compact representation for rotations are *quaternions*, a number system that extends complex numbers. A quaternion $\mathbf{q}$ is defined as

$$\mathbf{q} = s + xi + yj + zk, \tag{3.3}$$

where s, x, y, z are scalars and i, j, k are the fundamental imaginary quaternion units. Both three-dimensional points and rotations can be represented

using quaternions. When having a point $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)^T$ and a rotation r of θ radians around a unit vector $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)^T$, we can express them as quaternions as follows

$$\mathbf{q}_p = 0 + \mathbf{p}_x i + \mathbf{p}_y j + \mathbf{p}_z k, \tag{3.4}$$

$$\mathbf{q}_r = cos(\frac{\theta}{2}) + sin(\frac{\theta}{2})(\mathbf{u}_x i + \mathbf{u}_y j + \mathbf{u}_z k). \tag{3.5}$$

Before we can define a rotation operation using quaternions, we first need to introduce the notion of conjugate and the inverse of a quaternion $\mathbf{q}$

$$\bar{\mathbf{q}} = s - xi + yj + zk, \tag{3.6}$$

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{||\mathbf{q}||}, \tag{3.7}$$

where $||\mathbf{q}|| = \sqrt{s^2 + x^2 + y^2 + z^2}$ is the norm of the quaternion. A point $\mathbf{p}' = (\mathbf{p}'_x, \mathbf{p}'_y, \mathbf{p}'_z)^T$ is then obtained by rotating $\mathbf{p}$ by r with the following quaternion multiplication

$$\mathbf{q}_{p'} = 0 + \mathbf{p}'_x i + \mathbf{p}'_y j + \mathbf{p}'_z k = \mathbf{q}_r \mathbf{q}_p \mathbf{q}_r^{-1}. \tag{3.8}$$

By substituting the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ with quaternions, equation 3.1 becomes

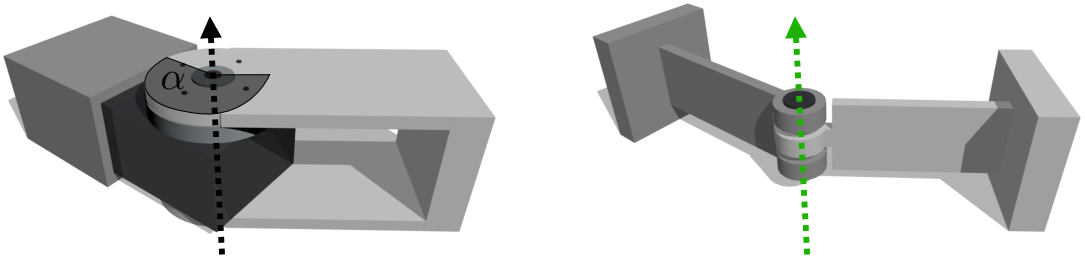
$$\mathbf{q}_{p,g} = \mathbf{q}_r \mathbf{q}_{p,l} \mathbf{q}_r^{-1} + \mathbf{q}_t, \tag{3.9}$$

where $\mathbf{q}_{p,g}$ and $\mathbf{q}_{p,l}$ are, respectively, the quaternion representation of $\mathbf{p}$ in global and local coordinate frames.

The quaternion representation reduces significantly the number of parameters required to store a rotation compared to a 3D matrix. However, as mentioned, a rotation could be represented using three parameters while quaternions need four. To tackle this issue, we express one of the four parameters as a function of the other three. In this thesis, our choice is to assume that the quaternion representing the rigid body orientation is a unit quaternion. Mathematically, we write

$$||\mathbf{q}_r||^2 = cos^2(\frac{\theta}{2}) + sin^2(\frac{\theta}{2})(\mathbf{u}_x^2 + \mathbf{u}_y^2 + \mathbf{u}_z^2) = 1, \tag{3.10}$$

thus defining our scalar function for the quaternion scalar component to be



(a) Motor which fully constrains the states of the connected rigid bodies.

(b) Hinge joint with its degree of freedom.

(c) Universal joint with its two degrees of freedom.

(d) Spherical joint with its three degrees of freedom.

Figure 3.2: *Four types of connectors.*

$$s_r(\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z) = \sqrt{1 - sin^2(\frac{\theta}{2})(\mathbf{u}_x^2 + \mathbf{u}_y^2 + \mathbf{u}_z^2)}. \tag{3.11}$$

The state of a rigid body is therefore representable by storing a three-dimensional vector for both the position and the orientation, for a total of six parameters.

3.1.2 Rigidly Articulated Rigid Bodies

When dealing with rigidly articulated characters, in addition to the rigid bodies states we need to account for the constraints caused by the joints found in the articulations. Whenever two or more rigid bodies are connected through a joint, we need to formulate a constraint vector as a function of the connected rigid bodies states. The constraint vector takes the value of $\mathbf{0}$ whenever the constraint is satisfied. There exist many different types of joints and connectors, but we will discuss only the four mainly relevant for this thesis: motors, hinge joints, universal joints, and spherical joints. Figure 3.2 shows the four types of connectors with the relative degrees of freedom.

Connectors connect pairs of components and constrain their relative motion. To formulate a constraint, we define the locations and the frames which move rigidly together with the two attached components. We label one connected rigid body *input* and the other *output* and then we define their relative local right-handed frames as

$$\mathbf{x}_l^{in}, \quad \left[\mathbf{a}_{x,l}^{in}, \mathbf{a}_{y,l}^{in}, \mathbf{a}_{z,l}^{in}\right] \tag{3.12}$$

$$\mathbf{x}_l^{out}, \quad \left[\mathbf{a}_{x,l}^{out}, \mathbf{a}_{y,l}^{out}, \mathbf{a}_{z,l}^{out}\right] \tag{3.13}$$

where $\mathbf{x}_l^{in}$, $\mathbf{x}_l^{out}$ are the frames origins and $\left[\mathbf{a}_{x,l}^{in}, \mathbf{a}_{y,l}^{in}, \mathbf{a}_{z,l}^{in}\right]$, $\left[\mathbf{a}_{x,l}^{out}, \mathbf{a}_{y,l}^{out}, \mathbf{a}_{z,l}^{out}\right]$ the three orthonormal vectors bases. Given the states s^{in} and s^{out} of the connected rigid bodies, we can transform frames 3.14 and 3.15 to global coordinates using equation 3.9, obtaining

$$\mathbf{x}_g^{in}, \quad \left[\mathbf{a}_{x,g}^{in}, \mathbf{a}_{y,g}^{in}, \mathbf{a}_{z,g}^{in}\right] \tag{3.14}$$

$$\mathbf{x}_g^{out}, \quad \left[\mathbf{a}_{x,g}^{out}, \mathbf{a}_{y,g}^{out}, \mathbf{a}_{z,g}^{out}\right], \tag{3.15}$$

With the help of the transformed global frames, we are able to formulate the constraint vectors for each connector type.

Spherical Joint

A spherical joint has three degrees of freedom (d.o.f.) which allow for any relative orientation between the constrained rigid bodies. However, it restricts the position of the global connector frames to be the same. Therefore we formulate such a constraint as follows

$$\mathbf{c} = \mathbf{x}_g^{in} - \mathbf{x}_g^{out} \tag{3.16}$$

Universal Joint

A universal joint removes an additional rotational degree of freedom with respect to a spherical joint. For the extra constraint, we need two axes to stay orthogonal to one another. To enforce orthogonality, we use the dot product between two vectors, thus leading to the formulation

$$\mathbf{c} = \begin{bmatrix} \mathbf{x}_g^{in} - \mathbf{x}_g^{out} \\ \mathbf{a}_{x,g}^{in} \cdot \mathbf{a}_{z,g}^{out} \end{bmatrix} \tag{3.17}$$

Hinge Joint

To include an additional constraint introduced by the hinge connection, we can use the same technique used for the universal joint. We use the remaining axis from the input frame (since $\mathbf{a}_{z,g}^{in}$ and $\mathbf{a}_{z,g}^{out}$ must stay parallel), and we enforce orthogonality analogously to 3.17

$$\mathbf{c} = \begin{bmatrix} \mathbf{x}_g^{in} - \mathbf{x}_g^{out} \\ \mathbf{a}_{x,g}^{in} \cdot \mathbf{a}_{z,g}^{out} \\ \mathbf{a}_{y,g}^{in} \cdot \mathbf{a}_{z,g}^{out} \end{bmatrix} \tag{3.18}$$

which results in having the rotation axis aligned with $\mathbf{a}_{z,g}^{in} = \mathbf{a}_{z,g}^{out}$.

Motor

Like for the hinge, we assume that the rotation axis points in the direction of the z axis of both global connection frames. To emulate the behavior of a motor moving of α radians, we first rotate the global input frame by α radians around the rotation axis, resulting in a rotated frame $\left[(\mathbf{a}_{x,g}^{in})', (\mathbf{a}_{y,g}^{in})', \mathbf{a}_{z,g}^{in}\right]$. We then match the positions and frames orientations of the input and output frames, resulting in

$$\mathbf{c} = \begin{bmatrix} \mathbf{x}_g^{in} - \mathbf{x}_g^{out} \\ (\mathbf{a}_{x,g}^{in})' \cdot \mathbf{a}_{z,g}^{out} \\ (\mathbf{a}_{y,g}^{in})' \cdot \mathbf{a}_{x,g}^{out} \\ \mathbf{a}_{z,g}^{in} \cdot \mathbf{a}_{y,g}^{out} \end{bmatrix} \tag{3.19}$$

The vectors defined in equations 3.16, 3.17, 3.18, and 3.19 provide an analytical way to quantify how much the constraints are violated. In section 3.2 we describe different methods to enforce the constraint vectors to be close to zero, therefore satisfying the restriction introduced by the connectors.

3.1.3 Compliant Connectors

Sections 3.1.1 and 3.1.2 describe how to model a rigidly articulated character or mechanism. The use of traditional connectors, fully limit the motion of the attached rigid bodies along certain directions and leave them completely free to move along the others. In this thesis, we also emphasize compliance, the property of transferring an input force or displacement from one point to another through elastic deformation. Compliance, differently from traditional joints, exerts some resistance to motion in all different directions.

In our work, we experiment with two type of compliant connectors: torsional springs and flexures. Torsional springs behave like hinge joints but exert a certain torque whenever displaced along the hinge degree of freedom. Flexures approximate the behavior of torsional springs and come with the advantage of, for example, the manufacturing of monolithic structures. However they are more expensive to simulate since, to accurately predict deformation, we need to use a more complex physics model. In this subsection, we describe the representation of the two types of compliant joints and the energy models behind the resistance generated by compliance.

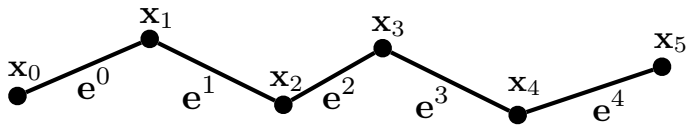
Torsional Spring

Torsional springs operate along a single degree of freedom, the same one allowed by hinge joints. It comes naturally then to represent part of a torsional spring using the formulation described in 3.18. Also, we need to introduce a parameter that relates the amount of generated energy given the change in angle. This parameter is the spring constant k which is intrinsic for each spring. Those are the only information we need to store to represent a torsional spring.

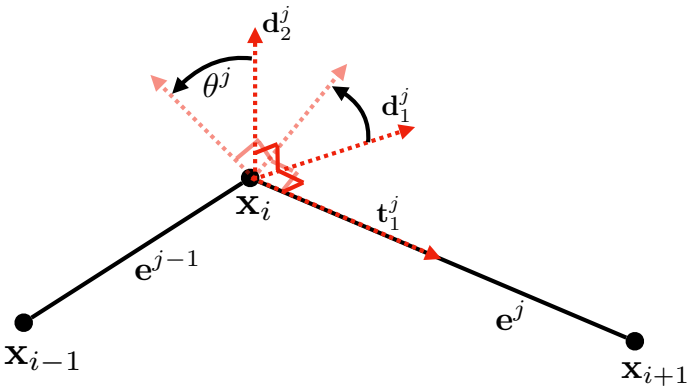
The simplest physics model for torsional springs is to use a quadratic relationship between the change in angle and the energy created. The expression for the spring energy E is the following

$$E = \frac{1}{2}k(\theta)^2 \tag{3.20}$$

While simple, this linear model provides us with the basis for the extended formulation we present in chapter 6.



(a) Centerline of a rod with six vertices and five edges.



(b) Close up of an inner vertex $\mathbf{x}_i$ and its relative frame. In transparency, the same frame rotated by θ^j radians around $\mathbf{t}^i$.

Figure 3.3: *Rod centerline and its degrees of freedom.*

Flexure

Flexures are flexible elements with the function to be compliant along specific directions and stiffer in others. In contrast to torsional springs, where we had the single degree of freedom θ, flexures allow the connected components to have any relative orientation; the shape of the flexure element determines which configuration generates more resistance. In this thesis, we decide to model flexures using the Discrete Elastic Rod model [Bergou et al., 2008]. This paragraph gives a short overview of the model, but for more details, we refer the reader to the original work by Bergou et al.

We describe a flexure element as a thin rod with elliptical cross section of radii a and b. The model stores the centerline of the rod as a sequence of $n+2$

vertices connected by $n+1$ edges. Together with the vertices $0 \leq i \leq n+1$ (lower indices) and the edges $0 \leq j \leq n$ (upper indices) we also store the positions and orientations of the adapter orthonormal frames

$$\mathbf{x}_i \in \mathbb{R}^3 \qquad\qquad \left[\mathbf{d}_1^j, \mathbf{d}_2^j, \mathbf{d}_3^j\right] \in SO(3)$$

The definition of the adapted frames is made, so that $\mathbf{d}_3^j$ lies along the edge $\mathbf{e}^j = \mathbf{x}_{j+1} - \mathbf{x}_j$, which leads to $\mathbf{d}_3^j \equiv \mathbf{t}^j = \mathbf{e}^j/|\mathbf{e}^j|$. The discrete material curvature is defined in an interior vertex i as follows

$$\boldsymbol{\kappa}_i = \frac{1}{2} \sum_{j=i-1}^{i} ((\kappa \mathbf{b})_i \cdot \mathbf{d}_2^j, -(\kappa \mathbf{b})_i \cdot \mathbf{d}_2^j)^T \tag{3.21}$$

where $(\kappa \mathbf{b})_i = \frac{2\mathbf{t}^{i-1} \times \mathbf{t}^i}{1 + \mathbf{t}^{i-1} \cdot \mathbf{t}}$ is the vertex-based curvature binormal. To fully describe flexures we also need to represent the twist m it can occur along the rod centerline. The vertex-related quantity m_i is computed based on how much rotation θ_i we observe between the frames associated with $\mathbf{e}^{i-1}$ and $\mathbf{e}^i$ around their tangents. An exhaustive explanation on how to exactly compute m is beyond the scope of this thesis, and we refer the reader to the work by Bergou et al. [2010].

The vertex-related quantities acting as degrees of freedom in the rod model are $\mathbf{x}_i$ and θ_i. Similarly for the torsional spring setting, whenever the degrees of freedom of the rod are displaced from the rest configuration, the flexure element stores energy. The energy, in this case, is split in three parts: stretch, bend and twist. The total energy E is the sum of the three different components potentials

$$E_s = \frac{1}{2} \sum_{j=0}^{n} k_s^j (\epsilon^j)^2 |\bar{\mathbf{e}}^j| \tag{3.22}$$

$$E_b = \frac{1}{2} \sum_{i=0}^{n} \frac{1}{\bar{l}_i} (\boldsymbol{\kappa}_i - \bar{\boldsymbol{\kappa}}_i)^T B_i (\boldsymbol{\kappa}_i - \bar{\boldsymbol{\kappa}}_i) \tag{3.23}$$

$$E_t = \frac{1}{2} \sum_{i=0}^{n} \beta_i \frac{(m_i - \bar{m}_i)^2}{\bar{l}_i} \tag{3.24}$$

where $\bar{l}_i = \frac{1}{2}(|\bar{\mathbf{e}}^{i-1}| + |\bar{\mathbf{e}}^{i}|)$ is the rest length associated with each vertex assumed to be the average between adjacent edges. Having defined the degrees of freedom and the energy models behind flexures, we are now able to simulate them.

3.2 Physics Simulation

Physics simulation allows predicting the evolution of the states of objects over time. Depending on the goal beyond the prediction, two different types of simulations can be used: *dynamic* or *static*. Dynamic simulations explicitly update time and track objects states and their time derivatives for each time update $t_{i+1} = t_i + \Delta t$. This type of simulation is used when it is important to know the evolution of the system at any point in time. In other scenarios, it might be enough just to know the point of *static equilibrium* of the system. In this case, we are not required to compute the intermediate time steps, and static simulation is used. The contributions proposed in this thesis heavily rely on static simulation, while we only use dynamic simulation to validate our results in chapter 4. For this reason, this section focuses only on how to find such static equilibrium states.

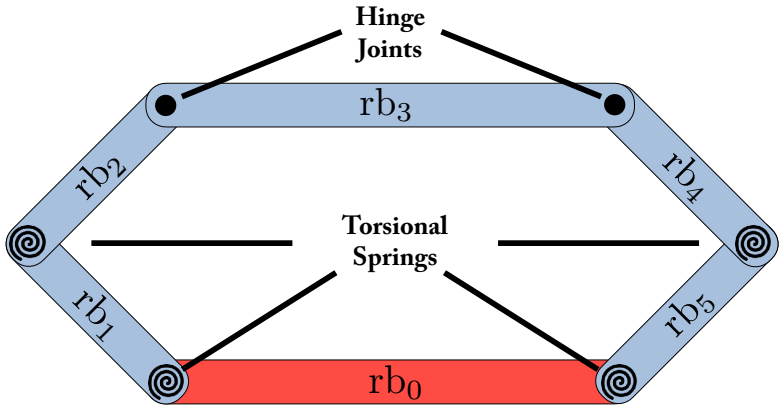
Stable static equilibrium points, lie on local minima of the constrained energy function E_{tot}, where the net sum of forces and torques acting on the system goes to zero. We can, therefore, reformulate the problem as

$$\begin{aligned} \underset{\mathbf{s}}{\text{minimize}} \quad & E_{tot}(\mathbf{s}) = E_{int}(\mathbf{s}) + E_{ext}(\mathbf{s}) \\ \text{subject to} \quad & C(\mathbf{s}) = 0 \end{aligned}$$

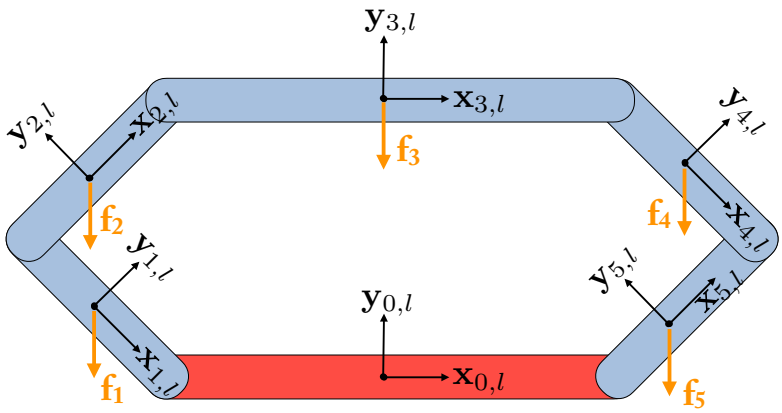
where $\mathbf{s}$ contains the simulation parameters corresponding to the degrees of freedom introduced in the previous sections. The internal energy E_{int} is the combination of the elastic energies generated by the compliant connectors, whereas the external energy E_{ext} describes the potential energy tangled with external forces such gravity or external loads.

Simulation Example

To better understand how to construct such a minimization problem we have a look at a simple example. Figure 3.4 shows an example of a traditional mechanism featuring six rigid bodies connected through four linear springs and two hinge joints. Moreover, gravitational forces are acting on the system, causing to change state. We are interested in finding the set of



(a) Six rigid bodies connected via four torsional springs and two hinge joints. Four rigid bodies are free to move (blue), and one is fixed (red).



(b) The six rigid bodies, showing their respective local coordinate frames. In orange the external gravitational forces acting on the system.

Figure 3.4: *Simple example showing six rigid bodies, connected through two hinge joints and four torsional springs. Gravitational forces are acting on the rigid bodies which are free to move.*

parameters $\mathbf{s}$ for which the total energy $E_{tot}(\mathbf{s})$ is minimal. For this example, the vector of degrees of freedom is the is the stacked sequence of parameters for the five, non-fixed, rigid bodies

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_0 \\ \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_3 \\ \mathbf{s}_4 \end{bmatrix}, \tag{3.25}$$

where $\mathbf{s}_i = (\mathbf{t}_i^T, \mathbf{u}_i^T)^T$, and the vectors $\mathbf{t}_i$ and $\mathbf{u}_i$ are, respectively, the location and orientation d.o.f. of rigid body i, for $0 \leq i \leq 4$.

The internal energy E_{int} is composed by combining the torsional spring energies, and it is defined as

$$E_{int}(\mathbf{s}) = \frac{1}{2} \sum_{j=0}^{3} k_i (\theta_j(\mathbf{s}))^2, \tag{3.26}$$

where $\theta_j(\mathbf{s})$, $0 \leq j \leq 3$, are the springs deformation angles, expressed as functions of the rigid bodies states. The external energy E_{ext} is given by the gravitational potential energy acting on the rigid bodies. We, therefore, write

$$E_{ext}(\mathbf{s}) = \sum_{i=i}^{5} m_i g \mathbf{t}_{i,y}, \tag{3.27}$$

where m_i and $\mathbf{t}_{i,y}$ are, respectively, the mass and the y-component of the position of rigid body i. Additionally, the gravitational field constant is $g = 9.81 N/Kg$. Consequently, the total energy $E_{tot} = E_{int} + E_{ext}$ is obtained by simply adding the internal and external energy components.

In this scenario, the constraints are brought by two hinge joints, and four torsional springs. While coming from different connectors, the constraints coming from traditional joints and springs have the same formulation. Therefore we follow equation 3.18 and we pairwise connect consecutive rigid bodies rb^i and $rb^{(i+1) \bmod 6}$, for $1 \leq i \leq 5$. The final vector of constraints $C(\mathbf{s})$ is composed by $5 \cdot 6 = 30$ scalar nonlinear equations.

Having defined degrees of freedom, energy, and constraints of the system, we are left with finding the set of states $\mathbf{s}$ which solve 3.2. The solution to this problem is typically found with the use of numerical methods, two of which are described in the following sections.

3.2.1 Newton-Raphson Method

The Newton-Raphson Method (also known as simply Newton Method) is a technique to find roots of nonlinear systems of equations numerically. We, therefore, can employ such method to find solution of problem 3.2 by finding a set of parameters $\mathbf{s}$ for which the net sum of forces and torques goes to zero.

Let us define a nonlinear multivariate vector function $\mathbf{f}(\mathbf{x})$ and a set of initial parameters $\mathbf{x}_0$. Our goal is to find a set of parameters $\mathbf{x}^*$ for which $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$. To reach our goal, we need to find a sequence of updates $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{\Delta x}_i$ which converges to $\mathbf{x}^*$. To find $\mathbf{\Delta x}_i$, we first approximate our function $\mathbf{f}(\mathbf{x})$ using the first order Taylor expansion around $\mathbf{x}_i$

$$\mathbf{f}(\mathbf{x}_i + \mathbf{\Delta x}_i) \approx \mathbf{f}(\mathbf{x}_i) + \frac{\partial \mathbf{f}(\mathbf{x}_i)}{\partial \mathbf{x}} \mathbf{\Delta x}_i$$

and by setting the value of the updated function to zero and rearranging, we obtain

$$\begin{aligned} \mathbf{0} &\approx \mathbf{f}(\mathbf{x}_i) + \frac{\partial \mathbf{f}(\mathbf{x}_i)}{\partial \mathbf{x}} \mathbf{\Delta x}_i \\ -\frac{\partial \mathbf{f}(\mathbf{x}_i)}{\partial \mathbf{x}} \mathbf{\Delta x}_i &\approx \mathbf{f}(\mathbf{x}_i) \\ \frac{\partial \mathbf{f}(\mathbf{x}_i)}{\partial \mathbf{x}} \mathbf{\Delta x}_i &\approx -\mathbf{f}(\mathbf{x}_i) \end{aligned} \tag{3.28}$$

Being $\mathbf{\Delta x}_i$ our unknown, each update requires solving the linear system 3.28 to find the step in parameter space which get us closer to the root.

To apply the same method to minimize our multivariate scalar energy function E_{tot}, we first need to differentiate with respect to the parameters $\mathbf{s}$ to obtain a vector function in the same form of $\mathbf{f}(\mathbf{x})$ and proceed analogously to get the adapted linear system

$$\frac{\partial^2 E_{tot}(\mathbf{s})}{\partial \mathbf{s}^2} \mathbf{\Delta s} = -\frac{\partial E_{tot}(\mathbf{s})}{\partial \mathbf{s}} \tag{3.29}$$

where $\partial E_{tot}(\mathbf{s})/\partial \mathbf{s}$ the gradient and $\partial^2 E_{tot}(\mathbf{s})/\partial \mathbf{s}^2$ is the Hessian (often denoted simply with H) of the energy function $E_{tot}(\mathbf{s})$. We then iterate and keep building and solving the same system of equations for each new vector candidate $\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{\Delta s}_i$ until convergence or, i.e., until the norm of the gradient is close enough to zero.

Line Search

The Newton method treats the energy function $E_{tot}(\mathbf{s})$ as if it was quadratic but, most of the times, this is not the case. The result is that a full step of $\mathbf{\Delta s}_i$ in parameter space might bring our set of parameters in a point which where $E_{tot}(\mathbf{s}_{i+1}) > E_{tot}(\mathbf{s}_i)$, even if $\mathbf{\Delta s}_i$ is a descending direction. This happens because the energy gradient is nonlinear and our predicted minimum lies in a place where the energy has a higher value. To solve this problem, we modify the update equation to be $\mathbf{s}_{i+1} = \mathbf{s}_i + \alpha\mathbf{\Delta s}_i$, with $0 < \alpha \leq 1$. To decide on which value we need to use for alpha, for each iteration i, we keep decreasing the value of alpha according to $\alpha^{j+1} = \alpha^j * 0.5$, where $\alpha^0 = 1$. We stop searching for α whenever $E_{tot}(\mathbf{s}_i + \alpha^j\mathbf{\Delta s}_i) < E_{tot}(\mathbf{s}_i)$. More advanced line search methods exist to find a value of α which decreases the energy value more but at the cost of being more expensive. In general, it is preferable to spend more computational resources in computing a new search direction $\mathbf{\Delta s}$ rather than in the line search parameter α.

The Newton method is a powerful tool for energy minimization problems but, taken as is, does not allow any constraints handling. Different techniques were proposed to minimize constrained multivariate functions, and in the next sections we present two: the *quadratic penalty method* and the *sequential quadratic programming* (SQP).

3.2.2 Quadratic Penalty Method

The simplicity of the quadratic penalty method made it popular among many domains focused on problem-solving. This method treat the constraints vectors as they part of the energy in the system, adding a weighted square norm term that modifies the energy function to

$$E_c(\mathbf{s}) = E_{tot}(\mathbf{s}) + \frac{\mu}{2}||C(\mathbf{s})||_2^2, \qquad \mu > 0. \tag{3.30}$$

We note that function E_c can now be minimized using the Newton method described above. While simple to implement and integrate into the original energy function E_{tot}, this method suffers from some drawbacks. First, the amount of constraint-violation depends on the penalty parameter μ, which must tend to infinity to bring the constraints vector value to zero. It is therefore tempting to use a very large value of μ to approach the constraints satisfaction, which in turn increases the ill-conditioning of the Hessian matrix of E_c (see section 3.2.1) leading to numerical problems. Moreover, the fulfillment of the constraints can conflict with a decrease in physical energy E_{tot},

and the solution of problem 3.2 might not coincide with the one of problem 3.30. Due to the issues described above, other numerical methods were designed to solve energy minimization problems, handling constraints differently; among those we find SQP, explained in the next section.

3.2.3 Sequential Quadratic Programming

Whenever the constraints functions are highly nonlinear, the sequential quadratic programming methods is especially effective to solve constrained minimization problems. As the name suggests, the idea behind SQP consists in iteratively solve a sequence approximations of the original problem using the quadratic programming technique. First, we need to introduce the Lagrangian of our energy function E_{tot}

$$\mathcal{L}(\mathbf{s}, \boldsymbol{\lambda}) = E_{tot}(\mathbf{s}) + \boldsymbol{\lambda}^T C(\mathbf{s}) \tag{3.31}$$

where $\boldsymbol{\lambda}$ is a vector of auxiliary variables called the *Lagrange multipliers*. Then, according to optimization theory [Nocedal and Wright, 2006], problem 3.2 translates to finding a set of parameters $\mathbf{s}^*$ and the corresponding Lagrange multipliers $\boldsymbol{\lambda}^*$ which satisfy the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{s}^*, \boldsymbol{\lambda}^*)}{\partial \mathbf{s}} &= 0, \\ C(\mathbf{s}^*) &= 0 \end{aligned} \tag{3.32}$$

To find a solution for problem 3.32, we can employ the Newton method (see section 3.2.1) to solve the following KKT system for each iteration i

$$\begin{bmatrix} H_i(\mathbf{s}_i, \boldsymbol{\lambda}_i) & -\mathbf{J}^T(\mathbf{s}_i) \\ \mathbf{J}(\mathbf{s}_i) & 0 \end{bmatrix} \begin{pmatrix} \Delta\mathbf{s}_i \\ \Delta\boldsymbol{\lambda}_i \end{pmatrix} = - \begin{pmatrix} \mathbf{g}(\mathbf{s}_i) - \mathbf{J}^T(\mathbf{s}_i)\boldsymbol{\lambda}_i \\ C(\mathbf{s}_i) \end{pmatrix} \tag{3.33}$$

where $H_i(\mathbf{s}, \boldsymbol{\lambda}) = \partial^2 \mathcal{L}(\mathbf{s}, \boldsymbol{\lambda}) / \partial \mathbf{s}^2$ is the Hessian of the Lagrangian, $\mathbf{J}(\mathbf{s}) = \partial C(\mathbf{s}) / \partial \mathbf{s}$ is the constraints Jacobian and $\mathbf{g}(\mathbf{s}) = \partial E_{tot}(\mathbf{s}) / \partial \mathbf{s}$ is the energy gradient. Similarly to section 3.2.1, due to nonlinearities, the solution of system 3.33 gives a search direction rather than the conclusive change in parameter space. The final computation of the iterates has the form

$$\begin{pmatrix} \mathbf{s}_{i+1} \\ \boldsymbol{\lambda}_{i+1} \end{pmatrix} = \begin{pmatrix} \mathbf{s}_i \\ \boldsymbol{\lambda}_i \end{pmatrix} + \alpha \begin{pmatrix} \boldsymbol{\Delta}\mathbf{s}_i \\ \boldsymbol{\Delta}\boldsymbol{\lambda}_i \end{pmatrix}, \qquad 0 < \alpha \leq 1. \tag{3.34}$$

Differently from the unconstrained case, our choice of α needs to take into account constraints violation. The construction of a proper *merit function* able to handle constraints is described in the next section.

The Merit Function

Solving system 3.33 ensures that no constraint violation happens along the found search direction, as long as the constraints are linear. When that is not the case, a reduction of the energy function E_{tot} can cause an increase in constraint violation and vice versa. Therefore, we need to construct and minimize a merit function which combines both the energy and the constraints. The ℓ_1 merit function we use in this thesis has the form

$$\phi_1(\mathbf{s}) = E_{tot}(\mathbf{s}) + \mu||C(\mathbf{s})||_1, \qquad \mu > 0. \tag{3.35}$$

Having our merit function ϕ_1 defined, a line search step $\alpha^j(\boldsymbol{\Delta}\mathbf{s}, \boldsymbol{\Delta}\boldsymbol{\lambda})$ is accepted if the following decrease condition holds

$$\phi_1(\mathbf{s}_i + \alpha^j \boldsymbol{\Delta}\mathbf{s}_i) \leq \phi_1(\mathbf{s}_i) + \eta\alpha^j(\mathbf{g}(\mathbf{s}_i)^T\boldsymbol{\Delta}\mathbf{s} - \mu||C(\mathbf{s}_i)||_1), \qquad \eta \in (0,1). \tag{3.36}$$

The choice of μ plays a significant role in the convergence of the method since inequality 3.36 holds only if μ is chosen sufficiently large. At each SQP iteration, we check if μ satisfies

$$\mu \geq \frac{\mathbf{g}(\mathbf{s}_i)^T\boldsymbol{\Delta}\mathbf{s}_i + (\sigma/2)\boldsymbol{\Delta}\mathbf{s}_i^T H(\mathbf{s}_i, \boldsymbol{\lambda}_i)\boldsymbol{\Delta}\mathbf{s}}{(1-\rho)||C(\mathbf{s}_i)||_1}, \qquad \rho \in (0,1), \tag{3.37}$$

otherwise, it is increased so that it satisfies inequality 3.37. The constant σ is a boolean parameter used to handle the case of a non-positive definite Hessian matrix $H(\mathbf{s}_i, \boldsymbol{\lambda}_i)$. We define σ to be

$$\sigma = \begin{cases} 1 & \text{if } \boldsymbol{\Delta}\mathbf{s}_i^T H(\mathbf{s}_i, \boldsymbol{\lambda}_i)\boldsymbol{\Delta}\mathbf{s} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.38}$$

By enforcing inequality 3.37 and choosing σ as defined, we can ensure that the search direction $(\Delta \mathbf{s}_i, \Delta \boldsymbol{\lambda}_i)$ is a descending path for ϕ_1. The details of the derivation of such property are described in [Nocedal, 1980] and are omitted from this thesis.

3.3 The Inverse Design Problem

Physics simulation is very useful in the manufacturing processes since, given a particular design, it can predict whether or not a desired property, say a desired shape under load, will be observed in the final fabricated object. Thanks to the fact that simulation requires a lot less time than the actual fabrication process, iterations to find the version that best satisfies the user requirements are significantly faster. However, even with the help of simulation, a multitude of properties are difficult to obtain just by using a trial and error approach. First, since the number of parameters that a user can change is very high and, second, a small variation in the design can lead to a very different final structure behavior. The lack of intuitiveness brings the need for tools that, given a set of goals, can to compute the design parameters producing such goals. The process of finding these parameters is called inverse designing. In general, the goals the user can infer are described mathematically through a multivariate scalar function $W(\mathbf{s}, \mathbf{p})$, with $\mathbf{p}$ being the design parameters, which can be minimized using standard numerical tools. However, whenever the system states $\mathbf{s}$ are the result of a static simulation dependent on $\mathbf{p}$, a specific way of handling objective derivatives is required.

Inverse Design Example

For concreteness, let us have a look again at the example shown in picture 3.4. A typical inverse design problem could be to find the four spring constants for which, after simulation, the rigid body rb_3 is at a height of $\tilde{y}$ while staying horizontal. The target configuration is shown in picture 3.5. In this example, the degrees of freedom $\mathbf{s}$ and energy E_{tot} are the ones defined in section 3.2. Additionally, we define our design parameters to be

$$\mathbf{p} = \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}. \tag{3.39}$$

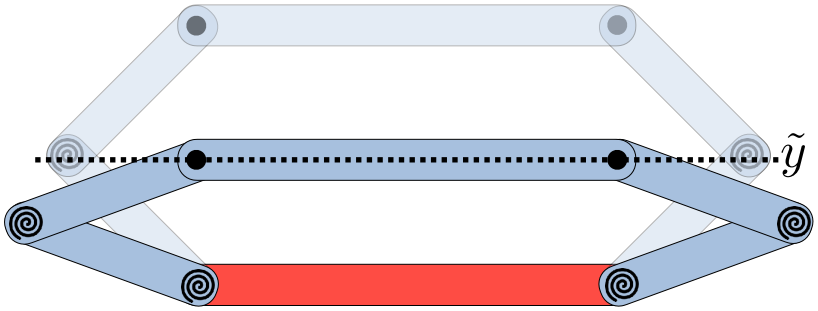


Figure 3.5: *Target configuration, with rb_3 being horizontal and at a hight of $\tilde{y}$. In transparency, the rest configuration before applying gravity.*

Given the specification of the desired configuration under gravity, the objective function $W(\mathbf{s}, \mathbf{p})$ has to include the formulation of two goals: match a target height and being horizontal. The first objective penalizes any deviation of $\mathbf{t}_{3,y}$ (the y component of the translation vector $\mathbf{t}$ related to tb_3) from $\tilde{y}$, and can be defined as

$$W_{height}(\mathbf{s}, \mathbf{p}) = \frac{1}{2}(\mathbf{t}_{3,y} - \tilde{y})^2. \tag{3.40}$$

The second objective encourages the orientation of rb_3 to stay horizontal, and can be formalized as

$$W_{horizontal}(\mathbf{s}, \mathbf{p}) = \frac{1}{2}||\mathbf{u}_3||_2^2. \tag{3.41}$$

Equation 3.41 emerges from the fact that the horizontal orientation of rb_3 can only be represented through an identity quaternion $\mathbf{q}_3 = s_3 + (i, j, k)\mathbf{u}_3 = 1 + 0i + 0j + 0k$. Therefore, by restricting $\mathbf{u}_3$ to be $\mathbf{0}$, we can achieve an horizontal orientation for rb_3. The final objective function is the sum of functions 3.40 and 3.41, and has the form

$$W(\mathbf{s}, \mathbf{p}) = W_{height}(\mathbf{s}, \mathbf{p}) + W_{horizontal}(\mathbf{s}, \mathbf{p}). \tag{3.42}$$

Once the design parameters $\mathbf{p}$ and the objective function $W(\mathbf{s}, \mathbf{p})$ are defined, we can apply numerical methods to find the design parameters which minimize W. In the next section, we describe a method to perform inverse

design optimization when a static simulation is involved in the evaluation of the objective function.

3.3.1 Sensitivity Analysis

Let us define an objective function $W(\mathbf{s}, \mathbf{p})$ dependant on both the set of systems degrees of freedom $\mathbf{s}$ and the design parameters $\mathbf{p}$. In addition, let the degrees of freedom be the result of an energy minimization stage for $E_{tot}(\mathbf{s}, \mathbf{p})$. We describe our inverse design problem as

$$\begin{aligned} \underset{\mathbf{p}}{\text{minimize}} \quad & W(\mathbf{s}, \mathbf{p}) \\ \text{subject to} \quad & \frac{\partial E_{tot}(\mathbf{s}, \mathbf{p})}{\partial \mathbf{s}} = \mathbf{f}(\mathbf{s}, \mathbf{p}) \quad = 0. \end{aligned} \tag{3.43}$$

To minimize the objective function W, we follow the same procedure described in section 3.2.1. Thus we are looking for a change $\mathbf{\Delta p}$ in the set of design parameter for which

$$\frac{d^2 W(\mathbf{s}, \mathbf{p})}{d\mathbf{p}^2} \mathbf{\Delta p} = -\frac{dW(\mathbf{s}, \mathbf{p})}{d\mathbf{p}}, \tag{3.44}$$

where, as opposed as for the Newton method, we are interested in the total derivatives rather than in the partial ones. First, we focus on the computation of the first order total derivative of W with respect to the design parameters $\mathbf{p}$ and postpone the discussion of the second order derivative in section 3.3.2.

With the help of the *implicit function theorem* [Krantz and Parks, 2002], we can formulate the gradient of the objective function W as

$$\frac{dW(\mathbf{s}(\mathbf{p}), \mathbf{p})}{d\mathbf{p}} = \frac{\partial W(\mathbf{s}(\mathbf{p}), \mathbf{p})}{\partial \mathbf{p}} + \frac{\partial \mathbf{s}(\mathbf{p})}{\partial \mathbf{p}}^T \frac{\partial W(\mathbf{s}(\mathbf{p}), \mathbf{p})}{\partial \mathbf{s}}, \tag{3.45}$$

where the degrees of freedom $\mathbf{s}(\mathbf{p})$ are now described as a function of the design parameters $\mathbf{p}$. After the expansion of the gradient, our task is to enforce the constraints we find in 3.43 and define the state function $\mathbf{s}(\mathbf{p})$. This goal is achieved simultaneously with the use of another Taylor expansion, this time of the constraints function $\mathbf{f}(\mathbf{s}, \mathbf{p})$.

We assume that $\mathbf{f}(\mathbf{s}, \mathbf{p}) = 0$ (assumption 3.3.1) holds. We are looking for a change $(\mathbf{\Delta s}, \mathbf{\Delta p})$ in both degrees of freedom and parameters space for which

$\mathbf{f}(\mathbf{s}+\mathbf{\Delta s}, \mathbf{p}+\mathbf{\Delta p}) = 0$ is still valid. Employing the first order Taylor approximation, we write

$$
\begin{aligned}
\mathbf{f}(\mathbf{s}+\mathbf{\Delta s}, \mathbf{p}+\mathbf{\Delta p}) &= 0 \\
\mathbf{f}(\mathbf{s},\mathbf{p}) + \frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{s}}\mathbf{\Delta s} + \frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{p}}\mathbf{\Delta p} &\approx 0 \\
\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{s}}\mathbf{\Delta s} + \frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{p}}\mathbf{\Delta p} &\approx 0, \qquad \text{assumption 3.3.1} \\
\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{s}}\mathbf{\Delta s} &\approx -\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{p}}\mathbf{\Delta p}, \\
\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{s}}\frac{\mathbf{\Delta s}}{\mathbf{\Delta p}} &\approx -\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{p}}, \\
\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{s}}\frac{\partial \mathbf{s}}{\partial \mathbf{p}} &\approx -\frac{\partial \mathbf{f}(\mathbf{s},\mathbf{p})}{\partial \mathbf{p}}, \qquad (3.46)
\end{aligned}
$$

for an infinitesimally small change $(\mathbf{\Delta s}, \mathbf{\Delta p})$. We note that $\partial \mathbf{f}(\mathbf{s},\mathbf{p})/\partial \mathbf{s}$ is the Hessian matrix $H(\mathbf{s})$ of the energy E_{tot}, and $\partial \mathbf{f}(\mathbf{s},\mathbf{p})/\partial \mathbf{p}$ is the Jacobian matrix of the energy gradient $\mathbf{g}(\mathbf{s})$ with respect to the design variables $\mathbf{p}$. By solving the set of equation systems 3.46, we compute the missing piece to evaluate the first order total derivative of W 3.45. To complete the computation of the full iteration 3.44, we also need the total second order derivative $d^2W(\mathbf{s},\mathbf{p})/d\mathbf{p}^2$. This matrix, however, happens to be difficult and expensive to compute, because $\partial^2\mathbf{s}/\partial\mathbf{p}^2$ extends to the third dimension. For this reason, using an approximation of the second order derivative, rather than its analytical version, can lead to a minimum faster. In this thesis, more in particular in chapter 5 and 6, we approximate the Hessians of the corresponding objective functions using the *BFGS* method, described in section 3.3.2.

Line Search

It is important to note that, by following 3.46 to compute $\partial\mathbf{s}/\partial\mathbf{p}$, the search direction $\mathbf{\Delta p}$ automatically ensures the satisfaction of the (linearized) constraints. However, as for the case of SQP, when performing the line search stage, we need to make sure that even nonlinear constraints fully satisfied. To this extent, during line search, the evaluation of the objective function W (required to evaluate the termination condition for α) is done involving a static simulation step. More in particular, given a line search update with the form $\mathbf{p}_{i+1}^j = \mathbf{p}_i + \alpha^j \mathbf{\Delta p}_i$, at each iteration j we first update $\mathbf{p}_{i+1}^j$ and, with the new design parameters, we compute the states $\mathbf{s}_{i+1}^j$ which satisfy

$\mathbf{f}(\mathbf{s}_{i+1}^j, \mathbf{p}_{i+1}^j) = 0$, using the methods described in section 3.2. Finally, having the iterate-candidate $(\mathbf{s}_{i+1}^j, \mathbf{p}_{i+1}^j)$, we can compare the values of the objective function W and accept the new iterate if $W(\mathbf{s}_{i+1}^j, \mathbf{p}_{i+1}^j) < W(\mathbf{s}_i, \mathbf{p}_i)$.

3.3.2 The Broyden–Fletcher–Goldfarb–Shanno Method

The BFGS method is a quasi-Newton algorithm for root finding, named after its discoverers Broyden, Fletcher, Goldfarb, and Shanno. The primary difference between BFGS and the Newtown method is that the former uses an approximate Hessian $\mathbf{B}$, rather than its analytical version H, to solve the iteration step described in equation 3.28.

Let us define an objective function $f(\mathbf{p})$ and its quadratic model around $\mathbf{p}_i$ to be

$$m_i(\mathbf{\Delta p}_i) = f(\mathbf{p}_i) + \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}} \mathbf{\Delta p}_i + \frac{1}{2} \mathbf{\Delta p}_i^T \mathbf{B}_i \mathbf{\Delta p}_i. \tag{3.47}$$

The minimizer $\mathbf{\Delta p}_i$ of the convex model 3.47 can then be written as

$$\mathbf{B}_i \mathbf{\Delta p}_i = -\frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}}, \tag{3.48}$$

and, after the line search stage, the new iterate is $\mathbf{p}_{i+1} = \mathbf{p}_i + \alpha_i \mathbf{\Delta p}_i$.

The approximate Hessian matrix $\mathbf{B}_i$ is updated at every iteration i, taking into account the curvature of the function measured at current and previous iterations. More in detail, let us assume that we want to estimate the new quadratic model of $f(\mathbf{p})$ around $\mathbf{p}_{i+1}$, which has the form

$$m_{i+1}(\mathbf{\Delta p}_{i+1}) = f(\mathbf{p}_{i+1}) + \frac{\partial f(\mathbf{p}_{i+1})}{\partial \mathbf{p}} \mathbf{\Delta p}_{i+1} + \frac{1}{2} \mathbf{\Delta p}_{i+1}^T \mathbf{B}_{i+1} \mathbf{\Delta p}_{i+1}. \tag{3.49}$$

To be consistent, matrix $\mathbf{B}_{i+1}$ should map the gradient of model m_{i+1} at $\mathbf{p}_i$ to the same as the gradient of function f at same location $\mathbf{p}_i$. This relation can be written as

$$\frac{\partial m_{i+1}(-\alpha \mathbf{\Delta p}_i)}{\partial \mathbf{p}} = \frac{\partial f(\mathbf{p}_{i+1})}{\partial \mathbf{p}} - \alpha \mathbf{B}_{i+1} \mathbf{\Delta p}_i = \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}}, \tag{3.50}$$

and after rearranging we obtain

$$\mathbf{B}_{i+1}s_i = y_i, \tag{3.51}$$

where

$$s_i \equiv \alpha_i \Delta \mathbf{p}_i \qquad \text{and} \qquad y_i \equiv \frac{\partial f(\mathbf{p}_{i+1})}{\partial \mathbf{p}} - \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}}, \tag{3.52}$$

are two new defined vectors to simplify further notations. To describe a convex quadratic model, $\mathbf{B}$ must be positive definite. This requirement introduces n inequalities in addition to the n equality constraints imposed by 3.51. Even combining these constraints is not enough to absorb the $n(n+1)/2$ degrees of freedom found in the symmetric matrix $\mathbf{B}$. To uniquely define the update for our approximate Hessian matrix $\mathbf{B}$, we require that $\mathbf{B}_{i+1}$ is *as close as possible* to $\mathbf{B}_i$ in *some metric*. Mathematically, we describe the problem as

$$\begin{aligned} \underset{\mathbf{B}_{i+1}}{\text{minimize}} \quad & ||\mathbf{B}_{i+1} - \mathbf{B}_i|| \\ \text{subject to} \quad & \mathbf{B}_{i+1} = \mathbf{B}_{i+1}^T, \\ & \mathbf{B}_{i+1}s_i = y_i, \end{aligned} \tag{3.53}$$

where $s_i^T y_i > 0$, and $\mathbf{B}_i, \mathbf{B}_{i+1}$ are symmetric positive definite matrices. The optimization problem 3.54 uniquely defines the update step for the Hessian approximation at every quasi-Newton iteration i.

The same approach described above can also be applied to compute an approximation of the inverse of the Hessian, rather than the Hessian itself. The advantage of directly operating on the inverse matrix is the ability to compute the quasi-Newton iteration step 3.48 by performing a matrix-vector multiplication rather than solving a linear system of equations. Following the same argumentation line we traced for $\mathbf{B}_{i+1}$, what we are interested in is a matrix $\mathbf{C}_{i+1}$ for which

$$\begin{aligned} \underset{\mathbf{C}_{i+1}}{\text{minimize}} \quad & ||\mathbf{C}_{i+1} - \mathbf{C}_i|| \\ \text{subject to} \quad & \mathbf{C}_{i+1} = \mathbf{C}_{i+1}^T, \\ & \mathbf{C}_{i+1}y_i = s_i. \end{aligned} \tag{3.54}$$

When the *weighted Frobenius norm* is used as closeness condition, $\mathbf{C}_{i+1}$ has a unique solution which is given by

$$\mathbf{C}_{i+1} = (\mathbf{I} - \frac{s_i y_i^T}{y_i^T s_i})\mathbf{C}_i(\mathbf{I} - \frac{y_i s_i^T}{y_i^T s_i}) + \frac{s_i s_i^T}{y_i^T s_i}, \tag{3.55}$$

where $\mathbf{I}$ is the identity matrix. The weighted Frobenius norm is chosen so that the problem becomes scale-invariant, i.e., the solution does not depend on the variables units. Since the implementation of the BFGS update formula does not require the knowledge of the of the norm formulation, we omit the details in this thesis and refer the interested readers to [Nocedal and Wright, 2006].

The last matter we need to address to be able to fully implement the BFGS method is the initialization of $\mathbf{C}_0$. Generally, the initialization of the approximate inverse Hessian matrix is done by assigning $\mathbf{C}_0 = \beta\mathbf{I}$, where the choice of the scalar β should match the scaling of the variables. Other starting values can be used for $\mathbf{C}_0$, but they are usually problem-specific and not equally suited for different cases. Another example of initialization is to use the gradient information to compute the initialization of the Hessian using finite difference and then invert it to obtain $\mathbf{C}_0$.

3.4 Conclusion

In this chapter, we discussed a way of representing rigid bodies and how to combine them into a single system to form an articulated character or mechanism. We introduced different kinds of connectors and the relative formulations for their constraints and energies. After, we showed multiple techniques to be employed to perform constrained and unconstrained energy minimization, allowing us to physically simulate our compliant-articulated assemblies and compute the states that bring the system in static equilibrium. Finally, we described an additional method that can be used for function minimization when a static simulation is required to evaluate an objective function. Such a technique is particularly suited whenever the goal is to find a solution for an inverse design problem, even more so, if the number of statically stable states that influence the objective function is high. The modeling, simulation and inverse problem solving of compliant assemblies are the foundations below the contributions proposed by this thesis, and this chapter is the attempt to provide a suitable introduction to the three topics.

In the following chapters, we will adapt the general optimization methods introduced above to fit better the specific problems that we tackle in this thesis. We will start by considering rigidly articulated and full-actuated robots, where our research goal will be to design an interactive interface,

able to guide casual users and experts through the process of designing 3D-printable walking creatures (Chapter 4). Our problem formulation allows us to avoid any static simulation during the objective function evaluation, which makes SQP (3.2.3) our choice to minimize our objective function. We then will transfer the focus on compliance, more in particular on how to convert traditional mechanical assemblies to compliant mechanisms which move and transfer the motion by storing and releasing elastic energy (Chapter 5). Lastly, this thesis proposes a method to solve the inverse design problem of creating characters connected with torsional springs, where the actuation is propagated exclusively by cables (Chapter 6). The latter two chapters presents formulations which relies on static simulation for objectives evaluation, therefore our choice of employing sensitivity analysis as our optimization tool.

CHAPTER 4

Interactive Design of 3D-Printable Robotic Creatures

In this chapter, we present an interactive design system that allows casual users to quickly create 3D-printable robotic creatures. Our approach automates the tedious parts of the design process while providing ample room for customization of morphology, proportions, gait and motion style. The technical core of our framework is an efficient optimization-based solution that generates stable motions for legged robots of arbitrary designs. An intuitive set of editing tools allows the user to interactively explore the space of feasible designs and to study the relationship between morphological features and the resulting motions. Fabrication blueprints are generated automatically such that the robot designs can be manufactured using 3D-printing and off-the-shelf servo motors. We demonstrate the effectiveness of our solution by designing six robotic creatures with a variety of morphological features: two, four or five legs, point or area feet, actuated spines and different proportions. We validate the feasibility of the designs generated with our system through physics simulations and physically-fabricated prototypes.

4.1 Introduction

The desire to create mechanical creatures that are capable of lifelike motions and behaviors dates back to ancient times. However, it was only during the last century that this vision started to become reality. Today, mobile robots are ubiquitous in industry and they start to enter our daily life in the form of electro-mechanical toys, robotic pets, and household assistants.

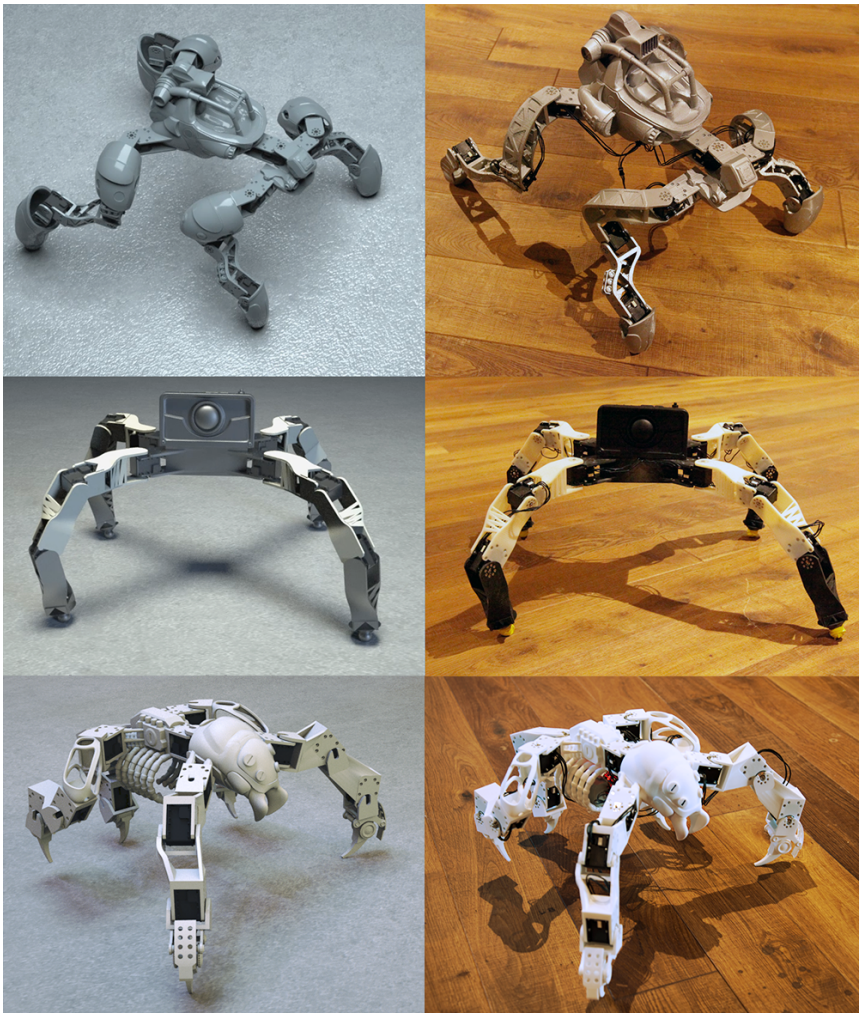


Figure 4.1: *Digital designs (left) and physical prototypes (right) for our* Ranger *(top),* Bobby *(middle) and* Predator *(bottom) designs, fabricated using 3D-printing and off-the-shelf servo motors.*

The recent progress in 3D-printing technology and the advent of powerful, simple-to-program hardware platforms like Arduino now open the door to a new generation of *personal robots*—unique companions that we custom-design according to our needs and preferences. Already now, the rapidly growing community of makers and technology enthusiasts indicates that there is significant interest in this topic. Nevertheless, creating compelling robotic creatures is currently a formidable task that only experienced engineers can successfully undertake.

Driven by the progress in rapid manufacturing technology, the graphics community has recently started to embrace the challenge of translating digital characters into physical artifacts [Zhu et al., 2012; Coros et al., 2013;

Ceylan et al., 2013; Thomaszewski et al., 2014]. While current methods can create physical characters whose motion closely resemble their digital counterparts, this motion is merely for display; stable locomotion, however, poses complex requirements on the physics and geometry of motion—criteria that digital animations fail to fulfill in general.

We present an interactive design system that allows casual users to quickly design 3D-printable robotic creatures. We are inspired by the simplicity, power and flexibility of the character design system by Hecker et al. [2008], empowering novice users to quickly create digital characters with custom shape and motion. Our ambition is to make the design of compelling robotic creatures as accessible and intuitive. A number of challenges have to be overcome in order to achieve this goal. First, as opposed to virtual characters, the motions designed for robotic creatures have to be physically correct; otherwise, the robot will not move as expected, fail to move, or simply fall over. This *physical feasibility* requires a high degree of coordination between the motions of different body parts, which is difficult to achieve with traditional animation approaches. Second, digital characters often exhibit a large number of degrees of freedom in order to allow for highly expressive animations. When creating robotic creatures, however, a much more careful balance between complexity of design, and therefore cost, and range of achievable motions is required.

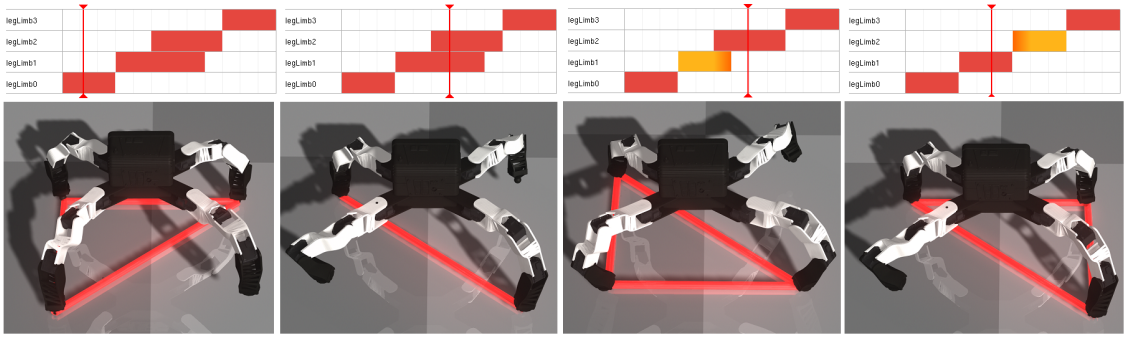


Figure 4.2: *The footfall pattern indicates which leg is in stance (white) or in swing (red) mode. In this example, the user interactively adjusts the footfall pattern such that three legs are always in stance mode.*

4.2 Overview

We propose an end-to-end solution for creating robotic creatures, implemented as an interactive tool that allows the user to design the structure

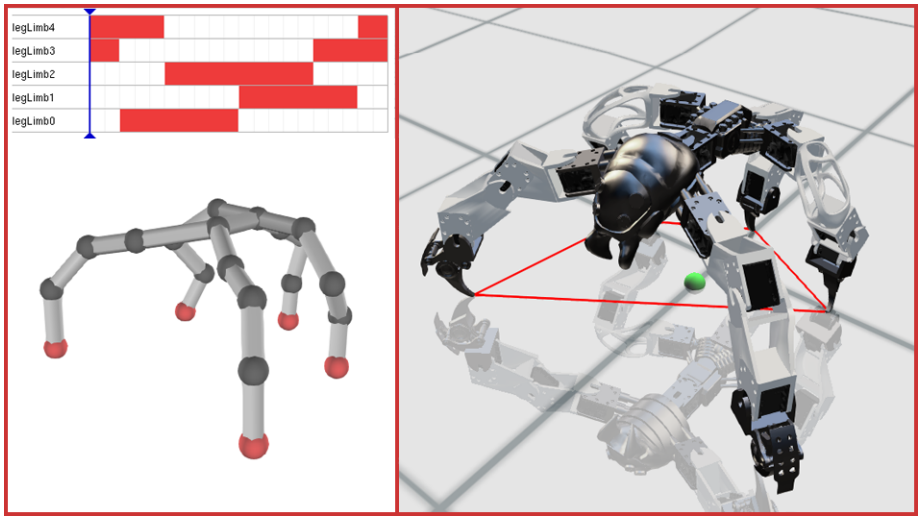


Figure 4.3: *Snapshot of the design interface.* Left*: the design viewport with the footfall pattern graph.* Right*: the preview window showing the center of pressure of the robot (green) and the support polygon (red).*

and motion of a robot while receiving immediate feedback on its expected real-world behavior.

Design Interface Our design interface is structured into two viewports (see Fig. 4.3): one for editing the structure and motion of the robot, one for displaying the resulting real-world behavior as predicted by our optimization or through physics simulation. The heart of the interface is formed by a set of easy-to-use editing tools.

Structure Editing The user starts by loading a description file that specifies an initial skeletal structure of the robot, as defined through a typical hierarchy of bones connected by joints. Initial geometry is created from this information and a virtual, uni-axial motor is placed at each joint position. The user can freely edit the robot's structure at all times by adding or removing motors, thus altering the morphology of the design, or by adjusting the position or orientation of the motors.

Motion Editing The motion of a robotic creature is completely described by the trajectories of its joints. However, authoring motions directly in this high-dimensional space is unintuitive, tedious, and very unlikely to lead to stable movements. We therefore propose a set of higher-level motion au-

thoring and editing tools, designed to be mutually orthogonal and intuitive for the user.

Motions are largely characterized by their footfall pattern, indicating for each instant during a gait cycle which of the legs are in contact with the ground (stance mode) and which ones are in flight (swing mode). Our interface displays these information as a time-dependent graph, allowing for quick inspection and direct editing by the user (see Fig. 4.2). In particular, the user can change the duration of the stance and swing phases for any leg and change the relative ordering of the footfalls. While not all patterns lead to desirable motions, this is immediately clear upon inspecting the evolution of the support polygon through time or by observing the motions generated by our framework. The immediate feedback provided by our framework allows the user to interactively adjust the footfall pattern in order to find satisfying solutions.

Higher-level goals such as the walking direction, speed or turning rate can be provided by the user so as to specify the behavior of the robotic creatures that are being designed. Further, the user can control the overall motion style by editing the movement of the robot's center of mass and of the feet trajectories. The motions generated by our optimization-based framework are guided by this user input, while a set of feasibility constraints ensures that they are always stable.

Optimization Given the structure and motion goals for a robotic creature, our system computes time-varying motor values for dynamically-stable motions using a trajectory optimization approach. The user can preview the optimized motions using physics-based simulation and iteratively adapt the design to explore the solution space and converge on a desired result. In order to enable this seamless forward design experience, to robustly support a wide variety of morphological features in a unified manner, and to ensure that the resulting robotic creatures function well using off-the-shelf components, the model that we propose requires a departure from conventional approaches.

Space-time optimization methods that consider the full dynamics of the systems they compute motions for are most general. However, the complexity of the underlying models brings about a significant computational overhead—even for simple creatures, generating motions is a matter of several minutes [Wampler and Popović, 2009b; Mordatch et al., 2012; Wampler et al., 2014], thus clearly prohibiting the type of interactive design process we seek to enable. Furthermore, space-time methods are notoriously char-

acterized by challenging optimization landscapes that often lead to undesirable local minima. Our model avoids the complexity of considering full system dynamics for three main reasons. First, we achieve important gains in efficiency, with the process of optimizing motions taking at most a few seconds. Second, in conjunction with the optimization scheme we employ, our system consistently converges to high-quality solutions. Last, because the internal torques and ground reaction forces computed through space-time optimization cannot be directly reproduced by off-the-shelf servomotors, and as physical actuators present considerable limitations in terms of speed, bandwidth and strength, we focus on generating motions that are more conservative (e.g. no flight phases). In this setting, the dynamic interplay between the instantaneous center of pressure and the motion of the center of mass is captured sufficiently well through an inverted pendulum approximation.

The simplified dynamics model we use is adopted from robotics, where it is commonly used for *model predictive control* (MPC). MPC formulations typically decouple the generation of center of mass trajectories, motion of the feet and full-body joint angles [Kajita et al., 2003; Dimitrov et al., 2008; Mastalli et al., 2015]. When the morphology and proportions of a robot are fixed, this strategy is typically sufficient. However, various heuristics are required to ensure that constraints between different modules are satisfied, e.g., stepping locations can be reached given the center of mass trajectory and robot kinematics. Given that our design system allows users to generate robot designs with a vast range of morphological features, such a decoupling is not feasible. Rather, as detailed in section 4.3, our trajectory optimization method provides an efficient, unified formulation for concurrently computing trajectories for the center of mass and feet that are consistent with the structure and range of motion of the robotic creatures.

Finishing Once the design iterations have converged, we automatically generate 3D geometry for all body parts, including connectors for the motors, which are then sent to a 3D printer for manufacturing (see section 4.4).

4.3 Motion Plan Generation

Given a robot morphology which includes an arbitrary number of legs with point or area feet, and possibly an actuated spine, our goal is to compute time-varying motor values that lead to user-controllable, stable motions. We formulate this task as a trajectory optimization problem whereby a set of

objectives is used to define an optimal motion. We represent a *motion plan* $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_T)$ as a time-indexed sequence of vectors

$$\mathbf{p}_i = (\mathbf{q}_i, \mathbf{x}_i, c_i^1, \ldots, c_i^n, \mathbf{e}_i^1, \ldots, \mathbf{e}_i^n\, w_i^1, \ldots, w_i^n)\,, \tag{4.1}$$

where $\mathbf{q}_i$ represents the pose of the creature, i.e., the position and orientation of the root as well as the angle values for all motors, and $\mathbf{x}_i$ denotes the desired position of the creature's center of mass. The feet of each limb are defined by one or multiple end effectors. For each end effector $j, 1 \leq j \leq n$, we use a contact flag c_i^j to indicate whether it should be grounded ($c_i^j = 1$) or not ($c_i^j = 0$) at a given time t_i. Furthermore, we denote the desired position of the end effectors as $\mathbf{e}_i^j$ and store a scalar weight w_i^j for each of them. Fig. 4.4 (left) visualizes our motion plan representation.

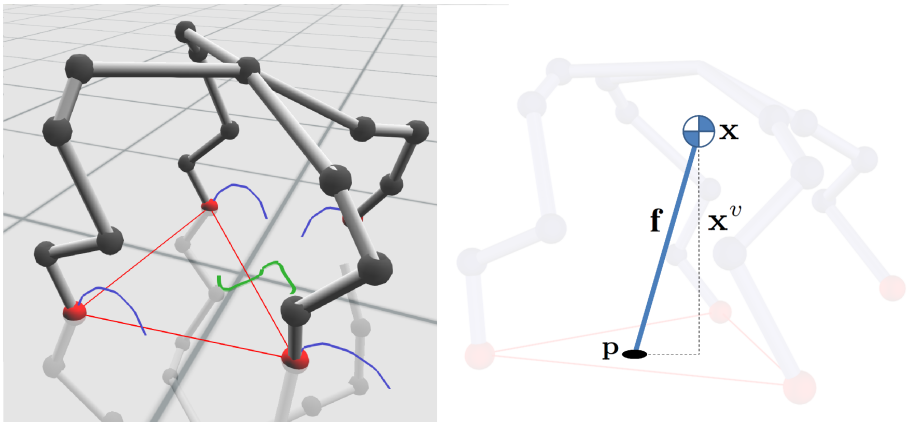


Figure 4.4: *The motion plans generated by our framework consist of trajectories for the center of mass (green), feet (blue), and corresponding full-body poses for the robotic creature (left). An inverted pendulum model is employed to obtain a relationship between the center of pressure and the center of mass (right).*

4.3.1 Constraints

We refer to a motion plan as *consistent* if, for every time sample i, the following set of conditions is satisfied:

$$\varphi_{\text{CoM}}(\mathbf{q}_i) - \mathbf{x}_i = \mathbf{0}\,, \tag{4.2}$$

$$\varphi_{\text{EE}}(\mathbf{q}_i)^j - \mathbf{e}_i^j = \mathbf{0}\,, \quad \forall j\,. \tag{4.3}$$

Here, $\varphi_{\text{CoM}}(\mathbf{q})$ is a forward kinematics function outputting the position of the creature's center of mass (COM) given pose $\mathbf{q}$. Similarly, the function $\varphi_{\text{EE}}(\mathbf{q})$ computes the end effector positions for the robot's limbs given pose $\mathbf{q}$.

In order to generate motions that do not require sophisticated sensors and complex feedback mechanisms, our framework computes motion plans that are naturally stable. Formally, this criterion is expressed as a constraint that ensures that at every discrete moment in time i, the *center of pressure* (COP) $\mathbf{p}_i$ falls within the support polygon defined by the end effectors that are in contact with the ground:

$$\sum_{j=1}^{n} c_i^j w_i^j \mathbf{e}_i^j - \mathbf{p}_i = 0 \,, \tag{4.4}$$

with the additional constraint that only convex combinations of grounded end effector positions are allowed to define the location of the COP:

$$\sum_{j=1}^{n} w_i^j c_i^j = 1 \,, \quad w_{lb} \leq w_i^j \leq 1 \,, \forall j \,, \tag{4.5}$$

where the lower bound limit w_{lb}, which is set to 0.1 for all our experiments, prevents the COP from moving too close to the boundary of the support region.

The COP position $\mathbf{p}_i$ at each time step i is not an explicit parameter of the motion plan. However, using an inverted pendulum model, a simple relationship between it and the optimized COM trajectory can be readily obtained [Kajita et al., 2003]. As illustrated in Figure 4.4 (right), the vertical component of the force $\mathbf{f}$ applied along the vector between the COM and the COP is $\mathbf{f}^v = mg + m\ddot{\mathbf{x}}^v$, where $g = 9.8m\backslash s^2$. Consequently, by computing the horizontal component of $\mathbf{f}$, and by observing the trigonometric relationship between the height of the COM, $\mathbf{x}^v$, and the horizontal projection of the vector from $\mathbf{p}$ to $\mathbf{x}$, the following relationship emerges:

$$\mathbf{p}_i = \mathbf{x}_i - \frac{\mathbf{x}_i^v \ddot{x}_i}{\ddot{\mathbf{x}}_i^v + g} \tag{4.6}$$

The acceleration of the COM trajectory, $\ddot{\mathbf{x}}_i$, including its vertical component, is expressed using finite differences as a function of $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$: $\ddot{\mathbf{x}}_i = (\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1})/h^2$, where h is the time step. We note that Equation 4.4 represents a *dynamic stability* criterion: while the COP is guaranteed to lie within the support polygon at all times, the projection of the COM on the ground plane does not have to. Consequently, the motions generated by our optimization framework are less conservative than if a static stability criterion acting solely on the COM was employed.

As the positions of the end effectors are independently defined at discrete moments in time, an additional constraint is needed in order to ensure temporal consistency of the motion plan and to avoid foot-slipping:

$$(\mathbf{e}_{i-1}^{j} - \mathbf{e}_{i}^{j})c_{i}^{j} = \mathbf{0}\ , (\mathbf{e}_{i}^{j} - \mathbf{e}_{i+1}^{j})c_{i}^{j} = \mathbf{0}\ , \tag{4.7}$$

for all $2 \leq i \leq T-1$. This expression implies that the target positions of the end effectors are only allowed to change freely when they are not in contact with the ground.

If a periodic motion is desirable, an additional constraint that relates the robot's joint angles, $J(\mathbf{q})$, at the start and end of the motion is added:

$$J(\mathbf{q}_1) - J(\mathbf{q}_T) = \mathbf{0}\ . \tag{4.8}$$

The robotic creatures generated with our system reproduce the planned motions by directly tracking the optimized joint angle trajectories. It is therefore crucial to ensure that the resulting motion plans fall within the range of capabilities of the physical motors that are used. As a simplified actuation model, we place bounds on the maximum angular velocity at each joint j:

$$-\omega_{max} \leq \frac{J(\mathbf{q}_{i+1})^{j} - J(\mathbf{q}_{i})^{j}}{h} \leq \omega_{max}, \forall i, \tag{4.9}$$

where ω_{max} is the maximum achievable speed of the motor.

4.3.2 Motion Style Objectives

If Equations 4.2 and 4.8 are satisfied, the motion plan is termed *admissible*, as it corresponds to a stable motion. In general, many such motion plans exist for a given robot morphology and footfall pattern. We therefore provide several objectives and high-level controls that allow users to intuitively explore the space of admissible motions.

The smoothness of the motion is the first attribute that users can control via an objective defined through a finite difference approximation of the second derivatives of the robot pose trajectory:

$$E_{\text{Smooth}} = \frac{1}{2}\sum_{i}^{T} ||\mathbf{q}_{i-1} - 2\mathbf{q}_{i} + \mathbf{q}_{i+1}||^{2}\ . \tag{4.10}$$

With the motion editing interface provided by our system, users can also directly influence the motion style of the robotic creatures they design. To

this end, two regularizing terms provide target trajectories for the motion of the COM and the end effectors:

$$E_{\text{StyleCOM}} = \frac{1}{2}\sum_{i=1}^{T} ||\mathbf{x}_i - \mathbf{x}_i^D||^2 \tag{4.11}$$

$$E_{\text{StyleEE}} = \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{n} ||\mathbf{e}_i^j - \mathbf{e}^{D}{}_i^j||^2 \tag{4.12}$$

We note that although the target trajectories may lead to unstable motions if employed directly, they are used just as a guide. Consequently, the user can edit them freely, without needing to worry about feasibility constraints, in order to uncover the range of motion styles achievable by the robot they are designing.

The walking and turning speed of the robotic creatures are controlled through two separate terms which measure the difference between the pose of the robot at the start and end of the motion trajectory:

$$\mathbf{x}_T - \mathbf{x}_1 = \mathbf{d}^D \tag{4.13}$$

$$\tau(\mathbf{q}_T) - \tau(\mathbf{q}_1) = \tau^D \tag{4.14}$$

where $\mathbf{d}^D$ and τ^D are the desired values for the net distance traveled and turning angle, respectively, and the function $\tau(\mathbf{q})$ returns the turning angle from pose $\mathbf{q}$.

4.3.3 Optimization

We cast the problem of generating feasible motion plans as a multi-objective, constrained optimization problem where the degrees of freedom are the robot poses at each time instance, the trajectories of the end effectors and the center of mass, and the end effector weights which define the trajectory of the center of pressure. The contact flags are directly specified by the footfall pattern and are thus not treated as free parameters during the optimization. As described above, we have structured the conditions on the robot's motion into constraints that model vital requirements for successful locomotion, and objectives that influence the style of the motion. When translating these terms into an optimization problem however, we have to carefully balance the importance of exact constraint satisfaction against the numerical difficulties associated with non-linear, non-convex systems of equations. We therefore treat the nonlinear Equations 4.2 and 4.4 as soft constraints by

minimizing their squared residuals weighted by a large constant (10^4 for all experiments). The linear equality and inequality relations described by Equations 4.5, 4.7, 4.8 and 4.9 are treated as hard constraints. The weights associated with the motion style objectives 4.10 and 4.14 can be interactively set by the users of our system to emphasize different priorities they might have. However, they are kept constant for all our experiments. To minimize the resulting constrained optimization problem we use OOQP [Gertz and Wright, 2003], as the inner loop of our Sequential Quadratic Programming solver (see Section 3.2.3.

As can be seen in Table 4.1, generating motion plans requires our framework to solve non-linear systems of equations with hundreds of unknowns—the exact number depends on the complexity of the robot's design and it is linear in the number of time samples that define the motion trajectories. Given that analytic derivatives for the constraints and objectives we formulate can be readily computed, and because the resulting Hessians are sparse, computing optimal motions can be done efficiently, typically requiring less than $3 - 5s$ of computation time when starting from scratch. During the iterative design process, as the user adjusts the proportions or motion style of their robot, previously computed motion plans are used to warm-start the optimization process, thus further reducing the computational burden and leading to a seamless interactive experience.

To increase convergence rates when optimizing a motion plan from scratch we employ a two-step optimization process. As illustrated in Figure 4.5, which shows the value of the objective while generating in-place walking motions for our *Ranger* robot, 10 iterations are first executed while the end effector trajectories are kept constant (546 parameters in total), followed by additional iterations where the full set of parameters (714) is optimized for. In contrast to a typical scheme where all parameters are optimized from the beginning, we observe improvements in convergence rates of up to 50%. This performance improvement is due to Equation 4.4 becoming convex, which gives rise to a smoother optimization landscape where intermediate solutions can be efficiently arrived at. These intermediate solutions then furnish a good starting point for the global optimization which quickly convergences to a nearby local minimum. Each step of the global optimization takes on average $0.08s$.

4.4 Generation of 3D Printable Mechanical Structures

The designs thus far produced by our system provide only an abstract description of the robotic creatures our users intend to create. In particular, the

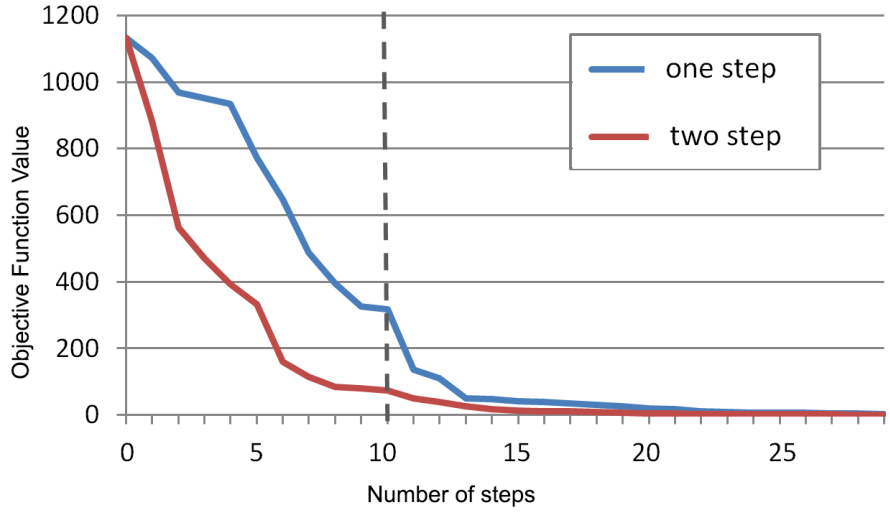
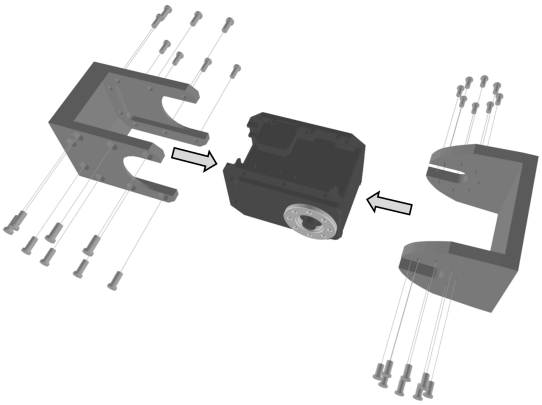


Figure 4.5: *Convergence plot showing the value of the objective function while optimizing all parameters at once (blue) versus a two-step optimization scheme (red).*

placement and orientation of virtual motors, the relative location of the end effectors and the skeletal structure that specifies their connectivity define the morphology and body proportions of the designs. Before proceeding to fabrication, our system automatically generates 3D printable geometry for the robot's mechanical structure. To this end, our system starts with a CAD model of available servomotors, as visualized in the inset figure. In a pre-processing step we generate two tight-fitting 3D models that directly attach to the servomotors with screws or rivets, enclose them, and become part of the geometry generated by our system. The geometry of the enclosing models allows for a ± 90 degrees range of motion for each servomotor.



We collectively refer to the attachment parts of each servomotor and the models that define the geometry of the end effectors as *structural features*. The problem of creating the mechanical design a robot's body parts thereby reduces to generating geometric models that connect consecutive pairs of structural features sf_i and sf_j. We note that mounting brackets are typically used for this purpose. However, as the shape and functionality of mounting brackets are pre-defined such that they can be mass-produced, they signifi-

cantly restrict the set of possible relative positions and orientations between pairs of structural features. We therefore opt to create custom, 3D printable connections instead.

As illustrated in Figure 4.6 a), each structural feature outputs a set of possible attachment points (green). As a step towards generating a geometric model that connects a pair of structural features, our system computes a mapping between the attachment points output by sf_i and sf_j. More specifically, we first compute the convex polygons of the attachment points projected on a plane perpendicular to the vector between sf_i and sf_j. We then compute the convex hull of the attachment points whose projections define the two convex polygons, as seen in Figure 4.6 b). If fabricated using a *Fused Filament Fabrication* device, where the infill rate can be used to control the trade-off between weight and structural integrity, then the geometry of the body part can be obtained by performing a union operation between the computed convex hull and the models that enclose the servomotors. However, for other 3D printing techniques, such as *Selective Laser Sintering*, our system can automatically create a lightweight structure inspired by engineering principles. More specifically, our system generates truss structures directly from the convex hull computed at the previous step. The edges of the convex hull that connect one attachment point on sf_i to another on sf_j define the main elements of the structure, and additional connective struts are added procedurally, as shown in Figure 4.6 c). The density of the connective struts and their thicknesses are user-specified, as they depend on the material used for fabrication.

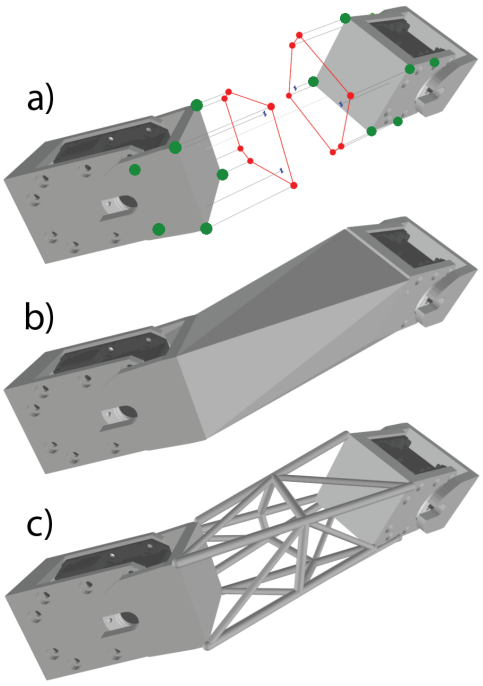


Figure 4.6: *Generating 3D-printable geometry.*

The geometric structures automatically generated by our framework are functional and they allow the robotic creatures designed with our system to be fabricated through 3D printing. However, in order to enhance the aesthetics of the designs, we allow users to augment the generated geometry with existing 3D models, if desired. To accomplish this, users position the existing 3D models relative to the desired robot body part, and a union

operation is performed to generate the fused mesh. As a simplification, the added 3D models are assumed to not significantly alter the mass distribution of the robotic creature (i.e., they are lightweight shells). If this assumption is not valid, it is trivial to recompute the mass and moment of inertia of each body part based on the user-provided geometry and re-optimize the motion plans.

4.5 Results and Discussion

We have used our interface to design a diverse set of robotic creatures, three of which were physically fabricated for validation. The results that we present in this section demonstrate that our method is indeed a powerful tool, allowing users to author a broad range of designs with explicit and intuitive control over morphology and motion style. Our design methodology offers a number of key benefits over the alternative of manual design. In particular, while keyframing and other conventional methods are very successful for digital animation, the motions of our robotic creatures are subject to real-world physics. Anticipating and incorporating these effects during the design process is very difficult, even for experts. The gaits generated using our motion optimization, in contrast, precisely coordinate the movements of the feet and the body such as to ensure smoothness and stability for robots of various designs. Consequently, our system allows for an intuitive exploration of the relationship between a robot's morphological features and its ability to produce compelling, purposeful motions.

Below we highlight several aspects that are key to our approach and discuss observations from experimental validation.

4.5.1 Design Interface & Workflow

Structure Editing Thanks to the fast turnaround rates of the underlying optimization, our interface allows for quick, easy, and intuitive editing of a creature's structure and motion. The user can freely edit the morphology of the robot by dragging on motor handles until the predicted motion—always visible and up-to-date in the preview viewport—is satisfying. For these edits in particular, convergence is very good since we can warm-start the optimization with a previously optimized motion plan. However, when changing axes of rotation, the joint-angle trajectories used for warm-starting can lead to drastically different trajectories of the end effectors. We therefore simply re-initialize the optimization process, which then takes about 3

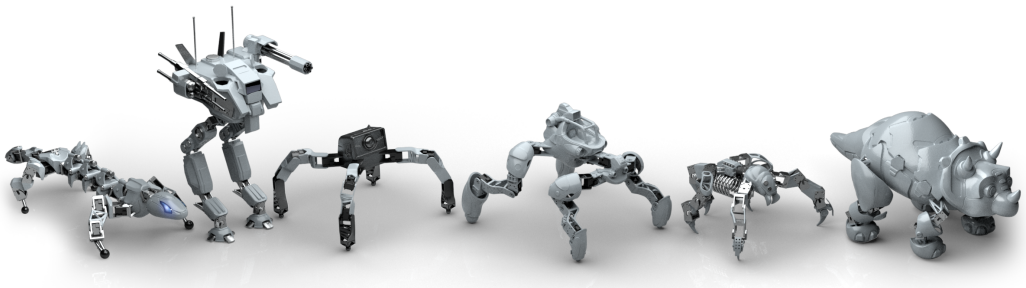


Figure 4.7: *Six robotic creatures designed with our interactive system: one biped, four quadrupeds and one five-legged robot.*

seconds on average to arrive at a solution. We also restart the optimization process from scratch when the morphology of a design is altered by adding or removing motors.

Morphology Editing A particularly compelling feature of our method is that it supports arbitrary morphologies. As demonstrated by our results, we support creatures with arbitrary numbers of legs, including bipeds, quadrupeds, and more exotic cases such as the five-legged creature shown in Figure 4.3. Furthermore, the configuration of the legs is very flexible: the robotic creatures we show are designed with three or four motors per leg, while our biped robot (Figure 4.7) also features actuated ankles for a total of five motors for each leg. Area or point feet are specified by designating one or more end effectors for each limb. Actuated bodies are also supported: our *Ranger* (Figure 4.1) and *Predator* (Figure 4.3) robots each have one motor that allows their shoulders to rotate relative to their pelvis, while the *Salamander* design (Figure 4.8) uses two actuators for the tail and five for its flexible spine. It is worth noting that this generality is a direct consequence of our formulation—no special treatment is required.

Motion Editing Our system offers three sets of tools that allow the user to author and edit the motion of a given robot in largely orthogonal dimensions: the footfall pattern, the velocity of the center of mass, and trajectories for the feet and the center of mass. We first tested these tools designing a range of motion edits on the *Bobby* model (Figure 4.2), a quadruped robot with 12 motors and thus similar in complexity to commercially-available mid-range robots. The footfall pattern graph allows the user to quickly explore different gaits, simply by dragging on the stance/swing phase widgets of the individual legs. The different gaits are also well-reflected by the physical prototype. High-level motion goals are formulated simply by prescrib-

ing constant linear or angular velocity for the body of the robot to achieve forward or sideway motion, or to turn in place. Once a rough motion has been laid out, the trajectory editing tool can be used to further flesh out the characteristics of the gait such as to give it personality and style.

Creature Finishing Our method automatically generates geometry for the various body parts based on the shape and location of the motors and end-effectors. However, in order to increase the aesthetic appeal of the final designs, the user can replace these meshes or augment them with manually designed shells. In that case, we assume that the mass and inertia properties for the new parts do not significantly deviate from the old geometry such that the robot can walk as expected with the original animation. We note that it is always possible to recompute angle trajectories in order to account for changed mass properties, but if these changes are significant, the optimization might have to substantial alter the style of the resulting motion.

On-Board Control After the robotic creatures are fabricated and assembled, off-the-shelf servos are used to drive their motions. We use a combination of position and velocity control to ensure that the servo motors produce smooth motions and remain in sync with each other. Control signals are computed at fixed time intervals. Briefly, for motor i at time t we estimate the target joint position $q^i(t + \delta t)$ by interpolating the corresponding optimized motion trajectory, and read the servo's current position, $\alpha^i(t)$. The control board then sets the motor's maximum angular speed

Figure 4.8: Salamander*: our framework automatically generates swaying tail and spine motion, leading to a natural-looking gait.*

to $(q^i(t+\delta t) - \alpha^i(t))/\delta t$, while its goal position is set to $q^i(t+\delta t)$. We use $\delta t = 0.15s$ for all our experiments.

4.5.2 Validation

We performed a set of experiments in order to assess the feasibility of the motion plans generated by our method. First, we employ black-box physics simulations as a way of providing a preview of the expected real-world behavior of each robot design. We use the Open Dynamics Engine [ODE, 2007] for this purpose and model the robots as articulated rigid body systems. The joint trajectories computed during the motion optimization stage are used to define time-varying targets for Proportional-Derivative controllers, used to model the actuators at each joint.

In order to achieve motion planning at interactive rates, our optimization scheme uses an approximate dynamics model. More concretely, asking the center of pressure to fall within the support polygon is a necessary but not sufficient condition, as it ignores the friction-induced limitation on the tangential contact forces and dynamic effects that may become significant for fast limb movements. In order to assess the impact of this simplification, we measured the error in tracking the planned center of mass trajectories over a full motion cycle for our *Ranger* model. We ran this experiment on three different motions with stride durations of 0.5, 1 and 2 seconds, and observed a net final error in center of mass position of 1.57, 0.62 and $0.18cm$ respectively, when comparing the planned motion against the result of the simulation. As a point of reference, each limb of this robotic creature is about $50cm$ in length and it was tasked with walking forward using step lengths of $20cm$. Although a growing discrepancy from the motion plan becomes apparent as the speed of the motion increases, the simulated robot was able to walk successfully each time. However, for biped designs like *Hunter*, such errors can lead to failures due to the high COM and comparatively small area of the support polygon. Physics simulations will immediately reveal such unwanted behaviors, allowing the designer to take appropriate measures by adjusting their design.

To further validate the results of our system, we created physical prototypes for the *Bobby*, the *Ranger*, and the *Predator* characters. We used 3D-printed body parts and off-the-shelf hardware—12 *Dynamixel MX-28* actuators daisy-chained to a *CM-700 Robotis Servo Controller* board and powered by a LiPo battery. Even before comparing the motion predicted in simulation to the results observed on the real-world protoypes, we can identify several sources of inaccuracy resulting from idealizing assumptions made

by the simulation model. In particular, we assume ideal actuators and perfectly rigid body parts. In reality, the amount of torque that motor can exert is limited, but the maximum torque also decreases with increasing angular velocity—a complex behavior that is more difficult to model. Furthermore, while 3D-printing allows for quick and easy customization of the robot's geometry, the limited accuracy of current consumer-end printers together with the finite compliance of the printed body parts leads to deformations that are not accounted for in the simulation. As a concrete manifestation, we observed that the feet of the *Ranger* model do not rise as high as seen in simulation (approximately $8cm$ measured at the apex vs. $10cm$ in simulation). Despite these sources of error, we observed good agreement between the overall motions of our physical prototypes and the behavior predicted in simulation.

Finally, it should be noted that it takes on the order of minutes to the design these creatures, but hours to assemble and even days to print. This fact implies that building prototypes is very time-consuming and expensive—and it is the ambition of our method to produce final *digital* designs without the need for *physical* iterations.

4.6 Limitations and Future Work

This chapter presented an interactive, end-to-end solution for designing 3D-printable robotic creatures whose morphology, proportions, gaits and motion styles can be easily personalized. Using our system, we were able to design a diverse set of legged robots, each created in a matter of minutes. Our method efficiently generates stable, user-controllable walking motions for robots with a vast array of morphological features. The most immediate benefit of our system is therefore that it enables an interactive exploration

	# Legs	Spine DOFs	# motors	# motion plan parameters
Bobby	4	0	12	429
Dino	4	0	14	735
Salamander	4	8	20	861
Ranger	4	1	13	714
Hunter	2	0	10	387
Predator	5	1	16	840

Table 4.1: *An overview of the complexity of robot designs we created with our system.*

of the space of feasible robot designs without requiring any domain specific knowledge from its users.

Although the space of robots and motions that our solution can currently generate is substantial, our system does have limitations that present exciting avenues for future work. For example, motions that exhibit flight phases cannot currently be produced, as our model requires the existence of at least one point of contact between the robot and the ground at any moment in time. Furthermore, the model simplifications that we exploit for efficiency result in differences between the predicted and real-world behavior of the robots. These differences are most pronounced for fast motions and increase as the size of the support polygon decreases relative to the height of the center of mass. For instance, the bipedal robot that we designed is capable of walking, but unsurprisingly, it can fall quite easily when perturbed. This limitation highlights the need to integrate feedback mechanisms that adapt the motion plan in real-time based on sensor data.

We are also excited about the challenge of making the process of authoring complex behaviors easily accessible to casual users. We are encouraged by the ease with which gaits and motion styles can be specified using our easy-to-use editing tools. We plan to extend these tools such that users can specify a motion repertoire that includes switching between gaits, portraying rich personalities and interacting appropriately with objects and humans. Finding appropriate abstractions for intuitively authoring such high-level behaviors is an interesting subject for future work.

CHAPTER 5

A Computational Design Tool for Compliant Mechanisms

In this chapter we present a computational tool for designing compliant mechanisms. In contrast to the rigid limbs characteristic of the creatures designed in the previous chapter, these devices perform their motions through flexible articulations. The design approach we propose, takes as input a conventional, rigidly-articulated mechanism defining the topology of the compliant design. This input can be both planar or spatial, and we support a number of common joint types which, whenever possible, are automatically replaced with parameterized *flexures*. As the technical core of our approach, we describe a number of objectives that shape the design space in a meaningful way, including trajectory matching, collision avoidance, lateral stability, resilience to failure, and minimizing motor torque. Optimal designs in this space are obtained as solutions to an equilibrium-constrained minimization problem that we solve using a variant of sensitivity analysis. We demonstrate our method on a set of examples that range from simple four-bar linkages to full-fledged animatronics, and verify the feasibility of our designs by manufacturing physical prototypes.

5.1 Introduction

Engineers routinely design for strength and stiffness. Steel and concrete prevent deflections in buildings, and machines resort to rigid articulation in order to avoid deformations. But although most human designs are inspired

by Nature, rigidity is a concept foreign to the living world: from a kangaroo's legs to the wings of a bat—bones, tendons, and cartilage are the nuts and bolts of organic machines, and deformation is an integral part of the design, crucial for both efficiency and robustness. Unfortunately, designing for flexibility requires deep understanding and precise predictions of finite deformations, which proves to be substantially more difficult than relying on rigidity.

Fueled by progress in technology and computation, however, many fields of engineering have started to *embrace deformation* and to leverage flexibility for better, more elegant, and ultimately more satisfying designs. Applied to machines, this turn to the flexible leads to *compliant mechanisms*, i.e., mechanical devices that perform motion not through rigid articulation but by virtue of elastically deforming flexures. Compliant mechanisms enjoy widespread use in industry, where they are valued for their accuracy, ease of manufacturing, scalability, and cost efficiency. The spectrum ranges from specialized microelectromechanical systems (MEMS) for miniature sensors and actuators [Kota et al., 2001], to more mundane devices including monolithic pliers and wiper blades, and to commonplace products such as binder clips, backpack latches, and shampoo lids.

In this chapter, we are primarily interested in exploring the potential of compliant mechanisms for personalized automata and animatronics. With the ability to create complex geometry and its repertoire of flexible, plastic-like materials, 3D printing is an ideal way of manufacturing compliant mechanisms. And thanks to the increasing availability of consumer-level printers, hobbyist mechanics and other non-expert users now have the machinery to create compliant mechanisms for use in their conceptions and contraptions. But perhaps even more than for conventional mechanisms, the path to a successful compliant design is littered with traps for the novice:

- Compliant mechanisms typically involve large deflections that give rise to nonlinearities in both geometry and material behavior, not rarely betraying intuition.
- For conventional mechanisms, the resistance to motion is either zero or infinite. In the compliant setting, any motion requires a finite amount of work and, depending on direction, the stiffness can vary by orders of magnitude. Shaping the corresponding energy landscape, i.e., finding a balance between stiffness and flexibility is one central aspect of this design problem.
- Compliant mechanisms provide friction- and wear-less motion, but

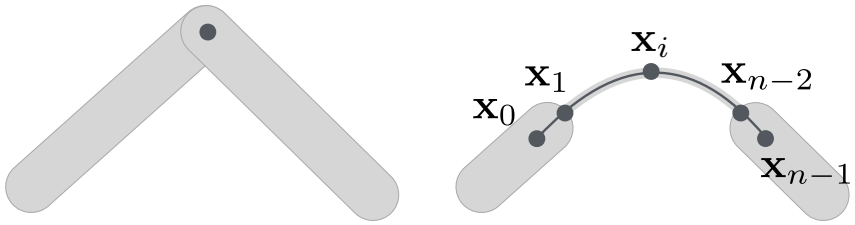


Figure 5.1: ***Conventional vs. compliant hinge.*** *We replace conventional joints (left) with a single or several flexures (right).*

incautious design can induce material fatigue and failure. In order to minimize this risk, high stress concentrations must be avoided.

- While the *forward* problem of predicting the motion of a compliant mechanism is a non-trivial task already, the *inverse* problem of determining parameter values that lead to a desired motion or function is extremely difficult.

Considering these challenges, designing compliant mechanisms is all but a hopeless endeavor for casual users.

5.2 Computational Model

The core of our method is formed by a computational model that allows us to simulate the behavior of a given compliant mechanism design (Section 5.2.1). The design itself is described in terms of a dedicated flexure-centric parameterization (Section 5.2.2), defining the interface to subsequent optimization (Section 5.3).

5.2.1 Simulating Compliant Mechanisms

We model compliant mechanisms as sets of rigid links interconnected by flexible joints. The state of each rigid link is represented as a vector $\mathbf{s}_i \in \mathbb{R}^6$ holding translational and rotational degrees of freedom. Each compliant joint is composed of a set of flexures—thin lamella which we model using discrete elastic rods 3.1.3. A flexure is represented by a piece-wise linear centerline, given as a set of vertices $\mathbf{x}_i$, as well as a set of radii, a_j and b_j, defining the width and height of the elliptical cross section for each edge of the centerline.

In addition to compliant flexures, our approach also supports conventional joints, which are used, e.g., when full revolutions are required. Following

Coros and colleagues [2013] we model conventional joints using simple geometric constraints $\mathbf{C}(\mathbf{s})$.

Our compliant mechanisms are actuated by connecting one or several of the rigid links to an input driver such as a motor or a crank. In order to obtain proper two-way coupling, we ask that the first and last two centerline vertices (compare with Figure 5.1 right) move rigidly with the incident links $\mathbf{s}_i$ and $\mathbf{s}_j$, which we implement by eliminating the corresponding degrees of freedom. The motion of the input link then propagates throughout the mechanism, causing flexures to stretch, bend, and twist. These deformations give rise to internal energy $E_{\text{int}}(\mathbf{x})$ that is computed according to [Bergou et al., 2008].

Given the states of all input links, we can compute the equilibrium configuration of the mechanism by solving the constrained optimization problem

$$\min_{\mathbf{x},\mathbf{s}} \; E_{\text{int}}(\mathbf{p},\mathbf{x},\mathbf{s}) \quad \text{s.t.} \quad \mathbf{C}(\mathbf{p},\mathbf{s}) = \mathbf{0}\,, \tag{5.1}$$

where $\mathbf{x}$ and $\mathbf{s}$ collect the degrees of freedom of all flexures and links, respectively, and $\mathbf{p}$ is a vector of design parameters, defining the rest state of the mechanism as described next.

5.2.2 Parameterizing Compliant Joints

Given a conventional mechanism as input, our goal is to replace rigidly-articulated joints with compliant counterparts wherever possible and desired. To this end, we can in principle choose from existing catalogs [Howell et al., 2013] that provide designs for many types of conventional joints, typically with several alternatives for each type. When choosing a particular compliant joint, we ask that the range of motion of the original joint be preserved as much as possible, including degrees of freedom as well as constraints. Moreover, we would like the compliant joints to be readily parameterized and easy to manufacture.

Revolute Joints As the most frequently found joint in planar and spatial linkages, hinges can, in principle, be modeled using only a single flexure. By choosing an elliptical cross section that is wide along the rotation axis $\mathbf{a}_i$ of the original joint, but thin in the orthogonal direction, high lateral stability can be achieved with small in-plane stiffness. However, the ratio between in-plane compliance and out-of-plane stiffness of a design with multiple antagonistic (i.e., 'crossing') flexures can be significantly higher for the same

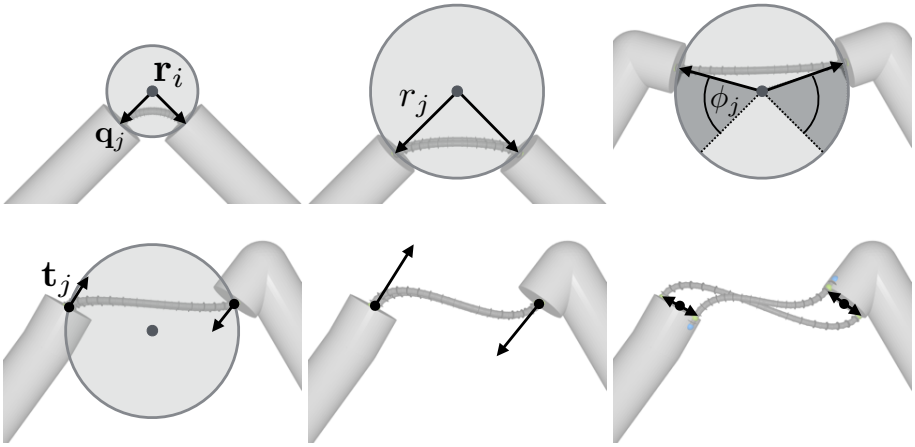
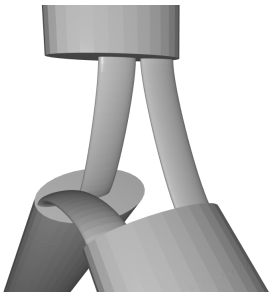


Figure 5.2: *Parameterizing a compliant hinge with two offset flexures.*

total width. We therefore adopt a layered flexure design that, as illustrated in Figure 5.2, consists of two flexures stacked along $\mathbf{a}_i$. In order to allow for smooth geometry with a minimum number of parameters, we model each flexure as a cubic Hermite spline, defined by two end points $\mathbf{q}_j$ and corresponding tangent vectors $\mathbf{t}_j$. The flexure is required to remain within its layer, which we achieve by using polar coordinates to parameterize the attachment points with respect to the location $\mathbf{r}_i$ of the original joint as

$$\mathbf{q}_j = \mathbf{r}_i + r_j(\cos\phi_j \mathbf{e}_1 + \sin\phi_j \mathbf{e}_2) \,. \tag{5.2}$$

In the above expression, r_j and ϕ_j are radial and angular coordinates as illustrated in Figure 5.2 (top, middle) and (top, right), and $[\mathbf{e}_1, \mathbf{e}_2]$ span the plane orthogonal to $\mathbf{a}_i$. Tangent vectors are described analogously (bottom, left) and (bottom, middle), and two additional parameters control the distance between the attachment points of the two flexures on the same link, such as to create, e.g., cross configurations with high lateral stability (Figure 5.2, bottom right). The layered design also generalizes readily to hinge joints connecting more than two components as illustrated in the inset on the left with a three-way coupling, where an additional layer is required to accommodate all three flexures.



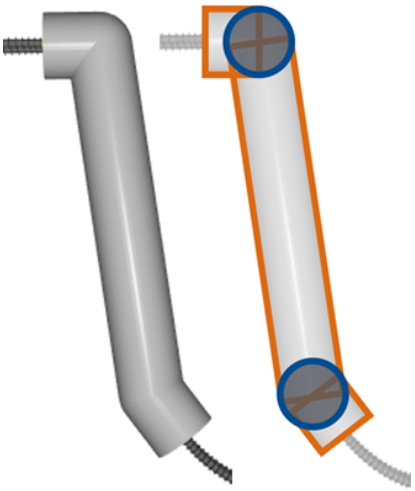
Other Joint Types For spherical joints, we use a single flexure with circular cross section (compare with Figure 5.3 left), offering comparatively low resistance to bending and twisting deformation, but orders of magnitude higher stiffness for stretching. Similar to hinge joints, flexures for spherical

joints are modeled as spline curves whose attachment points are required to remain within a certain spherical volume relative to the original joint location.

In addition to revolute and spherical joints, compliant universal joints can be modeled as well using, e.g., two flexures with mutually orthogonal axes connected serially as illustrated in Figure 5.3 (right). These three joint types allow us to create a large and diverse set of meaningful linkages and other mechanical assemblies. Note that connections involving translational motion are inherently difficult to achieve using compliant joints. However, our formulation supports rigidly-articulated versions of these joints, both in terms of simulation and optimization.

5.2.3 Generating Link Geometry

With a view to design optimization, we collect the parameters of all flexures in a vector $\mathbf{p}$. Taken together, these parameters completely define the undeformed configuration of the compliant mechanism. In particular, they define the rest state geometries for all flexures which, in turn, determine the behavior of the mechanism. In order to obtain a functional, printable mechanism, we have to generate geometry for all rigid links, and this geometry should automatically adapt to changes in the flexure parameters during optimization. To this end, we represent the geometry for each link as a union of capsules, a representation that is readily parameterized and simplifies collision tests (see Section 5.3.4). More concretely, we model each link as a union of three cylinders and two spheres, as illustrated in the inset figure. Two of the cylinders have fixed lengths and attach to the flexures on either side of the link. These two cylinders are, in turn, connected by a third cylinder of variable length, and we place spheres at the corresponding intersection points in order to obtain smooth transitions. Though simple, this procedural link geometry has the advantage that derivatives with respect to flexure parameters, which are required for design optimization, are readily computed.



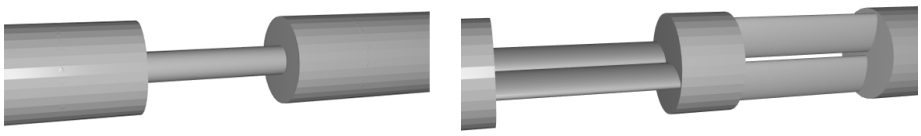


Figure 5.3: *Compliant ball-and-socket (left) and universal joint (right).*

5.3 Design Optimization

Given an initial design for a compliant mechanism, we seek to find parameter values for all flexures such as to optimize the function of the mechanism with respect to various objectives, including motion tracking, ease of actuation, and resilience to failure. We first introduce the individual objectives, then describe how to compute optimal parameter values.

5.3.1 Motion Tracking

Generating motion is a central function of both conventional and compliant mechanisms. For example, it is crucial for the end effector in a compliant leg mechanism to closely follow the corresponding trajectory of the conventional counterpart in order to ensure proper walking. Similarly, the finger of a compliant robot hand needs to offer the same range of motion to successfully perform grasping tasks. In order to encourage accurate motion approximation, we introduce a trajectory matching objective of the form

$$f_{\text{track}} = \frac{1}{2} \sum_t \left\| \mathbf{z}_t(\mathbf{x}_t, \mathbf{s}_t) - \hat{\mathbf{z}}_t \right\|^2 , \tag{5.3}$$

where $\mathbf{z}_t$ describes the discrete trajectory of a point on the compliant mechanism, and $\hat{\mathbf{z}}_t$ is the corresponding target trajectory on the input mechanism. We step through a full motion cycle or along a user-specified animation and compute the corresponding equilibrium states $(\mathbf{x}_t, \mathbf{s}_t)$ by minimizing (5.1).

5.3.2 Lateral Stability

A conventional mechanism exhibits zero resistance to motion corresponding to its degrees of freedom, and infinite stiffness in all orthogonal directions. However, this crisp picture becomes somewhat blurred for compliant mechanisms, which exhibit finite resistance to motion in all directions. In particular, instead of deriving from a degree of freedom, the motion of a compliant mechanism is a path through an energy landscape, lined with preferably steep walls to the sides, but sloping rather gently along its direction. Low

resistance to motion along this path is desirable as this leads to less stringent torque requirements. While some amount of compliance perpendicular to the trajectory can be desirable as well (e.g., for grasping or walking), a certain minimum stiffness to lateral motion is always required. One particular way of encouraging lateral stability is to apply a given force $\mathbf{f}_\text{l}$ to the mechanism and ask that the resulting displacement be minimal or bounded. Using the motion tracking objective (5.3) as a basis, the deviation from the original trajectory can be expressed as

$$f_\text{stab} = \frac{1}{2} \sum_t \|\mathbf{z}_t(\mathbf{x}_t, \mathbf{s}_t) - \tilde{\mathbf{z}}_t(\mathbf{x}_t, \mathbf{s}_t)\|^2 \,, \tag{5.4}$$

where $\tilde{\mathbf{z}}_t$ is the trajectory obtained under the action of $\mathbf{f}_\text{l}$. We generally apply $\mathbf{f}_\text{l}$ at the end effector in a direction approximately normal to the trajectory, but the user is free to change this direction if desired.

5.3.3 Actuation Requirements

Actuating a compliant mechanism to a given target configuration requires work, holding the position requires torque. Ideally, we would like to minimize the maximum required torque along the trajectory, as this will allow for more precise actuation and for using smaller, less expensive motors. The torque required to hold a compliant mechanism in a given position is equal to the derivative of its elastic energy with respect to the motor angle. Using central differences, we approximate the torque required to sustain a given configuration $(\mathbf{x}_t, \mathbf{s}_t)$ as

$$\tau_t = \frac{1}{\Delta\alpha_t} \left(E(\mathbf{x}_{t+1}, \mathbf{s}_{t+1}) - E(\mathbf{x}_{t-1}, \mathbf{s}_{t-1}) \right) \,, \tag{5.5}$$

where $\Delta\alpha_t$ is the corresponding change in motor angle. In order to minimize torque requirements, we define an objective function as

$$f_\text{act} = \frac{\sum_t \tau_t^2 e^{\beta_\text{act} \tau_t^2}}{\sum_t e^{\beta_\text{act} \tau_t^2}}, \tag{5.6}$$

where we set β_act to a large positive value.

5.3.4 Avoiding Collisions

In order to ensure proper functioning of the mechanism, we have to avoid collisions between its individual links and flexures. There are three cases

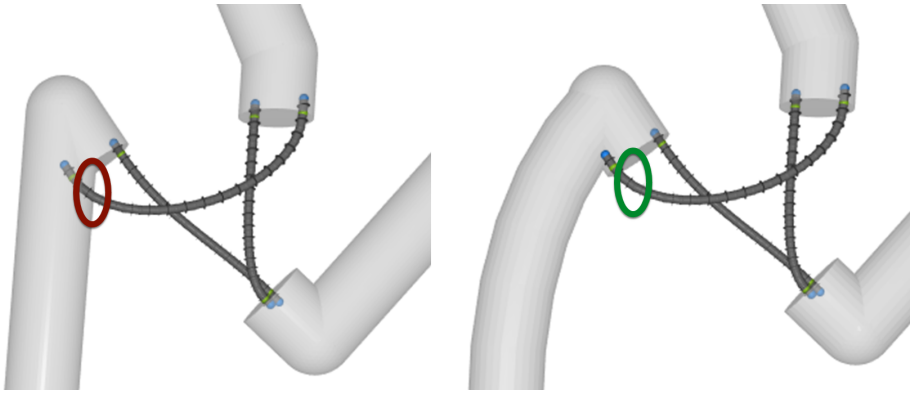


Figure 5.4: *A collision between a flexure and a link* (left) *is resolved by reshaping the corresponding geometry* (right).

that need to be handled: flexure-flexure, link-link, and flexure-link collisions. In order to prevent flexure-flexure intersections, we measure the distance between all pairs of edges of the two centerlines. Whenever the distance $d(\mathbf{e}_i, \mathbf{e}_j)$ between two edges $\mathbf{e}_i$ and $\mathbf{e}_j$ is less than a minimum distance d_{min}, we construct a penalty function of the form

$$f_{\text{coll}} = \left(\frac{d_{\text{min}}^2 - d(\mathbf{e}_i, \mathbf{e}_j)^2}{d_{\text{min}}^2} \right)^3, \tag{5.7}$$

that, by construction, has continuous second derivatives at $d = d_{\text{min}}$. While we could have used an exponential barrier for the penalty function, we found the polynomial version to be sufficient in practice, preventing intersections reliably. For the link-link case, we test all pair-wise combinations of spheres and cylinders and, whenever too close proximity is detected, activate a corresponding penalty term. Finally, in order to prevent intersections between links and flexures, we test each edge of the centerline against the five shapes of the link and issue a penalty term if distances are too small. It should be pointed out that we do not check for flexure-link collisions if the flexure is attached to the link. We found that doing so gives the mechanisms more freedom to adapt at the cost of sometimes introducing collisions; see Figure 5.4. These can, however, always be resolved by adapting the geometry of the link in a post-processing step (right). Finally, each of the three cases is implemented using the same closed-form function for measuring the distance between two capsules [Zehnder et al., 2016].

5.3.5 Preventing Material Failure

The flexures in a compliant mechanism are only elastic within a certain range of deformation, beyond which material fatigue and, ultimately, failure will occur. One central goal of our approach is to minimize the risk of such failures. The field of fracture mechanics has many ways of modeling various failure modes for different types of materials. One widely used model is the so called von Mises criterion [Hill, 1998], which states that the onset of failure—or yielding—is a function of the internal stress acting inside the material. Indeed, many manufacturers provide data for their printable materials that can readily be used in this criterion. But unfortunately, the volumetric stresses required to evaluate the yield criterion are not directly available from the discrete rod model. Our goal is therefore to transform the discrete centerline stretch, bending, and twist into a single volumetric strain tensor, from which the stress is then obtained by virtue of a continuum-mechanics material law.

We base below derivations on the elasticity theory on rods as described in Landau et al. [1986]. For bending, we extend their formulation to curved rest configurations and account for the coupling between stretch and bending away from the centerline. For twist, we largely follow their description, outlining their derivation for the reader's convenience and discussing the interface with the discrete elastic rod model [Bergou et al., 2008].

Volumetric Strain from Bending In the Kirchhoff-Love model of thin elastic rods, the different deformation modes are decoupled. In particular, bending does not induce centerline stretch and vice versa. When considering the volumetric picture, however, it is evident that bending will induce compression on one side of the centerline and stretching on the opposite side, even though the centerline itself remains unstretched (see Figure 5.5). In order to quantify this deformation, we start with a simple example of an initially straight rod bent into a state of constant curvature $\kappa = 1/R$. Let dz denote a length element along the centerline and let dz' be the length of a corresponding segment at a given location x in the rod, where $-a \leq x \leq a$. Furthermore, let $d\bar{z} = d\bar{z}'$ denote the corresponding lengths in the (straight) undeformed configuration. Since the centerline does not stretch under bending, we have $dz = d\bar{z}$. Due to the constant curvature, we have $\frac{dz'}{(R-x)} = \frac{dz}{R}$ and the deformation follows as

$$\varepsilon_z = \frac{dz' - d\bar{z}'}{d\bar{z}'} = -\frac{x}{R} = -\kappa x \,. \tag{5.8}$$

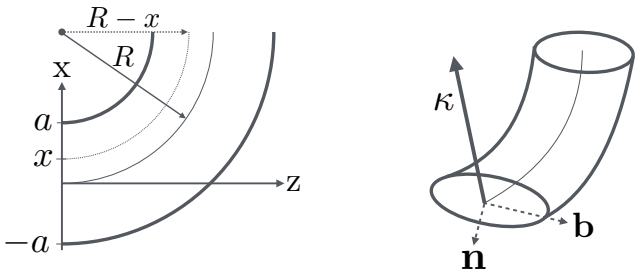


Figure 5.5: *Bending induces stretching and compression away from the centerline.*

For curved rest shapes, $dz = d\bar{z}'$ still holds, but additionally we have $\frac{d\bar{z}'}{(\bar{R}-x)} = \frac{d\bar{z}}{\bar{R}}$, where $\bar{R}$ is the radius of curvature for the undeformed state. This leads to

$$\varepsilon_z = -\frac{(\kappa - \bar{\kappa})x}{1 - \bar{\kappa}x} \text{ with } \bar{\kappa} = \frac{1}{\bar{R}} \,. \tag{5.9}$$

When accounting for simultaneous bending and stretching deformation of the centerline $\varepsilon_{\text{cl}} = \frac{dz}{d\bar{z}} - 1$, we arrive at

$$\varepsilon_z = \frac{\varepsilon_{\text{cl}}(1 - \kappa x) - (\kappa - \bar{\kappa})x}{1 - \bar{\kappa}x} \,. \tag{5.10}$$

Finally, in order to extend the above formulation to the full three-dimensional picture, we let $\mathbf{c}$ denote an arbitrary point within the elliptical cross section of the rod and write

$$\varepsilon_z(\mathbf{c}) = \frac{\varepsilon_{\text{cl}}(1 - \boldsymbol{\kappa} \cdot \mathbf{c}) - (\boldsymbol{\kappa} - \bar{\boldsymbol{\kappa}}) \cdot \mathbf{c}}{1 - \bar{\boldsymbol{\kappa}} \cdot \mathbf{c}} \tag{5.11}$$

where $\boldsymbol{\kappa}$ and $\bar{\boldsymbol{\kappa}}$ are 2D vectors holding curvature values relative to the two material directions of the rod. Under the assumption of no shearing [Landau et al., 1986], the bending strain tensor at a given point on the boundary of the cross section is obtained as

$$\boldsymbol{\epsilon}_b(\mathbf{c}) = \varepsilon_z(\mathbf{c}) \text{diag}\,(-\nu, -\nu, 1) \,, \tag{5.12}$$

where ν is Poisson's ratio of the material. It is worth noting that $\boldsymbol{\epsilon}_b$ is a 3×3 tensor expressed relative to the material frame $\mathbf{T} = [\mathbf{t}_1, \mathbf{n}_2, \mathbf{b}_3]$ of the rod, where $\mathbf{n}_1$ and $\mathbf{b}_2$ span the cross-sectional plane and $\mathbf{t}_3$ coincides with its tangent.

To evaluate the strain at vertex i of the discrete elastic rod model (compare with [Bergou et al., 2008]), we first compute the *curvature binormal*, then express it w.r.t. the material frame at i by averaging

$$\mathbf{k}_i = \frac{2\mathbf{t}^{i-1} \times \mathbf{t}^i}{1 + \mathbf{t}^{i-1} \cdot \mathbf{t}^i}, \quad \boldsymbol{\kappa}_i = \frac{1}{2\bar{l}_i} \begin{bmatrix} \mathbf{k}_i \cdot \mathbf{b}^{i-1} + \mathbf{k}_i \cdot \mathbf{b}^i \\ -(\mathbf{k}_i \cdot \mathbf{n}^{i-1} + \mathbf{k}_i \cdot \mathbf{n}^i) \end{bmatrix}$$

with $\left[\mathbf{t}^{i-1}, \mathbf{b}^{i-1}, \mathbf{b}^{i-1}\right]$ and $\left[\mathbf{t}^{i}, \mathbf{b}^{i}, \mathbf{b}^{i}\right]$ denoting the material frames of the two adjacent edges. Note that $\boldsymbol{\kappa}_i$ as defined in Bergou et al. [2008] is an integrated quantity. Hence, we divide by the integration domain $\bar{l}_i$ which equals the sum of the half lengths of the two adjacent, undeformed edges.

Volumetric Strain from Twisting Analogously to bending, we introduce relevant quantities for twist with an initially straight rod, rotated about the z-axis as depicted in Figure 5.6. Focusing our discussion on a cross section parallel to the x-y plane at a small distance z away from the origin, we seek to quantify the displacement a point on the cross section undergoes. To gauge the amount of rotation at z relative to the origin, we multiply z with the the torsion angle τ which measures the angle of rotation per unit length of the rod. We then rotate the cross-sectional point $[x, y, z]^T$ about the z-axis, leading to the in-plane displacement $\mathbf{u}$

$$\begin{bmatrix} -\tau z y \\ \tau z x \\ 0 \end{bmatrix} = \tau z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} . \tag{5.13}$$

Cross sections, however, do not stay planar under torsion. To account for out of plane displacement, we follow Landau et al. [1986] and introduce a *torsion function* $\psi(x, y)$ which is cross section dependent

$$\begin{bmatrix} -\tau z y \\ \tau z x \\ \tau \psi(x, y) \end{bmatrix} \tag{5.14}$$

with the displacement in the z-direction proportional to the torsion angle. This is a sensible choice because for $\tau = 0$ the displacement field is zero.

While significant at global scales, the relative displacement of neighboring points is small under torsion and the linearized Cauchy strain

$$\boldsymbol{\epsilon}_t(\mathbf{c}) = \frac{1}{2} \begin{bmatrix} 0 & 0 & \tau\left(\frac{\partial \psi}{\partial x} - y\right) \\ 0 & 0 & \tau\left(\frac{\partial \psi}{\partial y} + x\right) \\ \tau\left(\frac{\partial \psi}{\partial x} - y\right) & \tau\left(\frac{\partial \psi}{\partial y} + x\right) & 0 \end{bmatrix}$$

is sufficiently precise. Note that torsion does not lead to any volume changes (diagonal entries are all zero) and is a pure shear deformation, orthogonal to bending and stretching (only diagonal entries non-zero).

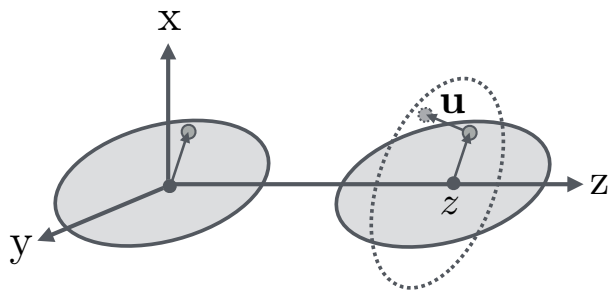


Figure 5.6: *Twist induces pure shear deformation about the rotation axis.*

For a particular cross section, the torsion function $\psi(x, y)$ is the solution to a Poisson's equation with Neumann boundary conditions that emerges when assuming a linear elastic material

$$\boldsymbol{\sigma}(\mathbf{c}) = \lambda \text{tr}\left(\boldsymbol{\epsilon}(\mathbf{c})\right)\mathbf{I} + 2\mu\boldsymbol{\epsilon}(\mathbf{c}) \tag{5.15}$$

and asking for static equilibrium $\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}$ (see Landau et al. [1986] for a detailed derivation). For an elliptical cross section, the analytical solution is

$$\psi(x, y) = \frac{b^2 - a^2}{a^2 + b^2} xy \; .$$

The torsion angle τ_i at vertex i of the discrete rod model is $\frac{\alpha_i}{l_i}$ where α_i is the integrated twist angle at i (see Bergou et al. [2008]).

Failure Criterion The contributions from centerline stretching, and volumetric bending and twisting are combined into the Cauchy strain $\boldsymbol{\epsilon}(\mathbf{c}) = \boldsymbol{\epsilon}_b(\mathbf{c}) + \boldsymbol{\epsilon}_t(\mathbf{c})$, which we plug into the linear elastic material (Equation 5.15) to compute the corresponding Cauchy stress $\boldsymbol{\sigma}(\mathbf{c})$. With the volumetric stress available, we are ready to define a penalty function to prevent material failure. To this end, we ask that the maximum von Mises stress remains below a given threshold value, dictated by the printing material.

Axial strains due to bending and twisting assume their maximum values on the surface of the rod, and so does the corresponding stress. In lack of an analytical expression, we evaluate the stress at a set of n sample points $\mathbf{c}_i = \left\{ \left[a_i \cos(\phi_j) \;\; b_i \sin(\phi_j) \right]^T \right\}$ distributed along the boundary of the cross section, then compute the von Mises stress for sample $\mathbf{c}_j$ as

$$\sigma_v(\mathbf{c}_j) = \sqrt{\frac{3}{2} \sum_{k,l} s_{kl}^2} \text{ and } \mathbf{s}(\mathbf{c}_j) = \boldsymbol{\sigma}(\mathbf{c}_j) - \frac{1}{3}\text{tr}(\boldsymbol{\sigma}(\mathbf{c}_j))\mathbf{I} \; .$$

Finally, the penalty term is defined using the soft maximum over all samples $\mathbf{c}_s^t$ on all rods along the trajectory t,

$$f_{\text{fail}} = \frac{\sum_{t,s} \sigma_v(\mathbf{c}_s^t) e^{\beta_{\text{fail}} \sigma_v(\mathbf{c}_s^t)}}{\sum_{t,s} e^{\beta_{\text{fail}} \sigma_v(\mathbf{c}_s^t)}}, \tag{5.16}$$

with large positive β_{fail}.

5.3.6 Optimization

With the design objectives defined, we seek to compute optimal values for all parameters. Since most of these objectives have quite complex derivatives, we first experimented with CMA-ES [Hansen et al., 2003], a widely-used stochastic optimization scheme that does not required derivative information. However, we found the convergence and performance of this algorithm unsatisfying and therefore switched to a more powerful approach based on the implicit function theorem (see Section 3.3.1).

To simplify notation, we summarize the system state as $\mathbf{y} = (\mathbf{x}, \mathbf{s})$ and condense all objectives in a single function $f(\mathbf{y})$. In order to compute optimal parameter values, we ask that the gradient of the objective function f vanishes,

$$\frac{\partial f(\mathbf{p}, \mathbf{y}(\mathbf{p}))}{\partial \mathbf{p}} = \mathbf{0}\,, \tag{5.17}$$

which requires the derivative of state with respect to parameters. To this end, we observe that all admissible states have to be equilibrium configurations, i.e.,

$$\mathbf{g}(\mathbf{p}, \mathbf{y}) = \frac{\partial E(\mathbf{p}, \mathbf{y})}{\partial \mathbf{y}} = \mathbf{0}\,, \tag{5.18}$$

establishing a map between state and parameters, $\mathbf{y} = \mathbf{y}(\mathbf{p})$. For any admissible changes in parameters, we therefore require that

$$\frac{d\mathbf{g}}{d\mathbf{p}} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{p}} = \mathbf{0}\,, \tag{5.19}$$

from which we can compute the derivative of state with respect to parameters and, consequently, the gradient of the objective function. We compute the remaining derivatives using a mix of auto-differentiation and manually-derived expressions. For minimization, we use a standard quasi-Newton scheme with L-BFGS (see Section 3.3.2.

5.4 Results

We have used our technique to estimate and fabricate compliant versions for a total of five planar and spatial mechanisms (see Figures 5.7-5.10). We minimize a weighted sum of the afore introduced objectives with default values $w_{\text{track}} = 10^5$, $w_{\text{stab}} = 5 \cdot 10^5$, $w_{\text{act}} = 10$, $w_{\text{coll}} = 10^7$, $w_{\text{fail}} = 10^{-6}$ for all our results.

Fabrication and Hardware For fabrication, we rely on a Stratasys Connex 350 and use their strong and flexible Rigur material. Both rigid links and flexures are printed with the same material and as single assembled pieces. We use Dynamixel's XL-320 and their OpenCM9.04 C-Type board to drive our mechanisms, adding a Robotis BT-210 Bluetooth communication controller for remote control of our RC Car.

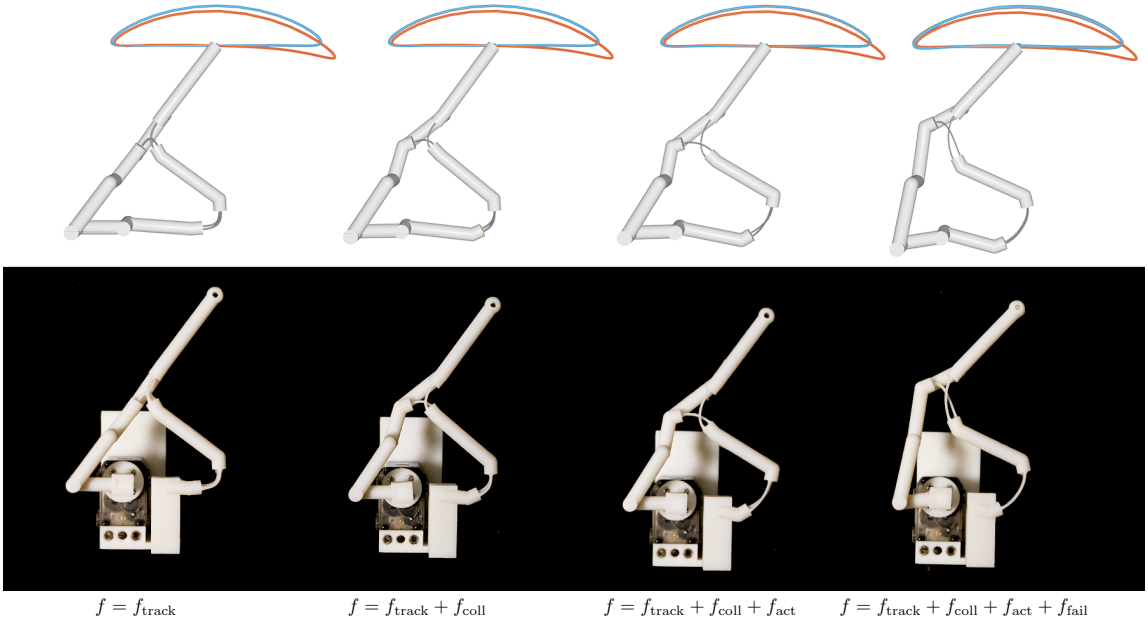
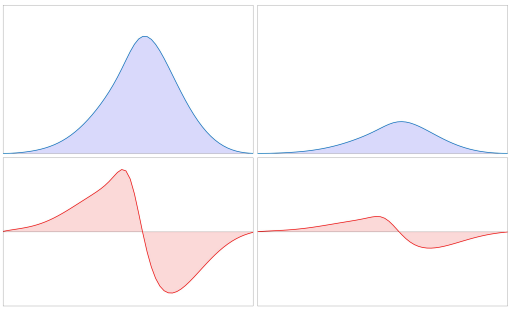


Figure 5.7: ***Chebyshev Linkage.*** *Simulated (top row) and fabricated (bottom row) compliant Chebyshev linkages that were optimized with different objective terms as indicated. Starting from the red trajectory after initialization, the trajectory (in blue) of a marker on the original assembly is recovered for all combinations of objectives. However, only with f_{fail} active (right most assembly), the compliant mechanism lasts.*

Chebyshev Linkage To analyze the impact of the individual design objectives, we replaced two out of three hinge joints of Chebyshev's Lambda Mechanism with our two-stacked-flexure design while leaving the third one

unchanged. Chebyshev's Lambda Mechanism is well-known for its characteristic trajectory with a long, approximately straight segment. As we illustrate in Figure 5.7 (top row), we can recover this characteristic trajectory (in blue) from the initially off one (in red) where the average error measured in simulation is lower than a tenth of a millimeter: if only motion tracking of the respective point is active (1st from left), the mechanisms is not fully functional due to a link-link collision. We can prevent this collision while still recovering the trajectory to the same degree (2nd from left) by setting



f to $f_{\text{track}} + f_{\text{coll}}$. By further activating f_{act} (3rd from left), we can significantly reduce the required motor torque as we illustrate with before (left column) and after (right column) energy (in blue) and torque profiles (in red) of the cycling motion in the inset on the left. However, while collision-free, the bending and twisting stresses are too high for the targeted printer material. With f_{fail} active (4th from left), we can reduce these stresses by a factor greater than an order of magnitude, resulting in a structurally-sound and functional mechanism. For validation, we fabricated physical prototypes corresponding to designs that were optimized with and without f_{fail}. Using only tracking and collision objectives, i.e., $f = f_{\text{track}} + f_{\text{coll}}$, the printed mechanism failed after one cycle through fracture in one of the flexures. The prototype for which f_{fail} was included in the optimization was able to run for more than an hour (this accounts for more than 2000 motion cycles) without noticing any sign of material failure nor onset of yielding.

Including additional objectives rather than simply using f_{track} leads to a decrease in tracking accuracy, but as can be seen from Table 5.2 and Figure 5.7, the functionality of the mechanism is still maintained. Evaluating the various objectives and their derivatives with respect to the design parameters is not expensive in comparison to the time spent on simulation: given parameter values for the flexures, we compute the equilibrium configuration of the mechanism for all 60 steps along the motion cycle, each of which means solving a nonlinear system of equations. Simulation is required at least once per quasi-Newton iteration for solving (5.17), and potentially more often during line search. Table 5.1 shows a comparison between the time spent on simulation against the time spent on objective evaluation, once using $f = f_{\text{track}} + f_{\text{coll}}$ and once with the addition of our most expensive objective f_{fail}.

Jansen's Linkage The fundamental building block of Theo Jansen's Strandbeests is a leg mechanism with a total of 9 hinges and 9 rigid links, driven by a single motor. We replaced 8 hinges with layered, compliant designs, two among which are three-way couplings. As can be see in Figure 5.8, the motion trajectory of the unoptimized mechanism (in red) is far off the original trajectory (in black). With our performance objectives, however, we can recover the function of the mechanism (fraction of the trajec-

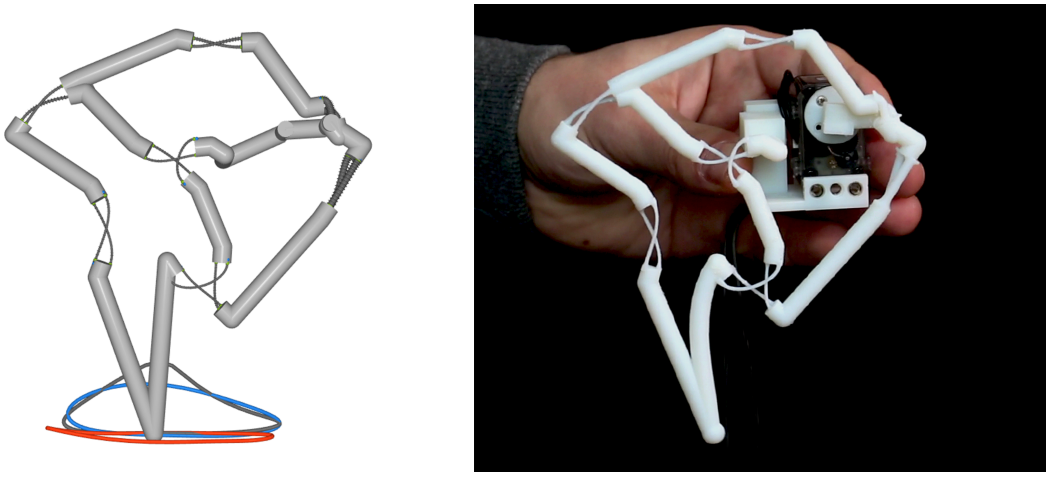


Figure 5.8: ***Jansen's Linkage.*** *After replacing conventional hinges with compliant flexures, the end effector trajectory (red) deviates significantly from the original trajectory (in black). After optimally placing, orienting, and sizing the flexures, we can recover the lower portion of the trajectory (in contact with the ground) to a large degree (in blue).*

tory in contact with the ground, in blue) while keeping the stresses of the leg mechanism within reasonable bounds.

As previously mentioned, we first experimented with CMA-ES due to the complex derivatives of our objectives. With Jansen's leg being one of the most complex examples when it comes to flexure count, it is well-suited for a comparison of a derivative-free optimization (CMA-ES) to our method of choice – a quasi-Newton with L-BFGS: after 50 iterations of quasi-Newton with only f_{track} active, the objective value is two orders of magnitude smaller than after the same amount of CMA-ES iterations. We further observe that quasi-Newton moves a subset of parameters such as, e.g., the flexures' cross

	$f_{\text{track}} + f_{\text{coll}}$	$f_{\text{track}} + f_{\text{coll}} + f_{\text{fail}}$
Full iteration	4.82s	5.35s
Simulation	3.63s	3.81s
Objective evaluation	0.037s	0.068s

Table 5.1: *Computation times for optimizing the Chebyshev linkage using different objective terms as indicated in the top row. Simulation is the dominant part in both cases, whereas the evaluation of the objective terms is negligible in comparison.*

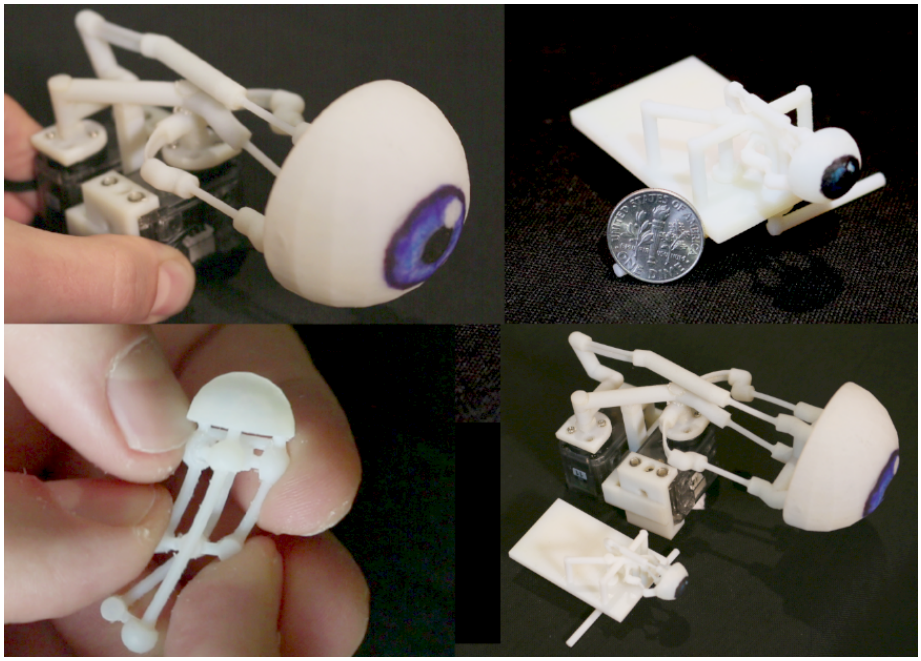


Figure 5.9: ***Eye Mechanism.*** *Compliant eye mechanism at human-scale (top left), at Dime-scale (bottom left and top right), and a side-by-side comparison (bottom right).*

section parameters significantly more than CMA-ES, suggesting that CMA-ES is ill-suited for the task at hand even if we ignore time complexity.

One quasi-Newton step for the Jansen leg (highest number of flexures) takes on average 39s on a machine with an Intel Core i7-6700 3.5GHz processor with a total of 32 GB of RAM. For improved efficiency, we parallelized evaluations along the trajectory and also along the line search direction. In comparison, one CMA-ES iteration takes on average 31s with objective evaluations parallelized.

Eye Mechanism The design of well-functioning animatronic eye mechanisms at small scales is a formidable task. For our Eye Mechanism (Figure 5.9), we estimated a fully compliant version from a spatial input with 2 hinges and 4 ball-and-sockets, jointing together a total of 8 rigid links, and driven by two rotational motors. The user starts by specifying motor profiles that lead to eye motion that sufficiently spans the desired range of motion. After fabricating our animatronic eye, we noticed that the compliant version preserves this range closely, hence, is function-preserving. The eye ball of the input roughly matches the size of a human eye. However, many animals

have eyes of far smaller scales: if, e.g., a lizard is replicated as an animatronic, one needs mechanisms that run reliably at very small scales. As we illustrate with a miniaturized eye mechanism (compare with Figure 5.9), the use of compliance enables fabrication at scales far beyond what is possible when relying on conventional mechanical assemblies. This is due to the scale invariant minimal tolerance between movable parts (approximately $0.25mm$ for Stratasys' Connex series): we had to significantly reduce the range of motion of the ball-and-sockets to prevent balls from popping out of their socket. Hence, the mechanical eye mechanism at Dime-scale is not function-preserving while our compliant version is.

RC Car For a steering mechanism, function-preservation is pivotal. For our RC CAR (see also Figure 5.10), we estimated a compliant version of a conventional steering mechanism consisting of 5 rigid bodies jointed together with 2 hinges and 2 ball-and-sockets, and driven by a single motor. By testing our car, i.e. driving it on a flat surface, we notice that the functionality of the original mechanism is preserved, even under self-weight of the car and frictional contact at the three wheel shafts. It is worth noting that the compliant ball-and-socket flexures are pulling or pushing dependent on the steering direction and do not buckle under compression. Due to symmetry, we optimized only half of the mechanism, then mirrored the resulting monolithic structure prior to fabrication.

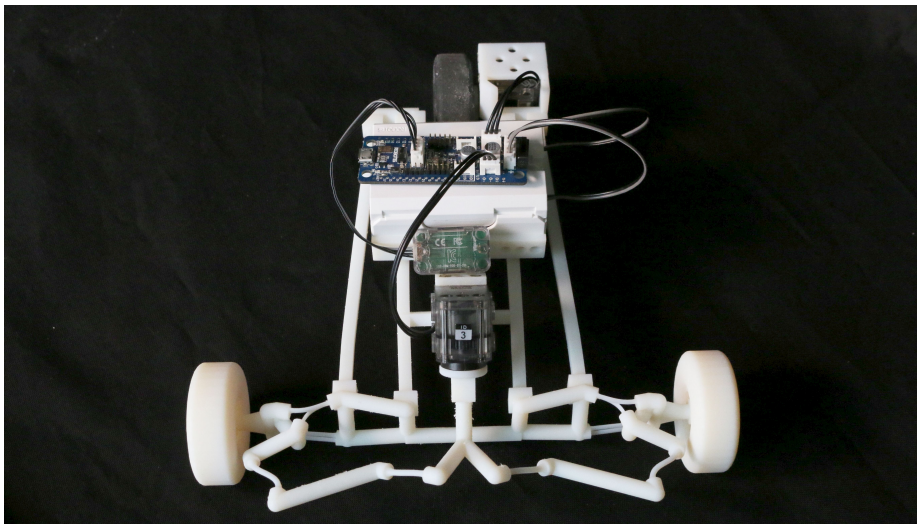


Figure 5.10: ***RC Car.*** *Remote controlled car featuring a fully compliant steering mechanism. In addition to preserving the steering functionality of the input mechanism, this design was optimized to sustain its own weight.*

Compliant Hand For our compliant hand design, we replaced all 9 hinges of a conventional finger mechanism (9 hinges, 1 linear actuator, 9 rigid links) with flexures, then optimized the resulting monolithic mechanism for fabrication with Rigur. We then attached 5 identical fingers to a laser-cut piece of Plexiglas and actuated them with strings from a distance (see Figure 5.11): by pulling on the strings we store potential energy in the fingers. This energy is released when reducing the actuation forces on the strings, causing the fingers to move back to the rest configuration where the elastic energy is zero.

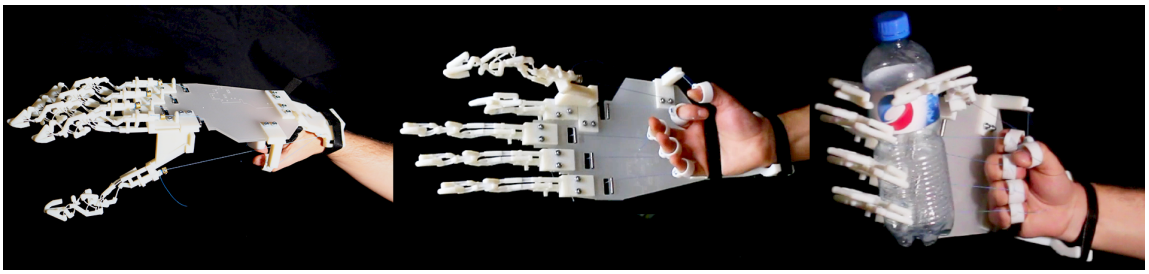
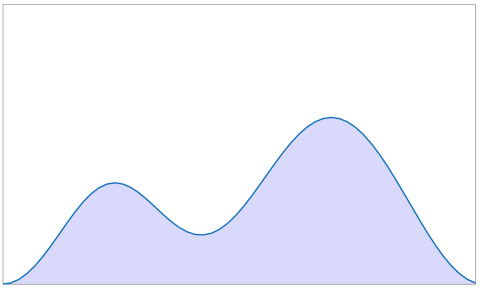


Figure 5.11: ***Compliant Hand.*** *A compliant finger mechanism is replicated and assembled to create a fully operational hand. Side and bottom views are shown on the left and in the middle. On the right we show the hand performing a teleoperated grasping task using cables for actuation. Thanks to restoring forces from the compliant flexures, the hand can be actuated using a single cable per finger.*

Dragon For our dragon, we estimated a compliant version of a spatial wing mechanism (1 motor, 3 hinges, 1 ball-and-socket, 7 rigid links). We replaced all joints but the motor with compliant flexures. While seemingly simple, we observe that the hinge flexures undergo twist deformations, underlining the importance of minimizing stresses due to torsion. These twist deformations are due to proximity of two compliant hinges with flexures aligned orthogonally to one another. If only failure prevention for stretching and bending is active, twist stresses are too high for the targeted printer material, and the risk of material failure high. We observe that the degrees of freedom of the conventional hinges are not well preserved. Nonetheless, the trajectory of the user-specified marker point is preserved to a high degree, resulting in a function-preserving and failure-resistant compliant wing mechanism with intricate 3D motion. Interestingly, this mechanism is bistable as we can see when looking at the energy profile. Besides the rest configu-



ration, there is a stable equilibrium at an intermediate step t along the cyclic motion.

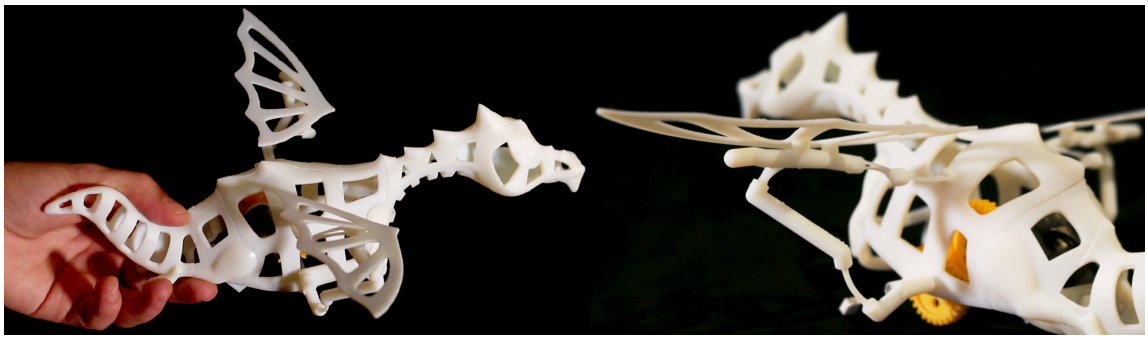


Figure 5.12: ***Dragon.*** *Full dragon (left) and close-up onto the wing mechanism (right). The wings of the dragon are two identical but mirrored versions of a spatial compliant mechanism. This example exhibits large deformations induced by twist, emphasizing the need for a twist-aware objective for preventing material failure.*

Summary We used our method to design and fabricate a set of planar and spatial compliant mechanisms showing different types of joints, varying complexity, and diverse functions. Though different, all these examples share the need for explicitly preventing material failure during optimization: all of our tests indicate that when only motion tracking is taken into account, the fabricated mechanism will invariably fail during operation. In contrast, by incorporating a failure-preventing objective in the optimization, we obtain compliant designs with largely improved robustness at the expense of somewhat reduced tracking accuracy. Table 5.2 reports comprehensive performance data for all examples that were fabricated.

5.5 Conclusions

We presented a computational tool for designing compliant mechanisms and demonstrated its use on several physical prototypes. While the types of joints supported by our method cover a large range of useful planar and spatial mechanisms, conventional joints performing full revolutions cannot be converted to compliant flexures. However, rather than the result of a specific design tool, this limitation is inherent to compliant mechanisms in general. Fortunately, the number of full-revolution joints is typically small.

In the future, we plan to extend our method to support topology changes. For conventional assemblies, a pivotal requirement is that the degrees of

Model	# DoF	# It.	Avg. It. Cost (s)	Mean Error (mm)	Max Error (mm)
Chebyshev	35	152	5	0.59	1.04
Jansen	112	880	39	1.67	8.16
Eye	110	283	34	1.65	5.04
Car	63	76	28	0.86	2.35
Hand	141	183	31	0.63	5.28
Dragon	70	164	11	1.15	3.42

Table 5.2: ***Statistics.*** *The columns (from left to right) list the number of degrees of freedom, the number of iterations required for convergence, the average iteration cost, as well as the mean and maximum error for the motion tracking objective.*

freedom of the joints are equal to the number of unknown state variables at all times, rendering the automated exploration of topological changes an utterly complex task. The use of compliance, however, paves the way for a more graceful exploration of this space thanks to the more even distribution of the infinite stiffness and compliance concentrations of conventional joints. The removal of individual links could make room for overall performance improvements because link-link or link-flexure collisions that prevented further stepping in respective descent directions may no longer be present.

As another limitation, we have so far focused on structurally-sound and function-preserving kinematic behavior. Dynamic effects, however, can play a role as can be observed when teleoperating with our compliant hand. These effects are, however, highly dependent on the choice of material: we sintered an individual finger of our compliant hand and observed a more high-frequent but far less pronounced dynamic behavior. An interesting direction for future work would be to extend our method to model and optimize for these dynamic effects.

If a legged mechanism undergoes a periodic motion, there are at least two stationary points at which the structure is in equilibrium—though one of them (the maximum) is unstable. While we have observed bi-stable behavior in some of our mechanisms (see, e.g., Dragon), designing for multi-stability [Pucheta and Cardona, 2010] could be an interesting avenue for further exploration.

CHAPTER 6

Designing Cable-Driven Actuation Networks for Kinematic Chains and Trees

In this chapter we focus on artist-controlled linkages that are actuated using cables routed through point-to-point connections spanning one or more joints in a linkage. The defining characteristic of this actuation paradigm is that only unidirectional actuation is possible – i.e. cables cannot *push* a link. Cable-driven designs have important design advantages over purely linkage- or gear-based approaches such as allowing significant control over the location of motor mass. For instance, a heavy motor can be located in the torso of a mechanical character while cables are used to actuate the limbs. This enables lightweight limbs that can therefore undergo more expressive motions. Furthermore, cables are easier to route than linkages meaning that they can more easily actuate several joints at once. Finally, because cables can span and couple multiple joints, cable-driven animatronic mechanisms may be able to better replicate the coupled motions inherent in many creatures.

Our system takes as input a hierarchical assembly consisting of rigid links jointed together with hinges. The user also specifies a set of target poses or keyframes using inverse kinematics. We depart from the approach followed in the previous chapter, and instead of employing fexures as our compliant joints, we experiment with different flexible articulations: torsional springs. We place a torsional spring at each joint location and we compute a cable network that allows us to reproduce the specified target poses. We start with a large set of cables that have randomly chosen routing points and

we gradually remove the redundancy. Then we refine the routing points taking into account the path between poses or keyframes in order to further reduce the number of cables and minimize required control forces. We propose a reduced coordinate formulation that links control forces to joint angles and routing points, enabling the co-optimization of a cable network together with the required actuation forces. We demonstrate the efficacy of our technique by designing and fabricating a cable-driven, animated character, an animatronic hand, and a specialized gripper.

6.1 Introduction

Graphics research has 30 years of expertise in developing tools which allow digital artists to create expressive animations by posing a hierarchical set of rigid links. They breathe life into these articulated assemblies by making them move or locomote like a human, a familiar character, an animal or a fantasy creature. Tools like these enable artists to bring animated characters in feature film to life, giving them a unique personality.

With the advent of consumer-level digital fabrication technologies and powerful yet affordable off-the-shelf electronic components, artists now have the machinery at their disposal to make these articulated, animated assemblies physical. Creating such devices involves the design of a kinematic structure, determining the possible range of motion along with an actuation mechanism in order to animate the structure. Applications include animatronics, personalized robotics, and marionette design.

As the digital animations do not have to obey the laws of physics, such kinematic assemblies can often be slender and their motion fast. This makes it infeasible to place a motor at each joint due to both their size and weight. An alternative to distributed actuation is to use cables, placing actuators in a centralized location away from the mechanical assembly, enabling lightweight designs.

This chapter describes a method that designs a cable network, aiding the artist and hobbyist with the design of cable-driven kinematic chains and trees (see Figure 6.1) that closely match a set of specified poses or keyframes when actuated. A theoretical upper bound on the number of required cables is two per rotational degree of freedom because we can only pull on cables. To reduce the cost and complexity of the kinematic assemblies, we wish to minimize the number of cables far beyond this upper bound. This presents a challenging design problem as it is of discrete nature and cables introduce non-trivial couplings and non-linearities. We place torsional springs at

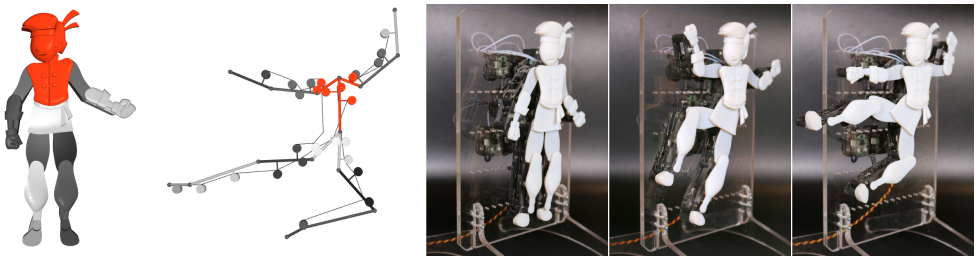


Figure 6.1: *Our computational tool for designing cable-driven kinematic chains and trees (left) enables artists and hobbyists to size and place a cable network (middle) in order to closely match a set of target poses or keyframes using co-optimized control forces (right).*

the joints to permit unidirectional actuation as cables can only exert pulling forces. We then co-optimize the number and placement of cables together with the control forces needed to drive the mechanical hierarchies.

We tackle the automated design with a two-step approach where we first identify the topology of a network by removing unactuated cables from a large set with routing points chosen at random. In a second step, we refine this network by parameterizing the routing points, taking the path between poses or keyframes into account, and further reducing the network and control forces if possible. To enable co-optimization of cable routing points and actuation forces, we introduce torque equilibrium equations that directly relate joint angles and routing points to the control forces.

The robotics community has proposed a myriad of cable-driven hands [Catalano et al., 2012; Ma et al., 2013; Grebenstein, 2014] or full-bodied robots [Rooks, 2006; Hannaford et al., 2013; Spröwitz et al., 2014]. However, designing these kinematic assemblies manually, roboticists focus on the optimal exchange of forces during physical interactions with humans or the environment. Our work is complementary to these techniques and targets an *automated* and *optimal* routing of a complex cable network under a no-contact assumption.

Our implementation is quasi-static and our physical assemblies are designed to meet this assumption. We validate our results by fabricating three of our optimized designs. With our examples, we illustrate applications in functional as well as artistic design.

6.2 Observations

Before we formalize our cable-driven simulation and optimization approach on general hierarchical input, we discuss a series of observations on a three-

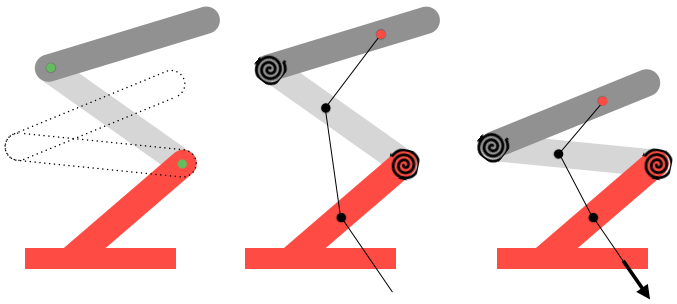


Figure 6.2: *The input to our system is a kinematic assembly consisting of rigid links, jointed together at their ends (left). The red link is fixed. A user first specifies target poses (dotted link contours) and adds torsional springs to the hinges (green circles). A set of cables is then optimized (middle) to hit the user-specified target poses as closely as possible when actuated (right). The cables are attached at one end (red circle) and are routed with pulleys (black circles) to the fixed link. Elastic springs are added to the hinges (black spirals) to define the rest configuration as the zero energy state.*

link kinematic chain (see Figure 6.2) that guide and motivate our representation and formulation.

As input we assume a *kinematic assembly* consisting of a single *kinematic chain* without loops, or a hierarchy of such chains to which we refer as *kinematic trees*. These assemblies consist of rigid links, connected to one another with mechanical hinges as illustrated in Figure 6.2 left.

Given a kinematic assembly and one or several target poses (see dotted contour lines in Figure 6.2 left), our goal is to determine a *cable network* (middle), i.e., a number of cables and corresponding routing points on the individual links of the assembly such that, when applying specific forces to the cables, the assembly approximates the target poses as closely as possible (right).

Valid alternatives to a cable network are motors at the hinges or the addition of mechanical couplings between links [Thomaszewski et al., 2014]. A kinematic tree such as, e.g., a mechanical hand design, however, becomes bulky and heavy if a motor is added to each individual joint and inertial forces become prohibitively high. While leading to more lightweight designs, mechanical couplings between links are of fixed length and many of them are needed to achieve simple motions such as the contraction of a chain (compare Figure 6.2 middle and right).

For a fully actuated kinematic tree, we would need twice the number of actuators (one to pull on either side of each hinge). However, although target poses generally involve non-zero deformations for all joints, we can exploit the inherent low-dimensionality of the problem. To this end, we first add

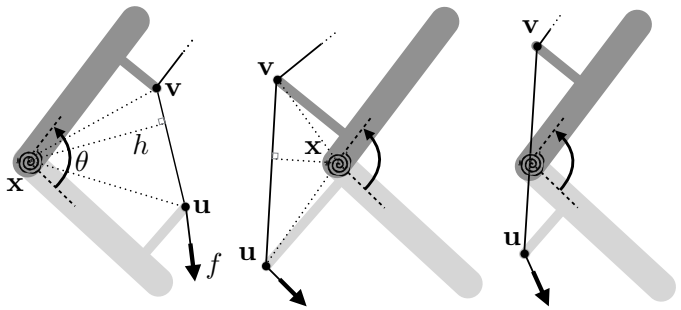


Figure 6.3: *The assembly is in equilibrium if the torque $\tau_s(\theta)$ of the torsional spring with joint angle θ equals the applied torque hf with signed moment arm $h = \frac{\det[\mathbf{u}-\mathbf{x},\mathbf{v}-\mathbf{x}]}{\|\mathbf{u}-\mathbf{v}\|}$.*

elastic springs to the joints (see Figure 6.2 middle). We then strategically place routing points of a cable network that leads to a complex coupling between individual joints and thus significantly reduces the number of cables (actuators) without compromising shape approximation. Note that the torsional springs uniquely define the rest configuration as the state of zero elastic energy.

Following the standard formulation described by Coros and colleagues [2013], we first experimented with representing the state of each rigid link with a position and orientation, and hinges and cables interconnecting them with non-linear constraints. However, the non-negativity of actuation forces (we can only pull on cables) require additional inequality constraints on the cable lengths that are non-trivial (refer to Figure 6.6 for a small example): if we pull on one cable with a non-zero force other cables may extend in length while they remain unactuated.

Departing from this full coordinate formulation, we introduce torque equilibrium equations that directly relate joint angles and routing points to the control forces, avoiding any non-trivial inequality constraints.

6.3 Simulating Cable-Driven Trees

We base our formulation on the following observation: if we pull on the cable in Figure 6.3 right with a force $f > 0$, we apply a torque hf about $\mathbf{x}$ where h is the moment arm of the oriented triangle $(\mathbf{x}, \mathbf{u}, \mathbf{v})$. The system is in equilibrium if this torque equals the one from the torsional spring $\tau_s(\theta)$ that evaluates to a positive value if the oriented joint angle θ is smaller than the angle at rest. This observation holds if the cable is attached to respective links with friction- and dimensionless pulleys (black circles). We defer a discussion of finite-dimensional pulleys to Sec. 6.5.

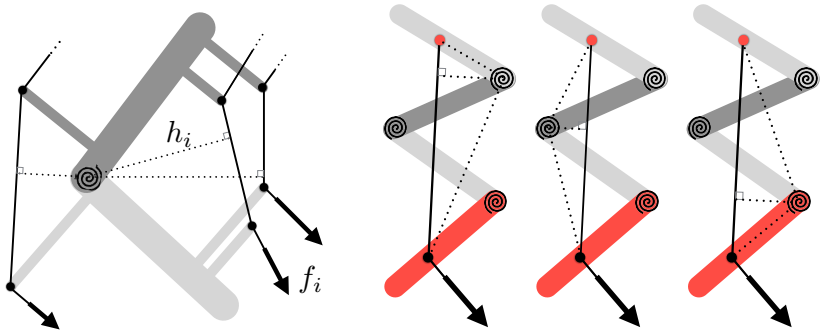


Figure 6.4: *Torques $h_i f_i$ from several cables can affect the position of a single joint (left) and a single cable can affect the positions of several joints (right).*

To relate the moment arm to the joint location $\mathbf{x}$ and the two routing points $\mathbf{u}$ and $\mathbf{v}$, we express the signed triangle area with Cramer's rule $\frac{1}{2}\det[\mathbf{u}-\mathbf{x}, \mathbf{v}-\mathbf{x}]$ and set it equal to half the triangle's altitude h times its base $\|\mathbf{u}-\mathbf{v}\|$.

This equilibrium condition still holds if the triangle orientation is flipped as illustrated in Figure 6.3 middle: if we pull on the cable, the oriented angle θ becomes larger than the one at rest. Hence, the torsional spring torque becomes negative. Due to the change of orientation of the triangle $(\mathbf{x}, \mathbf{u}, \mathbf{v})$, $\det[\mathbf{u}-\mathbf{x}, \mathbf{v}-\mathbf{x}]$ is negative and compensates this sign change. A special case is depicted in Figure 6.3 right: if the triangle area becomes zero ($h = 0$), the applied torque is zero independent of the magnitude of the applied force. Hence, the elastic spring remains in its zero energy state for all $f > 0$. However, such configurations are unstable equilibria and will therefore not be present in real systems.

The above observation is pivotal when we optimize routing points: if we move the routing points from one side to the other (flip of triangle orientation), the moment arm h changes smoothly and the behavior of the cable network is well defined.

For a hierarchical input assembly, we formulate an equilibrium condition for each individual mechanical hinge j. If several cables i exert torques at j (refer to Figure 6.4 left), we sum up their contributions

$$\tau_j(\theta_j) = \tau_s(\theta_j) - \sum_i h_{ij} f_i = 0 \tag{6.1}$$

where the elastic spring behavior $\tau_s(\theta_j)$ may vary from joint to joint and its stiffness could be non-linear.

To determine the torque a cable i exerts on j (if any), we seek the first routing points $\mathbf{u}_{ij}$ and $\mathbf{v}_{ij}$ on the paths from j to the root and the leaves, respectively

(compare with Figure 6.4 right). If only one routing point is found, no torque is exerted. Otherwise, the moment arm

$$h_{ij} = \frac{\det\left[\mathbf{u}_{ij} - \mathbf{x}_j, \mathbf{v}_{ij} - \mathbf{x}_j\right]}{\|\mathbf{u}_{ij} - \mathbf{v}_{ij}\|} \tag{6.2}$$

is computed and the torque $h_{ij} f_i$ added to equation j.

Collecting all actuation forces f_i in a vector $\mathbf{f}$ and the per-joint angles θ_j in a vector $\boldsymbol{\theta}$, we can then find the equilibrium state $\boldsymbol{\theta}$ for a given $\mathbf{f}$ by solving the non-linear torque equations

$$\boldsymbol{\tau}(\boldsymbol{\theta}) = 0 \tag{6.3}$$

using the Levenberg-Marquardt algorithm [Levenberg, 1944; Marquardt, 1963], a variation of the Newtown method (see Section 3.2.1).

It remains to discuss how we express the global joint and node locations (in brown in Figure 6.5 left) with joint angles and the routing points w.r.t. these node locations (right).

As aforementioned, we assume our input to be hierarchical and therefore loop-free. Hence, we rely on a recursive definition as commonly used for rigs in character animation: the topology of the hierarchy is uniquely defined with a function r that maps a node k to its respective parent $r(k)$ (see Figure 6.5 left). The root node is mapped to the origin o. To transform the position $\mathbf{x}^k$ in frame k to the frame of its parent $r(k)$, we first apply a rigid transformation with constant rotation $\mathbf{R}^{k\to r(k)}$ and translation $\mathbf{t}^{k\to r(k)}$, then rotate counterclockwise by angle θ_j if the parent is a mechanical hinge j

$$\mathbf{x}^{r(k)} = \mathbf{R}^{r(k)}\left(\mathbf{R}^{k\to r(k)}\mathbf{x}^k + \mathbf{t}^{k\to r(k)}\right)$$

where the rotation $\mathbf{R}^{r(k)}$ is either

$$\begin{bmatrix} \cos(\theta_j) & -\sin(\theta_j) \\ \sin(\theta_j) & \cos(\theta_j) \end{bmatrix}$$

if $r(k)$ equals a joint j and the identity otherwise. Using this rule recursively, we transform positions from local to the world frame $\mathbf{x}^o(\boldsymbol{\theta})$, omitting the superfix o for positions in global coordinates. Note how nodes that move rigidly with a link depend on all joint angles θ_j on the path from this link's frame to the root. Hence, leaves depend on more angles than nodes closer to the root, leading to sparsity differences in derivatives of $\mathbf{x}(\boldsymbol{\theta})$ w.r.t. the joint angles.

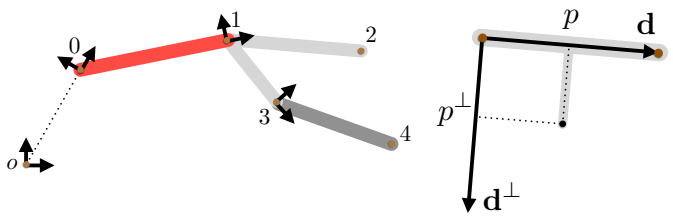


Figure 6.5: *To describe the kinematics of our hierarchical input, we use a recursive definition similar to the one used for rigs in character animation (left): this example consists of two hinges (at nodes 1 and 3) and three components (fixed link in red, link consisting of two segments in light grey, "leaf" link in dark grey). The topology is uniquely defined by the function* $r = \{(0, o), (1, 0), (3, 1), (4, 3), (2, 1)\}$. *We express routing points in local, per-segment frames* $\left[\mathbf{d}^\perp, \mathbf{d}\right]$ *with coordinates* $(p^\perp, p)$ *(right).*

To rigidly move the routing points with their respective links, we define local, per-link-segment frames (compare with Figure 6.5 right) where the difference vector $\mathbf{d}$, pointing from parent to child node, and its perpendicular vector $\mathbf{d}^\perp$ define the local link frame uniquely. Routing points $\mathbf{u}$ and $\mathbf{v}$ are then computed according to

$$\mathbf{x} + p^\perp \mathbf{d}^\perp + p\mathbf{d} \quad \text{with} \quad \mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad \text{and} \quad \mathbf{d}^\perp = \begin{bmatrix} d_y \\ -d_x \end{bmatrix}$$

where $\mathbf{x}$ is the node position of the parent. Note that $\mathbf{d}^\perp$ stays on the right of $\mathbf{d}$, independent of the link's orientation.

While we keep the coordinates of these routing points constant during simulations, we optimize their number and placement to closely match user-specified target poses. To this end, we collect all coordinates $p^\perp$ and p in a parameter vector $\mathbf{p}$.

6.4 Placing and Sizing Cable Networks

Before optimizing a cable network, a user defines target poses or a sequence of keyframes t by specifying desired locations $\tilde{\mathbf{x}}_s$ for a subset of the tree nodes. We then use standard inverse kinematics (IK)

$$\min_{\boldsymbol{\theta}} \sum_s \|\mathbf{x}_s(\boldsymbol{\theta}) - \tilde{\mathbf{x}}_s\|^2 + \gamma \|\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{prev}}\|^2$$

to solve for the target angles $\boldsymbol{\theta}^t$ where we add a regularization term that keeps the current solution close to the previous one, controlled by a weight γ.

6.4.1 Identifying the Network Topology

Given a target pose $\boldsymbol{\theta}^t$, we can compute the torsional spring torques $b_j^t = \tau_s(\theta_j^t)$ that require compensation, directly. And if we temporarily assume for a moment that the cables with their routing points $\mathbf{p}$ are known, we can compute the moment arms h_{ij} from the nodal positions $\mathbf{x}(\boldsymbol{\theta}^t)$. Hence, the equilibrium equations 6.1 become linear in the unknown actuation forces

$$\mathbf{H}^t\mathbf{f}^t = \mathbf{b}^t.$$

This observation paves the way for a simple heuristic to identify the topology of a cable network that is able to hit the target poses exactly: we generate many cables (more than a thousand) of varying "lengths" by randomly choosing the number and coordinates $p^\perp$ and p of their routing points in user-provided ranges smaller or equal to $[-1, 1]$ for $p^\perp$ and $[0, 1]$ for p. In other words, we randomly fill in rows of the $\mathbf{H}^t$ matrices, rendering these equation systems highly underdetermined. And because the probability of choosing two linear dependent rows at random is practically zero, the underdetermined, per-target systems can be solved exactly even if we constrain the forces $\mathbf{f}^t$ to be positive.

Such a cable network is highly impractical though. As previously mentioned, for a fully actuated assembly two cables per mechanical hinge – one on either side – are sufficient. Given a set of target poses, we aim at finding a network with far fewer cables than this upper bound.

To this end, we formulate a sparsity regularizer R_{sparse} that favors a cable to remain unactuated across all targets and solve the resulting constrained optimization problem

$$\min_{\mathbf{f}^t} R_{\text{sparse}}\left(\mathbf{f}^t\right) \text{ s.t. } \mathbf{H}^t\mathbf{f}^t - \mathbf{b}^t = 0 \text{ and } \mathbf{f}^t > 0 \tag{6.4}$$

where we bound the forces to only be pulled on.

For our sparsity regularizer, we approximate the $L1$-norm similar to Skouras et al. [2013]

$$R_{\text{sparse}}(\mathbf{f}^t) = \sum_i \left(\sum_t f_i^t\right)^\alpha$$

where index i runs over all cables and t over all targets. α is set to a fraction smaller than 1. We use $\alpha = 0.3$ for all our results.

If a cable remains unactuated for all targets, we remove it from the initial set, resulting in a small cable network with routing points $\mathbf{p}$ and actuation forces $\mathbf{f}^t$.

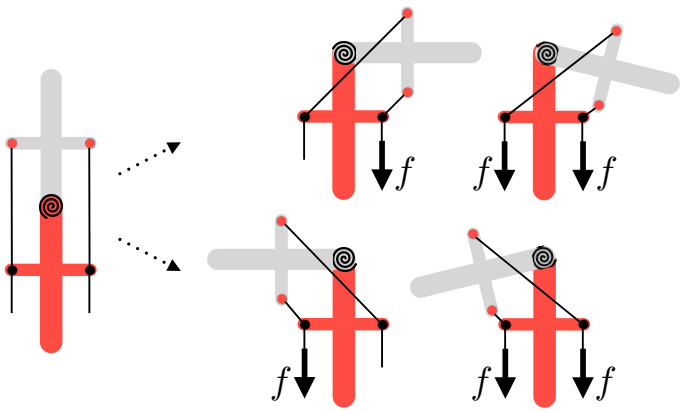


Figure 6.6: *Dependent on the sequence of actuation, the symmetric single-joint assembly with two cables (left) ends up in two completely different configurations: the force f is first applied to the right, then to the left cable (top row). If this sequence is reversed (bottom row), the assembly tilts to the left instead.*

6.4.2 Refining the Cable Network

When solving for sparse actuation forces by minimizing Problem 6.4, we ignore the path from the unactuated configuration to a particular target or, alternatively, from one keyframe to another. However, in general, the end configuration depends on the sequencing of actuations as we illustrate in Figure 6.6 with a two-cable example. Only if none of the moment arms h_{ij} flips its sign under active cables i is the end configuration independent of the sequence of actuation. We experimented with complementarity constraints to enforce path independence. Disallowing sign changes, however, is too restrictive and leads to networks with more cables than necessary.

To take path dependence into account, we simulate from unactuated to target configurations or from keyframe to keyframe each time we evaluate our refinement objective, constraints, and their derivatives. We thereby ensure that the static equilibrium 6.3 can be reached from the respective starting point.

To keep the actuated assembly close to the target poses during co-optimization of $\mathbf{p}$ and $\mathbf{f}^t$, we formulate a target matching objective

$$g_{\text{target}}(\mathbf{p}, \mathbf{f}^t) = \sum_t \sum_k \|\mathbf{x}_k(\mathbf{p}, \mathbf{f}^t) - \tilde{\mathbf{x}}_k^t\|^2$$

where index k runs over the nodes of the assembly and $\mathbf{x}_k^t = \mathbf{x}_k(\theta^t)$.

A second goal of our refinement optimization is the sizing of motors where we strive for the weakest actuation necessary, and the removal of additional

cables if possible. To this end, we add an *actuation* regularizer

$$R_{\text{actuation}}(\mathbf{f}^t) = \gamma_{\text{n}} \sum_t \|\mathbf{f}^t\|^2 + \gamma_{\text{s}} R_{\text{sparse}}(\mathbf{f}^t)$$

with two relative weights γ_{n} and γ_{s} to regulate their influence.

Minimizing actuation forces alone, however, renders the optimization problem unbound because we can compensate a smaller force f by increasing the moment arm h. Hence, we constrain the routing points to stay within reasonable bounds away from their rigid links $\mathbf{p} \in [\mathbf{p}_{\text{lower}}, \mathbf{p}_{\text{upper}}]$.

To avoid inconsistencies in the angles due to the periodicity of the sines and cosines defining our joint rotations, we add an additional regularizer that keeps the rotation angles close to the user-specified ones, normalized to the range $[0, 2\pi]$

$$R_{\text{period}}(\mathbf{p}, \mathbf{f}^t) = \gamma_{\text{p}} \sum_t \|\boldsymbol{\theta}(\mathbf{p}, \mathbf{f}^t) - \boldsymbol{\theta}^t\|^2$$

with a weight γ_{p}.

Eyeballing the expression for our signed moment arm (see Equation 6.2), we divide by zero if the routing points $\mathbf{u}_{ij}$ and $\mathbf{v}_{ij}$ adjacent to joint j collapse to a single point. To prevent close proximity between neighboring routing points, we add inequality constraints on their distance. Deferring a detailed discussion of finite-dimensional pulleys to the next section, these constraints safeguard against pulley-pulley collisions and we add similar constraints to keep joints and pulleys a safe distance away from one another.

In summary, we solve the constrained co-optimization problem

$$\begin{aligned}
\min_{\mathbf{p},\mathbf{f}^t} \quad & g_{\text{target}}(\mathbf{p}, \mathbf{f}^t) + R_{\text{actuation}}(\mathbf{f}^t) + R_{\text{period}}(\mathbf{p}, \mathbf{f}^t) \\
\text{s.t.} \quad \forall i, j: \quad & \|\mathbf{u}_{ij}(\mathbf{p}, \mathbf{f}^t) - \mathbf{v}_{ij}(\mathbf{p}, \mathbf{f}^t)\| > 2r_p \\
& \|\mathbf{u}_{ij}(\mathbf{p}, \mathbf{f}^t) - \mathbf{x}_j(\mathbf{p}, \mathbf{f}^t)\| > r_p + r_h \\
& \|\mathbf{v}_{ij}(\mathbf{p}, \mathbf{f}^t) - \mathbf{x}_j(\mathbf{p}, \mathbf{f}^t)\| > r_p + r_h \\
\text{and} \quad & \mathbf{p}_{\text{lower}} < \mathbf{p} < \mathbf{p}_{\text{upper}}
\end{aligned}$$

with r_p and r_h referring to pulley and mechanical hinge radii, respectively.

To avoid coupling between neighboring moment arms h_{ij} (see Figure 6.7), we use two instead of a single routing point per rigid link (top row). While these almost double the number of pulleys per cable, it has an important advantage when generating the geometry for our output assemblies (bottom row, see Sec. 6.5): when rotating the oriented triangle $(\mathbf{x}, \mathbf{u}, \mathbf{v})$ about the

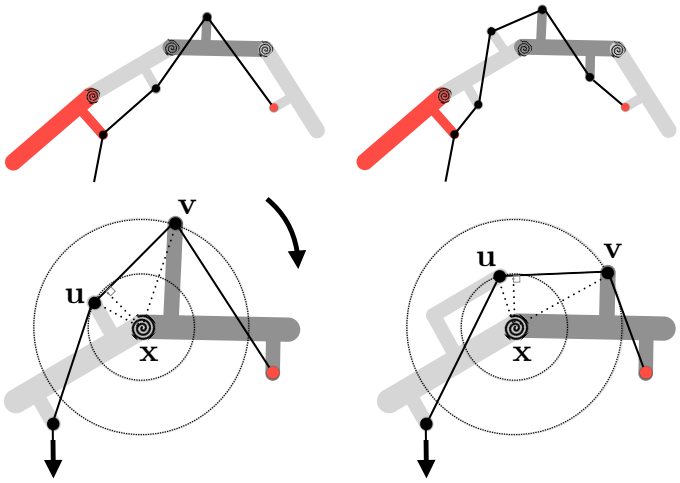


Figure 6.7: *We use two instead of a single routing point per cable and link (top row). While this adds to the complexity of the manual assembly task (the pulleys almost double per cable), it has a an important advantage when generating geometry for the final assemblies: we can rotate oriented triangles* $(\mathbf{x}, \mathbf{u}, \mathbf{v})$ *about the hinge axis without changing the torque equilibrium in any given pose.*

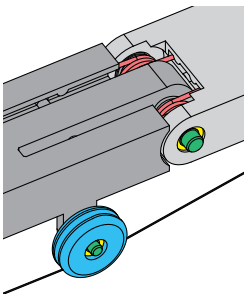
hinge axis, the moment arm and resulting torque remains the same, independent of the pose we are in. To favor a particular solution in these joint-cable subspaces, one can add a weak regularizer on the routing points. Note that the coupling due to the constant force magnitude along the cable remains.

To compute gradients, we first simulate to equilibrium $\boldsymbol{\tau}(\boldsymbol{\theta}) = 0$. Collecting the routing points and per-target actuations $\mathbf{q} = (\mathbf{p}, \mathbf{f}^t)$, we then use the implicit function theorem (see Section 3.3.1)

$$\frac{d\boldsymbol{\tau}(\mathbf{q}, \boldsymbol{\theta}(\mathbf{q}))}{d\mathbf{q}} = \frac{\partial \boldsymbol{\tau}(\mathbf{q}, \boldsymbol{\theta})}{\partial \mathbf{q}} + \frac{\partial \boldsymbol{\tau}(\mathbf{q}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}(\mathbf{q})}{\partial \mathbf{q}} = 0$$

to compute the analytical gradients $\frac{\partial \boldsymbol{\theta}(\mathbf{q})}{\partial \mathbf{q}}$. Remaining derivatives are computed at runtime using symbolic differentiation [Guenter, 2007].

6.5 Fabrication Considerations



To fabricate our cable-driven kinematic chains and trees, we use a combination of 3D printing and standard off-the-shelf parts as illustrated in the inset on the left with a computer-aided design (CAD) drawing: our mechanical hinges consist of a standard helical torsion

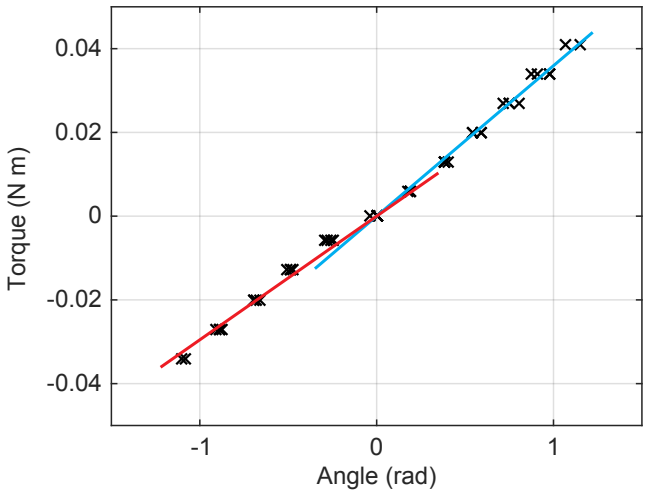


Figure 6.8: *A characterization experiment reveals a linear but asymmetric behavior with k_{pos} in blue and k_{neg} in red.*

spring (in red), a shaft (in green), and a bearing (in yellow) at either end of the shaft. For the routing points we use a similar design with a pulley (in blue) and for our cables (in black) fishing line.

Friction It is of paramount importance to reduce friction at the hinges and pulleys to a minimum and bearings serve this purpose. However, in physical kinematic assemblies friction is unavoidable and as a rule of thumb, friction in pulleys leads to a decrease in tension along the cable from the root to the leaves of the hierarchy. At mechanical joints, we may observe several equilibrium states in a local neighborhood.

Torsional Springs Our standard torsional springs are labeled as linear and symmetric. However, a characterization experiment reveals a linear but asymmetric behavior (see Figure 6.8) even after oiling the springs to avoid friction between spring windings. To avoid a C^0 spring torque, we use a sigmoid function

$$\tau_s(\theta) = s(\theta)k_{\text{pos}} + (1 - s(\theta))k_{\text{neg}} \quad s(\theta) = \frac{1}{1 + e^{-\beta\theta}}$$

in our simulation and co-optimization where we set β to 100.

Spring Stiffness Range When choosing a stiffness range for our torsional springs, we strive for balancing the trade-off between meeting our quasi-

static assumption (lower stiffness bound) and the maximum actuation forces (upper bound). If springs are too soft, a quasistatic simulation is insufficient due to the non-negligible effect of inertia. If the control forces are too high, actuation by hand or with inexpensive, low-end servos is infeasible.

A dynamic simulation of our cable-driven kinematic assemblies is straightforward

$$I\frac{d^2\theta_j}{dt^2} + C\frac{d\theta_j}{dt} + \tau_s(\theta_j) = \sum_i h_{ij} f_i$$

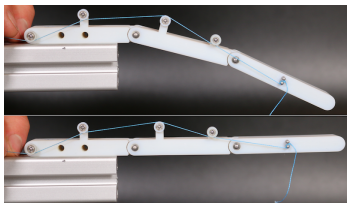
with moment of inertia I and damping C. However, the corresponding cable network optimization is difficult as it requires a space-time formulation [Witkin and Kass, 1988] where error may accumulate over time.

Because of the use of bearings at joints and pulleys, the damping C in our system is small. Hence, the frequency of vibration at a joint is expected to be close to the natural resonance frequency

$$f_n = \frac{1}{2\pi}\sqrt{\frac{k}{I}}$$

where k is the linearized spring stiffness. Given a lower bound on the frequency and an estimate of the moment of inertia, we can use above relation to compute a first order approximation of the lower bound of the stiffness range that renders a quasi-static assumption valid. An upper bound can be computed from a desired maximum force magnitude.

Gravity Compensation Due to this upper bound on stiffness, our assemblies sag under gravity and require compensation as illustrated in the inset



on the left: the initially straight, unactuated three-link assembly "bends" under gravity (top) and only if accounted for, we can hit the target of a straight, actuated assembly (bottom). To compensate for gravity at a joint $\mathbf{x}_j$ (see Figure 6.9), we keep all other torsional springs fixed, transform centers of mass $\mathbf{c}$ of child links, joints, and pulleys from local to global coordinates, then add their gravitational torques

$$(\mathbf{c} - \mathbf{x}_j) \times \begin{bmatrix} 0 \\ -mg \end{bmatrix}$$

to the equilibrium equation 6.1 for joint j. W.l.o.g. we assume gravity to point in the negative y direction. m is the mass of the respective link, joint,

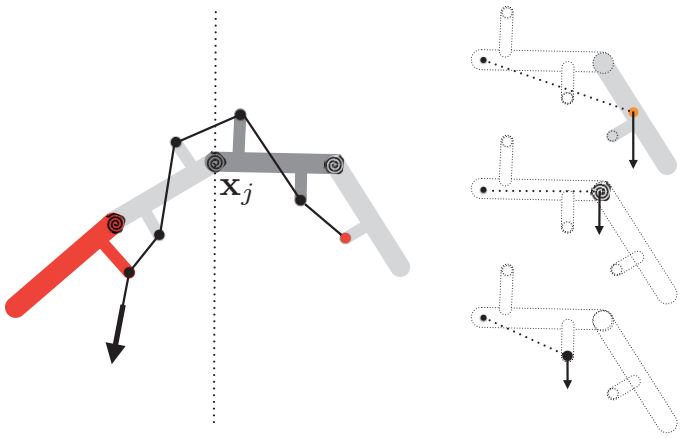


Figure 6.9: *We compensate for gravity at a particular joint $\mathbf{x}_j$ by holding all other torsional springs fixed (left), adding up gravitational torques of rigid links (right, top), torsional springs (right, middle), and pulleys (right, bottom) on the path from j to the leaves.*

or pulley and g is the standard gravity. During network topology identification (see Equation 6.4), we ignore gravitational effects of pulleys as the randomly chosen cables with their many routing points would lead to a massive weight, rendering the optimization unrealistic. During our refinement, gravity compensation is fully switched on.

Finite-Dimensional Pulleys Pulleys are not dimensionless and we cannot route cables through their centers. As shown in Figure 6.10 we account for their finite dimension by adding an offset r_p to all moment arms h_{ij} during simulations and optimization where r_p is the radius of the pulley. Note that this offset is – independent of the pose – of constant value r_p as the cable is perpendicular to the h_{ij}s and the pulleys are circular. As illustrated in Figure 6.10 in blue, we wrap cables once around the pulleys to ensure that they do not detach during actuations. We do that in a consistent manner for all our pulleys.

Generating Geometry For the database of supported springs, we manually generate negative and positive hinge geometry that supports mounting of respective springs by snapping them in place. The joint geometry is instantiated and remaining geometry automatically generated by our system. If a routing point collides with a rigid link other than the one it corresponds to, the user rotates the oriented triangle to avoid these collisions as illustrated in Figure 6.7 bottom.

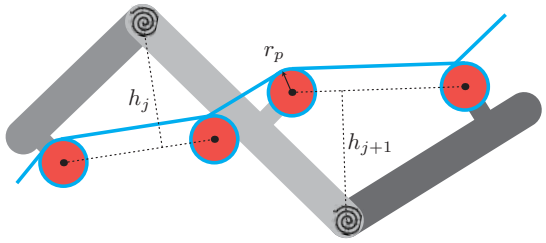


Figure 6.10: *We account for the finite dimension of pulleys by offsetting all moment arms h_j and h_{j+1} by the constant pulley radius r_p. To ensure that cables do not detach during actuation, we wrap them once around the pulleys (see cable in blue).*

Model	links	joints	targets	cables	pulleys
Fighter	12	11	2	7 (4000, 12)	31
Hand	13	12	4	4 (3500, 12)	26
Gripper	13	12	2	4 (6000, 18)	48

Table 6.1: *We summarize the complexity (number of links, joints, specified targets, cables, and pulleys) of our three example assemblies. Our two-step optimization reduces the initial number of cables x to y after the first, then to z after the second step, formated in column "cables" using z (x, y).*

6.6 Results

We have used our computational tool to size and place cable networks for a total of three examples: a gripper (Figure 6.12), an animatronic hand (Figure 6.13), and an animated fighter character (Figure 6.1). We summarize key features and complexity of the physical assembly in Table 6.1.

Fabrication The rigid links for all our examples were printed on an Object Connex 350 with Vero White and Vero Black. After printing and cleaning, manual assembly of a model takes a few hours. We use double torsional springs (parts 8548 and 8555, Lesjofors AB), with tabulated spring stiffnesses of 0.0484 and 0.0950 Nm rad^{-1}, respectively. Pins and bearings are press fitted, speeding up the assembly process.

Validation We validated our cable network optimization on the lower body part of our Fighter character seen in Teaser 6.1. It is a kinematic tree

Figure 6.11: *We validate our cable network optimization on the lower body of our Fighter character on two target poses (bottom, top) by controlling the cable forces with standard Newton meters. The simulated poses (left) match the physical poses (right) well.*

with multiple cables acting on the same link and an example that requires gravity compensation. We built a setup to replicate the optimized actuation forces experimentally. To this end, we use standard Newton meters pulling on the three ends of the cables. Excursions are adjusted manually in order to match the prescribed forces. As we show in Figure 6.11, we validate the performance of our target-matching objective on two target poses specified by the user. While we observe a slight mismatch for the lower knee joint of the right leg (bottom, right in Figure 6.11), it can be seen that simulated and physical poses match well. This validates our modeling assumptions.

Performance For the lower body of the Fighter, we chose 1600 cables at random, then used our topology identification (Equation 6.4) to reduce it the number of cables to 8 in 25 seconds. We then use our refinement to further reduce the number of cables to a total of 3. Our refinement takes 181 seconds on this example. Note that for hierarchies with several branches attached to the fixed link (compare with skeleton in our teaser), we optimize each branch independently. The lower body of the Fighter is the most complex branch with regards to time complexity, hence, representative.

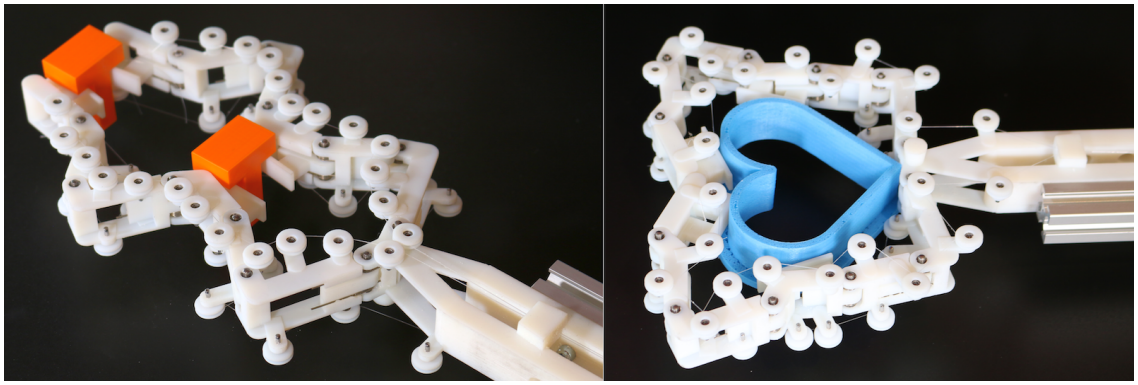
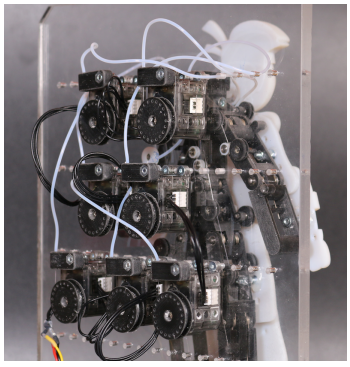


Figure 6.12: *Our gripper is optimized to pick up two T-shapes when one cable is actuated, and a heart-shape if only the other is actuated.*

Gripper As a functional example, we designed a gripper to be able to pick up two kinds of differently shaped objects: in one configuration we can pick up two T-shaped objects, in the other a heart-shape. No gravity compensation is needed as the gripper operates in the horizontal plane. The gripper is symmetric and uses two cables on either side. As the gripper is operated manually, we wish to reduce the control complexity and optimize one cable per target pose and side.

Hand Increasing the complexity, we optimize an animatronic hand with three fingers and a thumb as well as an actuated wrist. Note that the thumb is operated on a plane that differs from the horizontal one, hence, requiring gravity compensation where we work with projected gravity. The closing motion of each finger is optimized to follow four keyframes. Actuation of the middle fingers is coupled to movement of the wrist (fixed link) to yield a more realistic closing of the hand. The cables of the thumb and pinky end on the palm and we use Bowden cables between the palm and the wrist to avoid exertion of torques on the joint connecting wrist with palm. The hand assembly can perform a wide range of gestures as can be see in Figure 6.13. This example illustrates that our approach extends to the third dimension by allowing joints to lie in different planes.



Fighter Our technique can be used to create animated, mechanical characters by actuating the cables with servos. Our Fighter is driven by a total of seven servos as shown in the inset on the left. We use Dynamixel XL-320 servos, with a stall

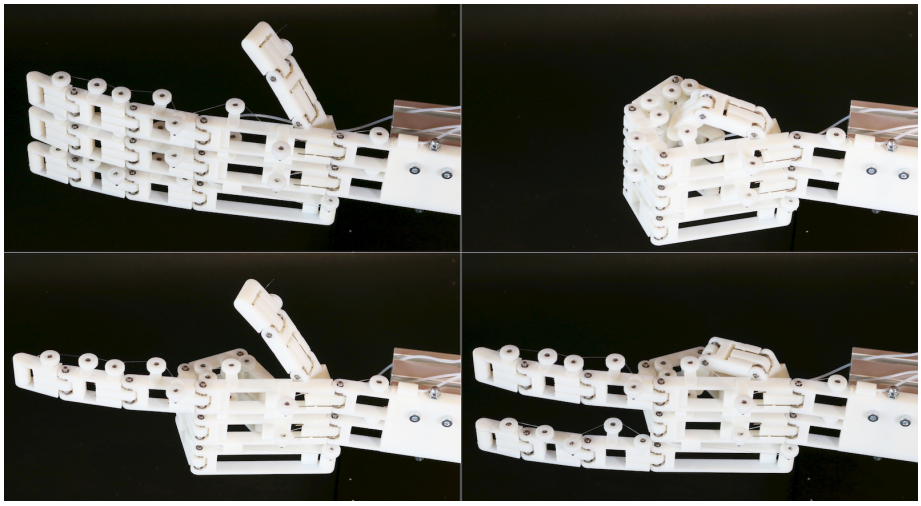


Figure 6.13: *Our animatronic hand can perform a wide range of gestures. Its thumb is operating in a different plane than the remaining fingers and the wrist.*

torque of 0.39 Nm. The servos are located outside the character allowing for a lightweight design and we again use Bowden cable to connect cables to servos. Note that the Bowden cables lead to non-negligible friction. As the servos are not force controlled, we adjust the excursions of the servos to match the user-specified keyframes.

6.7 Conclusion

We have devised a method to design complex cable networks for animating mechanical characters or controlling the motion of functional assemblies. We have introduced a reduced coordinate formulation that links joint angles and routing points to control forces, enabling co-optimization of routing and actuation. To identify the topology of a cable network, we route a large set of cables from root to leaves, attaching them to rigid links with routing points chosen at random, then removing redundancy. We further refine these networks, taking path dependence into account.

Limitations For our gripper design, we assume contact forces to be negligible, an assumption not holding true when picking up objects of considerable weight. When in contact with the lightweight T-shapes or the heart, we do observe deviations from the intended gripping poses if we pull the cables beyond the optimized actuations. External forces lead to additional,

pose-dependent torques that we could compensate for by adding them to the equilibrium equations as we did for the gravitational torques.

Future Work There are several challenges remaining. Our technique assumes hierarchical input and an extension to mechanisms with loops is left as future work. We further plan to extend our technique to spatial input, adding support for joints with more than a single degree of freedom. As another interesting direction, we would like to explore optimization of a cable-network for assemblies where the source of compliance is not added by springs but by flexures, a foam, or silicone skin instead. Note that our spring torques allow for non-linear behavior as would be expected when using foam or silicone. Relaxing our assumption on quasi-statics opens another interesting avenue, requiring space-time optimization to size and place cable networks to control the dynamic behavior of assemblies.

CHAPTER 7

Conclusion

This thesis addresses the problem of making the design of robots and animatronics easier, more efficient and, more accessible. The work we described in the previous chapters focuses on empowering novices and experts with intuitive tools to create manufacturable characters. These tools allow to directly specify high-level goals, while the algorithms underneath take care of providing a design that satisfies these goals. In Chapter 3, we discussed the models and methods employable for the implementation of the systems presented later in Chapters 4, 5, and 6. While these models and methods are general and serve as a basis for the design tools, the problems addressed by this thesis require specific tailoring of the algorithms to be properly treated. The chosen degrees of freedom, design space parameterization, and objective functions to measure optimality, play a crucial role in a successful implementation. These aspects closely relate to the specific instances of the problems and affect the capabilities of the tools and the final design systems. For this reason, we analyzed and developed solutions for three distinct problems. In combination they allow us to reach the goal of creating personalized robotic and animatronic creatures.

First, we looked at the problem of designing legged robots and their open-loop controllers. We aimed at developing a versatile system which allows the user to create robots of different sizes and morphologies without any prior knowledge of robotics. We provided easy-to-use tools for the design stage, which work together with an optimization algorithm capable to interactively preview the emerging locomotion trajectory. We demonstrated how to provide interactivity by introducing a two-step approach. The first step uses a simplified physics model, which makes the optimization fast.

The second employs a full physics simulation that can better predict dynamic behaviours. This allows the user to detect stability problems before the costly and time-consuming manufacturing process. We note that, when a complete physics model is applied to a simulation tool, rather than to the optimization, interactivity is not broken. When the user is satisfied with a design, our system automatically generates the code for an off-the-shelf microcontroller and the geometry ready for 3D-printing. Although functional, the geometry generated by our system does not contain any aesthetic components. However, the resulting structure can be augmented and beautified by the user, according to his or her preferences. To test our tool, we designed six robotic creatures featuring a different number of legs and motors. Three of these designs were fabricated and examined in the real world. While being successful in its task, the system presented in Chapter 4 cannot produce robots which are as agile as the creatures found in Nature. The origin of this limitation resides partially in the simplified model which the optimization relies on, and partially in the commodity hardware we use to operate the robots. We can find more agile creatures in the field of robotics, but they integrate customized hardware in the robot design. The adoption of commodity hardware is desirable to make our tool more accessible and, hence, reach a broader audience.

By looking at Nature, one can notice how rigid and deformable bodyparts work synchronously to fulfill locomotion tasks elegantly and efficiently. However, the design tool described above does not allow for flexibility and we, therefore, also researched compliant structures to be integrated into our robotic designs. The recent development of fabrication technologies and new materials, pave the way to possibilities that could not be explored before. Examples include custom-made, flexible body parts to be incorporated into our robots. The creation of compliant structures, designed to provide certain functions, was broadly studied in the field of mechanical engineering. However, the approaches proposed to design compliant mechanism assume small motions, and close to linear behaviour. To be considered valuable for our purpose, the compliant mechanisms under analysis should be capable to perform large displacements emerging from a nonlinear behavior. Therefore, we looked at the problem of designing compliant mechanisms with a broader range of motion.

In Chapter 5, we presented a comprehensive process to create compliant mechanisms. The system takes as input a rigidly articulated assembly, composed of rigid bodies and traditional joints, and converts it into compliant design. Although our system takes a traditional input mechanism as-is, the user has the option to edit the input assembly using existing tools recently developed in the field of digital fabrication. A complete design pipeline

could be, for example, to first create the linkage topology [Thomaszewski et al., 2014], then editing its shape while ensuring the functionality [Bächer et al., 2015], and ultimately converting the assembly to a compliant mechanism using our method. From the input linkage, we read the topology and the initialization parameters used to replace traditional joints with flexures. While supplying a meaningful initial structure, a naive replacement does not ensure motion preservation and cannot possibly account for the different problems a compliant mechanism can exhibit. Therefore, a specific optimization is required to handle the distinct pitfalls a user can encounter when designing compliant mechanisms. The optimization needs to account for objectives which are critical for the correct functioning of the final device. First, we ask the optimization to preserve the mechanism end-effector motion, hence, preserving the assembly's functionality. Second, we demand from the optimization to imperatively avoid the damage of the flexible parts during actuation. In the field of mechanical engineering, different techniques were proposed to prevent material failure. Such techniques assume the possibility of computing the volumetric stress of the material under deformation. Using a volumetric model during simulation would allow us to obtain directly the stress information we require, but it would also make the simulation step prohibitively expensive. The work we present in this thesis leverages the efficiency of a centerline model [Bergou et al., 2008; Bergou et al., 2010] and, at the same time, provides an approximation for a full volumetric strain, which can be used to avoid material failure. Having in place a motion preservation and material failure objective, we also formulate objectives for other important properties such as a low-torque requirements, and a high stiffness in the direction orthogonal to the motion. These four objectives, together with an additional one ensuring a collision-free movement, constitute a minimal set of conditions that must be met to manufacture a functional compliant mechanism.

To be put in motion, traditional mechanisms need to have an equal number of degrees of freedom and actuators. Installing fewer actuators into mechanical assemblies is desirable since it results in lighter and more energy efficient machines. For example, linkages featuring a single degree of freedom are very attractive since they require a single motor or crank to be driven. However, the fewer degrees of freedom we desire, the more difficult the design task. Compliance helps reducing the of degrees of freedom, thanks to its innate property of introducing soft constraints. Therefore, characters composed of simple structures such as chains and trees would require fewer actuators when designed with compliant articulations. While being a necessity, compliance alone might not be sufficient to allow the actuation of a character with multiple degrees of freedom. To gain more control while avoiding

the usage of many motors, we can route a network of cables through a character to passively propagate the motion along the structure.

Inspired by the graceful looking movements found in Nature, tendon-driven systems have been investigated in the field of computer graphics, robotics, and biomechanical engineering. We find work in the direction of designing folding surfaces [Kilian et al., 2017], actuate manipulators [Fang et al., 2004; Behzadipour, 2005] or create controllers for specific string-based structure designs [Rucker and III, 2011; Whitney et al., 2014]. In contrast, as described in Chapter 6, we target cable-driven linkages which can be animated and controlled depending on the user preferences. We tackle the problem of finding the right amount of cables to be employed, where to place the pulleys, together with how much force we need to apply to animate a physical character. The co-optimization of control and design for the motion allows our system to put the user in control of the trade-off between the cable-network complexity, motion fidelity, and actuation cost. Our system allows the user to animate a virtual character using inverse kinematics, where for each one of the joints we place a virtual actuator to fully determine the pose of the figure. Subsequently, we split the task of reproducing the motion provided by the virtual motors into two stages. First, we identify a viable topology for the cable network, which ensures to keep the character in place for each user-defined keyframe. Among the different possible solutions we prefer the one requiring fewer cables. Then we refine the pulleys location maintaining the topology fixed, with the intention of further improving the motion and ensuring the existence of a sequence of forces which can reproduce the input animation. Whenever the user is satisfied with the design produced by the optimization, we semi-automatically generate the geometry for printing.

7.1 Future Directions

This thesis describes a set of novel tools and algorithms to help expert and casual users to design and manufacture animated physical characters. We approached and looked at this design problem from several angles. However many aspects were not covered in this work, including potential applications and their related challenges, leaving many exciting future research directions to be explored.

7.1.1 Project-Related Research Directions

Designing 3D-printable figures is a challenging task, even more so if the resulting characters should be animated while ensuring stability, preventing material failure, or keeping them light. The long-term objective of this thesis is to capacitate users of any level of expertise to design legged robots which can exhibit agile and natural looking motions. While the work presented in the previous chapters brings us closer to this goal, many obstacles need to be overcome before we can reach it. First, in Chapter 4 we base our optimization on a physics model which does not incorporate full dynamics. This simplified model, while sufficient for the medium pace walking gait we designed for our robots, might need to take into account more information regarding dynamics, when dealing with faster motions. Moreover, fast-moving strides often include a flying phase where no leg is in contact with the ground. Our optimization imposes to have at least one foot on the ground at any point in time. Therefore, we would need to make some changes to the formulation at the core of our algorithm, to allow for fast-paced gaits. While the necessary adjustments to account for dynamics and flight phase can be directly integrated into our system, different tests need to be performed to understand the right level of complexity for such a tool. For example, instead of limiting the system to only allow for motion trajectories explained by linear momentum, one could expand the trajectory space to also account for angular momentum. Some investigation should then go into finding how much more powerful the new model is compared to the old, and how much efficiency is traded off. Depending on the result, one might decide to pursue a more complex model, which operates directly on the torque generated by the motors and which computes the ground reaction forces explicitly. However, such a choice would bring us towards the methods used in the field of robotics, most definitely break interactivity, and require specialized hardware to operate the designed creatures.

In this thesis, we demonstrated how to implement novel tools to design legged robots and their relative motion style, convert traditional assemblies in compliant mechanisms, and animate physical characters by pulling cables. Although the described systems are individually successful in their tasks, the next step for future research is to combine these three tools and their relative technologies into a single framework. In order to achieve a single system pipeline, the formulations described in the previous chapters need to undergo some edits, and multiple experiments must be carried out.

First, in Chapters 5 and 6, we make use of quasi-static assumptions to simulate our mechanisms. This assumption might not be valid when such mecha-

nisms are mounted on a walking creature. To be able to predict the behavior of compliant and cable-driven limbs accurately, we need to take into account the effects of dynamics. For example, the moment carried by the structures might cause oscillations that could play a prominent role when a locomotion gait is executed. On the one hand, when not accounted for, oscillations could bring the manufactured robots to behave differently than anticipated by the optimization, leading to instability and the creature's fall. On the other hand, when properly considered and included in the optimization, oscillations could come into favor by allowing for faster and more efficient leg motion. Accounting for the dynamic behavior would improve the predicting power of our framework, but it would also require the introduction of a time dimension, over which we would need to integrate. In the context of highly dynamic gaits, our simulation must have a high level of accuracy to ensure the stability of the physical character. Such a degree of accuracy asks for implicit integration schemes, which are expensive to compute due to the nonlinear relationship between the variables.

Second, in Chapter 5 we account for external forces when preserving the motion of a converted compliant mechanism, and demonstrate the validity of our objective, building a remotely controllable car with a compliant steering mechanism. However, when a compliant mechanism is used as an integral part of a more complex robot, it might undergo heavier loads compared to our car example. Such a load could be inferred by the weight of the robot or the friction caused by the ground, and it might be too high in magnitude to be endured by the flexible parts. Thankfully, in such a case, our system would be able to detect a possible failure and would try to stay away from the fracture point. However, depending on the load and the material, it might not always be possible to avoid failure while preserving the mechanism's functionality. To solve this problem, one could analyze where the high-stress peaks appear during actuation and decide to convert it back to a traditional joint. While such a choice is already being viable option, a dedicated implementation could point to the weak spots and suggest topological changes to create a hybrid traditional-compliant mechanism to improve the results.

Lastly, in Chapter 6 we focus on the production of animations for 2D physical characters, whereas the previous two chapters looked into the design of fully 3D structures. Extending our framework to support the optimization of three-dimensional, cable-driven structures, would give more artistic freedom to the users while enlarging the design space of stable locomotion parameters. We note that a 3D mechanism can be built combining planar submodules, as demonstrated by our animatronic hand example (Figure 6.13). However, to be able to design an entirely spatial cable-driven structure, we

would need to adjust our gravity compensation term to be orientation dependent. Furthermore, when extending to the third dimension, the pulleys we used when designing planar structures, might not be the best choice to route the cables. Other options to route the cables could be explored, such as low-friction metal rings, but a dedicated attention must be paid to collisions. As an additional extension to our system, 3D compliant connectors could also be included, to grow the space of possible solutions currently dictated by the operating planes of the torsional springs.

7.1.2 General Research Directions

When developing design tools to create physical characters, it is common to face the problem of topology optimization. While for the case of designing robotic creatures (Chapter 4), we leave the opportunity to choose the robot's morphology to the user, the automation of such a decision is beneficial when designing compliant mechanisms or cable-driven linkages. In Chapter 6 we solve for a sparse set of cables capable to actuate a 2D character, given a set of target keyframes. The discrete problem of finding a topology which best suits an optimization objective is combinatorial; hence we opted for relaxing its discrete nature to be able to solve it efficiently. However, the type of formulations used to relax the topology problem can make the Hessian matrices ill-conditioned and present high nonlinearities. For this particular reasons, we decoupled the topology optimization from the remaining design tasks, leading to a two-step algorithm. In general, because of their intrinsic difficulties, many combinatorial problems remain open and would deserve proper investigation. Both the systems presented in Chapters 5 and 6 would benefit from a general purpose topology optimization technique. First, we might be able to efficiently explore the space of compliant mechanisms with fewer parts and that require less torque to be actuated. Second, for the case of the cable-driven characters, we could take into consideration the full design problem instead of splitting it into two. Looking simultaneously at the discrete and continuous problem may, for example, increase motion fidelity while keeping the complexity of the character low.

Modern computer numerical controlled (CNC) machines are powerful allies for nowadays makers and hobbyists. Milling machines, 3D-printers, and laser-cutters made prototyping a faster, more precise, and less error-prone process. Every physical result presented in the past chapters was built using one (or a combination) of the digital manufacturing techniques mentioned above. While having been of incredible help, this machinery still has room for improvement to increase the productivity during fast prototyping. For

example, when analyzing the entire creation process of a mechanical creature (from a model's conception, through its design, to the assembling procedure), the most time-consuming part of the pipeline is often the 3D-printing. On the one hand, this limitation highlights the need for more powerful design tools to reduce the number of costly 3D-printing iterations. On the other hand, an increase in 3D-printers performances would largely improve the user's experience while utilizing our tools.

Additionally, the limited availability of 3D-printing material types constitutes another restriction to the manufacturing process. At consumer-level, 3D-printer producers mostly support the use of plastic- and rubber-like materials. Although other materials with different properties can be used, 3D-printers often fail to complete the printing process accurately and reliably. In Chapter 5, where strength and flexibility were essential for our flexures, we had a hard time finding a reliable material. Also, most of the materials we experimented with, are highly affectable by the surrounding environmental conditions, and their properties change over time. For example, even when left untouched, the fabricated prototypes can exhibit unwanted plastic deformation induced by gravity, which damages the performances of both rigid and compliant mechanical machines. Moreover, the curing process of certain materials can take a variable number of days to complete. During this time, properties such as the yield strength can significantly deviate from the ones provided by the 3D-printer manufacturers. These limitations call for further research in existing materials to make them more reliable, both during and after the printing process.

This thesis emphasizes how difficult it is for robots to mimic an organic movement. Such an accomplishment is even harder to achieve when the mechanics of the used hardware is inherently different than flash, bones, and tendons. We try to emulate the behavior of the last two structural components with 3D-printed parts and fishlines, but standard actuation for mechanical characters (i.e., motors) operates differently than muscles. Devices capable of performing similarly to muscles exist, and among them, we find electroactive polymers (EAP). These polymers can exhibit different properties and functional abilities [Bar-Cohen, 2001], for example, they can change in form by applying an electric field; such behavior resembles the activation of biological muscles. Inversely, electroactive polymers can also produce a change in electric charge or voltage when mechanical forces are applied. By reading the rates of voltage change, we could measure the strain induced by a deformation, thus making EAP well-suited to be used as sensors. Sensing the environment is a necessity when generating highly dynamic movements, to accordingly adjust the control actions of a robot. Therefore, thanks to their actuation and sensing properties, we can conclude that an advance in EAP-

related technologies could improve the efficiency of robots and animatronics. By making them lighter and stronger, ensuring low power consumption, and providing embedded sensing capabilities, electroactive polymers could push forward the abilities of the current generation of robots.

Automatically creating control policies that can produce stable locomotion is a crucial feature for an easy-to-use tool to design legged robotic creatures. The system we proposed in Chapter 4 focuses on medium-speed walking pace and the resulting manufactured robots can stably locomote using an open loop control strategy. However, faster motions would require feedback to be integrated to change the action strategy online, thus adapting to unpredicted changes in the environment. A multitude of techniques to produce robust control policies was proposed in the field of robotics (see Section 2.1). These strategies often rely on actively sensing the environment, providing feedback to the computing unity, and accordingly adjust the torques to apply to the actuators. In contrast, other proposed methods are based on passive dynamic walking [Iribe and Osuka, 2006; Vargas and González-Hernández, 2013; Zang et al., 2017], allowing the robot to walk by virtue of dynamics only. Because no computing power is required, and stability comes exclusively from the mechanical properties of the design, these devices necessitate less electrical energy to operate and are cheaper to produce. However, designing creatures capable of passively locomote and react to the environment proves to be a difficult task, above all if specific artistic requirements need to be met. An interesting and exciting venue for future work would be to investigate the possibilities of integrating passive dynamics in a tool that helps users designing walking creatures, while still providing artistic freedom.

Producing convincing, organic, and natural motions is essential for animatronic figures. Aside from the movements, other aesthetic components are also of crucial importance to deliver visually appealing characters. Stuffing and upholstering mechanical creatures is, therefore, an interesting topic for makers, hobbyists, and the industry of animatronics. Ways of covering and stuffing a moving creature were already investigated [Bickel et al., 2012; Bern et al., 2017]. While being successful, they tackle specific problems, leaving room for further explorations. For example, an interesting future research direction could be the investigation of methods capable of covering and stuffing animatronics, while not interfering with the functional parts (i.e., actuators and sensors) and providing a plausible and/or pleasing tactile feedback for the interacting person.

The design tools we propose heavily rely on solving nonlinear objectives under nonlinear constraints, most of them being nonconvex. The exploration of

novel optimization techniques, together with alternative problem formulations, can improve the quality and efficiency of our tools and their produced results. Moreover, the effectiveness of our frameworks is based on physics simulation, since our goal is to fabricate the resulting creatures or mechanisms. Differently from animators, the *reality gap* is a major obstacle that roboticists, mechanical engineers, and digital fabricators have to face. The reality gap refers to the difference between simulated and physical environments, often making the prediction of simulation useless if not tweaked accordingly to specific scenarios. A decrease in the discrepancy between the digital world and physical reality would benefit not only our tools but all robots- and animatronics-related products. Faster and more precise simulation techniques would allow designers and engineers to converge to a final product quicker while decreasing the number of required physical prototypes. These advantages would lower the cost of production for robots and animatronics, with regard to time, personnel, and material.

7.1.3 Final Considerations

In conclusion, this thesis attempts to bring to the robotics, digital fabrication, and mechanical engineering communities a novel set of tools to help the manufacturing of robots and animatronics. We highlighted pros and cons of previously proposed methods, and introduced innovative solutions to overcome their limitations. The techniques we presented aim at bridging the gap between robots and makers, democratizing the personalization of functional and expressive mechanical characters. By pushing the boundaries of current robotic and animatronic designs, we hope to help and inspire future research directions, to boost towards a new and improved generation of robots.

References

[Agrawal et al., 2013] Shailen Agrawal, Shuo Shen, and Michiel van de Panne. Diverse motion variations for physics-based character animation. *Symposium on Computer Animation*, 2013.

[Albanesi et al., 2010] Alejandro E. Albanesi, Victor D. Fachinotti, and Martin A. Pucheta. A review on design methods for compliant mechanisms. *Mecanica Computacional*, 2010.

[Auerbach et al., 2014] Joshua Auerbach, Deniz Aydin, Andrea Maesani, Przemyslaw Kornatowski, Titus Cieslewski, GrÃ©goire Heitz, Pradeep Fernando, Ilya Loshchilov, Ludovic Daler, and Dario Floreano. RoboGen: Robot Generation through Artificial Evolution. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 136–137. The MIT Press, 2014.

[Bächer et al., 2012] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. In *Proc. of ACM SIGGRAPH '12*, 2012.

[Bächer et al., 2014] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 33(4):96:1–96:10, 2014.

[Bächer et al., 2015] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. Linkedit: Interactive linkage editing using symbolic kinematics. *ACM Trans. Graph.*, 34(4):99:1–99:8, July 2015.

[Bar-Cohen, 2001] Y. Bar-Cohen. *Electroactive Polymer (EAP) Actuators as Artificial Muscles: Reality, Potential, and Challenges*. Press Monographs. SPIE Press, 2001.

[Behzadipour, 2005] Saeed Behzadipour. *Ultra-high-speed Cable-based Robots*. PhD thesis, University of Waterloo, 2005.

[Bergou et al., 2008] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. *ACM Trans. Graph.*, 27(3):63:1–63:12, August 2008.

[Bergou et al., 2010] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Discrete viscous threads. *ACM Trans. Graph.*, 29(4):116:1–116:10, July 2010.

[Bern et al., 2017] James M. Bern, Kai-Hung Chang, and Stelian Coros. Interactive design of animated plushies. *ACM Trans. Graph.*, 36(4):80:1–80:11, 2017.

[Bickel et al., 2012] Bernd Bickel, Peter Kaufmann, Mélina Skouras, Bernhard Thomaszewski, Derek Bradley, Thabo Beeler, Phil Jackson, Steve Marschner, Wojciech Matusik, and Markus Gross. Physical face cloning. *ACM Trans. Graph.*, 31(4):118:1–118:10, July 2012.

[Borghesan et al., 2010] G. Borghesan, G. Palli, and C. Melchiorri. Design of tendon-driven robotic fingers: Modeling and control issues. In *2010 IEEE International Conference on Robotics and Automation*, pages 793–798, May 2010.

[Bosworth et al., 2015] William Bosworth, Sangbae Kim, and Neville Hogan. The MIT super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2015, West Lafayette, IN, USA, October 18-20, 2015*, pages 1–8, 2015.

[Bosworth et al., 2016] Will Bosworth, Jonas Whitney, Sangbae Kim, and Neville Hogan. Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ. In *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 3582–3589, 2016.

[Bryson et al., 2016] Joshua T Bryson, Xin Jin, and Sunil K Agrawal. Configuration Robustness Analysis of the Optimal Design of Cable-Driven Manipulators. *Journal of Mechanisms and Robotics*, 8(6):061006, December 2016.

[Calì et al., 2012] Jacques Calì, Dan Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 3D-printing of non-assembly, articulated models. In *Proc. of ACM SIGGRAPH Asia '12*, 2012.

[Camarillo et al., 2008] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury. Mechanics modeling of tendon-driven continuum manipulators. *IEEE Transactions on Robotics*, 24(6):1262–1273, Dec 2008.

[Catalano et al., 2012] M. G. Catalano, G. Grioli, A. Serio, E. Farnioli, C. Piazza, and A. Bicchi. Adaptive synergies for a humanoid robot hand. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 7–14, Nov 2012.

[Ceylan et al., 2013] Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. Designing and fabricating mechanical automata from mocap sequences. In *Proc. of ACM SIGGRAPH Asia '13*, 2013.

[Chen et al., 2016] Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. Synthesis of filigrees for digital fabrication. *ACM Trans. Graph.*, 35(4):98:1–98:13, July 2016.

[Coros et al., 2013] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. In *Proc. of ACM SIGGRAPH '13*, 2013.

[Cutkosky and Kim, 2009] Mark R Cutkosky and Sangbae Kim. Design and fabrication of multi-material structures for bioinspired robots. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1894):1799–1813, 2009.

[Delp et al., 2007] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, Nov 2007.

[Dimitrov et al., 2008] D. Dimitrov, P.-B. Wieber, H.J. Ferreau, and M. Diehl. On the implementation of model predictive control for on-line walking pattern generation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2685–2690, May 2008.

[Fang et al., 2004] Shiqing Fang, D. Franitza, M. Torlo, F. Bekes, and M. Hiller. Motion control of a tendon-based parallel manipulator using optimal tension distribution. *IEEE/ASME Transactions on Mechatronics*, 9(3):561–568, Sept 2004.

[Frecker et al., 1997] M. I. Frecker, G. K. Ananthasuresh, S. Nishiwaki, N. Kikuchi, and S. Kota. Topological synthesis of compliant mechanisms using multi-criteria optimization. *Journal of Mechanical Design*, 2(119):238–245, Jun 1997.

[Garg et al., 2014] Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Trans. Graph.*, 33(4):66:1–66:12, July 2014.

[Gay et al., 2013] S. Gay, J. Santos-Victor, and A. Ijspeert. Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 194–201, Nov 2013.

[Gehring et al., 2013] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart. Control of dynamic gaits for a quadrupedal robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 3287–3292, May 2013.

[Geijtenbeek et al., 2013] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6), 2013.

[Gertz and Wright, 2003] E. Michael Gertz and Stephen J. Wright. Object-oriented software for quadratic programming. *ACM Trans. Math. Softw.*, 29(1):58–81, 2003.

[Grebenstein, 2014] Markus Grebenstein. *Approaching Human Performance*. Springer International Publishing, 2014.

[Guenter, 2007] Brian Guenter. Efficient symbolic differentiation for graphics applications. *ACM Trans. Graph.*, 26(3), July 2007.

[Hannaford et al., 2013] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White. Raven-ii: An open platform for surgical robotics research. *IEEE Transactions on Biomedical Engineering*, 60(4):954–959, April 2013.

[Hansen et al., 2003] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evol. Comput.*, 11(1):1–18, March 2003.

[Hecker et al., 2008] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. In *Proc. of ACM SIGGRAPH '08*, 2008.

[Hergel and Lefebvre, 2015] Jean Hergel and Sylvain Lefebvre. 3d fabrication of 2d mechanisms. In *Computer Graphics Forum*, volume 34, pages 229–238. Wiley Online Library, 2015.

[Hill, 1998] Rodney Hill. *The Mathematical Theory of Plasticity*. Clarendon Press, 1998.

[Honda Motor Co., Ltd., 1993] Honda Motor Co., Ltd. Honda humanoid prototype p1. `http://world.honda.com/ASIMO/history/p1_p2_p3/index.html`, 1993. Accessed on August 16, 2017.

[Honda Motor Co., Ltd., 1996] Honda Motor Co., Ltd. Honda humanoid prototype p2. `http://world.honda.com/ASIMO/history/p1_p2_p3/index.html`, 1996. Accessed on August 16, 2017.

[Honda Motor Co., Ltd., 1997] Honda Motor Co., Ltd. Honda humanoid prototype p3. `http://world.honda.com/ASIMO/history/p1_p2_p3/index.html`, 1997. Accessed on August 16, 2017.

[Honda Motor Co., Ltd., 2000] Honda Motor Co., Ltd. Honda humanoid prototype p4. `http://world.honda.com/ASIMO/history/p1_p2_p3/index.html`, 2000. Accessed on August 16, 2017.

[Honda Motor Co., Ltd., 2005] Honda Motor Co., Ltd. Asimo. `http://asimo.honda.com`, 2005. Accessed on August 16, 2017.

[Hopkins and Culpepper, 2010a] Jonathan B. Hopkins and Martin L. Culpepper. Synthesis of multi-degree of freedom, parallel flexure system concepts via freedom and constraint topology (fact) – part i: Principles. *Precision Engineering*, 34(2):259 – 270, 2010.

[Hopkins and Culpepper, 2010b] Jonathan B. Hopkins and Martin L. Culpepper. Synthesis of multi-degree of freedom, parallel flexure system concepts via freedom and constraint topology (fact). part ii: Practice. *Precision Engineering*, 34(2):271 – 278, 2010.

[Howell and Midha, 1994] L.L. Howell and A. Midha. A method for the design of compliant mechanisms with small-length flexural pivots. *ASME. J. Mech. Des.*, 116(1):280–290, 1994.

[Howell et al., 2013] Larry L. Howell, Spencer P. Magleby, and Brian M. Olsen. *Handbook of Compliant Mechanisms*. Wiley, 2013.

[Howell, 2001] Larry L. Howell. *Compliant Mechanisms*. Wiley-Interscience, 2001.

[Huang et al., 2013] Zhen Huang, Qinchuan Li, and Huafeng Ding. *Basics of Screw Theory*, pages 1–16. Springer Netherlands, Dordrecht, 2013.

[Hutter et al., 2012] M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, C. D. Remy, and R. Siegwart. StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *15th International Conference on Climbing and Walking Robot - CLAWAR 2012*, 2012.

[Iribe and Osuka, 2006] Masatsugu Iribe and Koichi Osuka. Analysis and stabilization of the passive walking robot via analogy with the phase locked loop circuits. In *2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, December 4-6, 2006*, pages 548–553, 2006.

[Ito et al., 2016] Takashi Ito, Marcin L. Pilat, Reiji Suzuki, and Takaya Arita. Population and evolutionary dynamics based on predator-prey relationships in a 3d physical simulation. *Artif. Life*, 22(2):226–240, May 2016.

[Kajita et al., 2003] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, and Kensuke Harada Kazuhito Yokoi. Biped walking pattern generation by using preview control of zero-moment point. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1620–1626, 2003.

[Kilian et al., 2017] Martin Kilian, Aron Monszpart, and Niloy J. Mitra. String actuated curved folded surfaces. *ACM Transactions on Graphics (to appear)*, 2017.

[Konsella et al., 2017] Riley Konsella, Frank Chiarulli, John Peterson, and John Rieffel. A baseline-realistic objective open-ended kinematics simulator for evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1113–1116, New York, NY, USA, 2017. ACM.

[Koo et al., 2014] Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. Creating works-like prototypes of mechanical objects. *ACM Transactions on Graphics (Special issue of SIGGRAPH Asia 2014)*, 2014.

[Kota and Ananthasuresh, 1995] Sridhar Kota and GK Ananthasuresh. Designing compliant mechanisms. *Mechanical Engineering-CIME*, 117(11):93–97, 1995.

[Kota et al., 2001] Sridhar Kota, Jinyong Joo, Zhe Li, Steven M. Rodgers, and Jeff Sniegowski. Design of compliant mechanisms: Applications to mems. *Analog Integr. Circuits Signal Process.*, 29(1-2):7–15, October 2001.

[Krantz and Parks, 2002] S.G. Krantz and H.R. Parks. *The Implicit Function Theorem: History, Theory, and Applications*. The Implicit Function Theorem: History, Theory, and Applications. Birkhäuser, 2002.

[KUKA, 1974] KUKA. Famulus. `https://www.kuka.com/en-de/about-kuka/history`, 1974. Accessed on August 16, 2017.

[Landau et al., 1986] L.D. Landau, L.P. Pitaevskii, A.M. Kosevich, and E.M. Lifshitz. *Theory of Elasticity*. Elsevier, 1986.

[Lau et al., 2011] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3D furniture models to fabricatable parts and connectors. In *Proc. of ACM SIGGRAPH '11*, 2011.

[Leger, 1999] Patrick (Chris) Leger. *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1999.

[Levenberg, 1944] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[Limaye et al., 2012] Padmanabh Limaye, G. Ramu, Sindhuja Pamulapati, and G.K. Ananthasuresh. A compliant mechanism kit with flexible beams and connectors along with analysis and optimal synthesis procedures. *Mechanism and Machine Theory*, 49:21 – 39, 2012. https://www.youtube.com/watch?v=HuOEG5FqTAE.

[Lin et al., 2016] Minmin Lin, Tianjia Shao, Youyi Zheng, Niloy J. Mitra, and Kun Zhou. Recovering functional mechanical assemblies from raw scans. *Transactions on Visualization and Comptuer Graphics*, 2016.

[Lipson and Pollack, 2000] Hod Lipson and Jordan B. Pollack. Towards continuously reconfigurable self-designing robotics. In *ICRA*, pages 1761–1766. IEEE, 2000.

[Lu et al., 2014] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph.*, 33(4):97:1–97:10, July 2014.

[Ma et al., 2013] R. R. Ma, L. U. Odhner, and A. M. Dollar. A modular, open-source 3d printed underactuated hand. In *2013 IEEE International Conference on Robotics and Automation*, pages 2737–2743, May 2013.

[Makino, 1982] Hiroshi Makino. Assembly robot, July 27 1982. US Patent 4,341,502.

[Marquardt, 1963] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1 Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science. Genghis, a six legged autonomous walking robot. `http://dspace.mit.edu/handle/1721.1/14531`, 1989. Accessed on August 16, 2017.

[Massachusetts Institute of Technology, 1995] Massachusetts Institute of Technology. Robotuna. `http://tech.mit.edu/V115/N49/robotuna.49n.html`, 1995. Accessed on August 16, 2017.

[Mastalli et al., 2015] Carlos Mastalli, Alexander Winkler, Ioannis Havoutis, Darwin G. Caldwell, and Claudio Semini. On-line and on-board planning and perception for quadrupedal locomotion. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*. IEEE, May 2015.

[Megaro et al., 2014] Vittorio Megaro, Bernhard Thomaszewski, Damien Gauge, Eitan Grinspun, Stelian Coros, and Markus Gross. Chacra: An interactive design system for rapid character crafting. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, pages 123–130, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association.

[Mehta and Rus, 2014] Ankur M. Mehta and Daniela Rus. An end-to-end system for designing mechanical structures for print-and-fold robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014.

[Mehta et al., 2014] Ankur M. Mehta, Joseph DelPreto, Benjamin Shaya, and Daniela Rus. Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, Sep 2014.

[Methenitis et al., 2015] Georgios Methenitis, Daniel Hennes, Dario Izzo, and Arnoud Visser. Novelty search for soft robotic space exploration. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 193–200, New York, NY, USA, 2015. ACM.

[Miguel et al., 2016] Eder Miguel, Mathias Lepoutre, and Bernd Bickel. Computational design of stable planar-rod structures. *ACM Trans. Graph.*, 35(4):86:1–86:11, July 2016.

[Mitra et al., 2010] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating how mechanical assemblies work. *ACM Trans. Graph.*, 29(4):58:1–58:12, July 2010.

[Mitsui et al., 2013] K. Mitsui, R. Ozawa, and T. Kou. An under-actuated robotic hand for multiple grasps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5475–5480, Nov 2013.

[Mizuuchi et al., 2002] I. Mizuuchi, R. Tajima, T. Yoshikai, D. Sato, K. Nagashima, M. Inaba, Y. Kuniyoshi, and H. Inoue. The design and control of the flexible spine of a fully tendon-driven humanoid "kenta". In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2527–2532 vol.3, 2002.

[Mordatch et al., 2012] Igor Mordatch, Emanuel Todorov, and Zoran Popović.

Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012.

[Mori and Igarashi, 2007] Yuki Mori and Takeo Igarashi. Plushie: An interactive design system for plush toys. In *Proc. of ACM SIGGRAPH '07*, 2007.

[Musialski et al., 2015] Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. Reduced-order shape optimization using offset surfaces. *ACM Transactions on Graphics (ACM SIGGRAPH 2015)*, 34(4):to appear–9, August 2015.

[Neuhaus et al., 2011] Peter Neuhaus, Jerry Pratt, and Matthew Johnson. Comprehensive summary of the institute for human and machine cognition's experience with littledog. *International Journal Of Robotics Research*, 30(2):216–235, February 2011.

[Nocedal and Wright, 2006] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

[Nocedal, 1980] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[ODE, 2007] ODE. Open dynamics engine, http://www.ode.org/, 2007.

[Ozawa et al., 2009] R. Ozawa, K. Hashirii, and H. Kobayashi. Design and control of underactuated tendon-driven mechanisms. In *2009 IEEE International Conference on Robotics and Automation*, pages 1522–1527, May 2009.

[Pardo et al., 2016] Diego Pardo, Lukas Möller, Michael Neunert, Alexander W. Winkler, and Jonas Buchli. Evaluating direct transcription and nonlinear optimization methods for robot motion planning. *IEEE Robotics and Automation Letters*, 1(2):946–953, 2016.

[Park and Kim, 2015] Hae-Won Park and Sangbae Kim. Quadrupedal galloping control for a wide range of speed via vertical impulse scaling. *Bioinspiration & Biomimetics*, 10(2):025003, 2015.

[Park et al., 2014] Hae-Won Park, Meng Yee Chuah, and Sangbae Kim. Quadruped bounding control with variable duty cycle via vertical impulse scaling. In *IROS*, pages 3245–3252. IEEE, 2014.

[Park et al., 2015] Hae-Won Park, Patrick M. Wensing, and Sangbae Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In Lydia E. Kavraki, David Hsu, and Jonas Buchli, editors, *Robotics: Science and Systems*, 2015.

[Park et al., 2016] Chonhyon Park, Jae Sung Park, Steve Tonneau, Nicolas Mansard, Franck Multon, Julien Pettré, and Dinesh Manocha. Dynamically

balanced and plausible trajectory planning for human-like characters. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, pages 39–48, New York, NY, USA, 2016. ACM.

[Peng et al., 2015] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Trans. Graph.*, 34(4):80:1–80:11, July 2015.

[Peng et al., 2016] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, 35(4), 2016.

[Peng et al., 2017a] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36(4), 2017.

[Peng et al., 2017b] Xue Bin Peng, Michiel van de Panne, and KangKang Yin. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2017.

[Pérez et al., 2015] Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. Design and fabrication of flexible rod meshes. *ACM Trans. Graph.*, 34(4):138:1–138:12, July 2015.

[Potkonjak et al., 2011] Veljko Potkonjak, Bratislav Svetozarevic, Kosta Jovanovic, and Owen Holland. The puller-follower control of compliant and noncompliant antagonistic tendon drives in robotic systems. *International Journal of Advanced Robotic Systems*, 8(5):69, 2011.

[Pratt et al., 2006] Jerry E. Pratt, John Carff, Sergey V. Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *Humanoids*, pages 200–207. IEEE, 2006.

[Prévost et al., 2013] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. In *Proc. of ACM SIGGRAPH '13*, 2013.

[Pucheta and Cardona, 2010] Martín A. Pucheta and Alberto Cardona. Design of bistable compliant mechanisms using precision–position and rigid-body replacement methods. *Mechanism and Machine Theory*, 45(2):304 – 326, 2010.

[Rob, 2016] Roboy. `http://roboy.org`, 2016. Accessed: 2016-12-28.

[Rooks, 2006] Brian Rooks. The harmonious robot. *Industrial Robot: An International Journal*, 33(2):125–130, 2006.

[Roy et al., 2013] Nicholas Roy, Paul Newman, and Siddhartha Srinivasa. *Tendon-Driven Variable Impedance Control Using Reinforcement Learning*, pages 504–. MIT Press, 2013.

[Rucker and III, 2011] D. C. Rucker and R. J. Webster III. Statics and dynamics of continuum robots with general tendon routing and external loading. *IEEE Transactions on Robotics*, 27(6):1033–1044, Dec 2011.

[Rutishauser et al., 2008] S. Rutishauser, A. Sproewitz, L. Righetti, and A.J. Ijspeert. Passive compliant quadruped robot using central pattern generators for locomotion control. In *2008 IEEE International Conference on Biomedical Robotics and Biomechatronics*, pages 710–715, 2008.

[Sachdeva et al., 2015] Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K. Pai. Biomechanical simulation and control of hands and tendinous systems. *ACM Trans. Graph.*, 34(4):42:1–42:10, July 2015.

[Samuel, 1959] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, July 1959.

[Sawada and Ozawa, 2012] D. Sawada and R. Ozawa. Joint control of tendon-driven mechanisms with branching tendons. In *2012 IEEE International Conference on Robotics and Automation*, pages 1501–1507, May 2012.

[Schüller et al., 2016] Christian Schüller, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. Computational thermoforming. *ACM Trans. Graph.*, 35(4):43:1–43:9, July 2016.

[Semini et al., 2011] C Semini, N G Tsagarakis, E Guglielmino, M Focchi, F Cannella, and D G Caldwell. Design of hyq – a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[Seok et al., 2013] Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey Lang, and Sangbae Kim. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. In *ICRA*, pages 3307–3312. IEEE, 2013.

[Sims, 1994] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 15–22, New York, NY, USA, 1994. ACM.

[Skouras et al., 2013] Mélina Skouras, Bernhard Thomaszewski, Stelian Coros,

Bernd Bickel, and Markus Gross. Computational design of actuated deformable characters. In *Proc. of ACM SIGGRAPH '13*, 2013.

[Smith, 2000] Stuart T. Smith. *Flexure: Elements of Elastic Mechanuisms*. CRC Press, 2000.

[Sony Corporation, 1999] Sony Corporation. Aibo. `http://www.sony-aibo.com/`, 1999. Accessed on August 16, 2017.

[Sony Corporation, 2003] Sony Corporation. Qrio. `https://www.sony.net/SonyInfo/News/Press_Archive/200312/03-060E/`, 2003. Accessed on August 16, 2017.

[Spa, 2016] Sparky. `http://rasc.usc.edu/sparky.html`, 2016. Accessed: 2016-12-28.

[Sproewitz et al., 2009] A. Sproewitz, M. Fremerey, K. Karakasiliotis, S. Rutishauser, L. Righetti, and A.J. Ijspeert. Compliant leg design for a quadruped robot. In *Proceedings of Dynamic Walking 2009*, Vancouver, Canada, 2009.

[Spröwitz et al., 2013] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950, 2013.

[Spröwitz et al., 2014] A. T. Spröwitz, M. Ajallooeian, A. Tuleu, and A. J. Ijspeert. Kinematic primitives for walking and trotting gaits of a quadruped robot with compliant legs. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 8, 2014.

[Spröwitz et al., 2014] A. Spröwitz, R. Moeckel, M. Vespignani, S. Bonardi, and A.J. Ijspeert. Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7):1016 – 1033, 2014. Reconfigurable Modular Robotics.

[Stava et al., 2012] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: improving structural strength of 3d printable objects. In *Proc. of ACM SIGGRAPH '12*, 2012.

[Stulp et al., 2010] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal. Reinforcement learning of full-body humanoid motor skills. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 405–410, December 2010.

[Sueda et al., 2008] Shinjiro Sueda, Andrew Kaufman, and Dinesh K. Pai. Musculotendon simulation for hand animation. *ACM Trans. Graph.*, 27(3):83:1–83:8, August 2008.

[Sueda et al., 2011] Shinjiro Sueda, Garrett L. Jones, David I. W. Levin, and Dinesh K. Pai. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph.*, 30(4):39:1–39:10, July 2011.

[Sun et al., 2015] X. Sun, S. M. Felton, R. Niiyama, R. J. Wood, and S. Kim. Self-folding and self-actuating robots: A pneumatic approach. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3160–3165, May 2015.

[Thomaszewski et al., 2014] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design of linkage-based characters. In *Proc. of ACM SIGGRAPH '14*, 2014.

[Tuleu et al., 2011] Alexandre Tuleu, Mostafa Ajallooeian, Alexander Sproewitz, P. Loepelmann, and Auke Ijspeert. Trot gait locomotion of a cat sized quadruped robot. 2011. Accepted for poster and oral session at Workshop on bio-inspired robotics, Nantes (France), 6-8th of April 2011.

[Umetani and Schmidt, 2013] Nobuyuki Umetani and Ryan Schmidt. Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, pages 5:1–5:4, New York, NY, USA, 2013. ACM.

[Umetani et al., 2012] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. In *Proc. of ACM SIGGRAPH '12*, 2012.

[Umetani et al., 2014] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 33(4):65:1–65:10, July 2014.

[Umetani et al., 2016] Nobuyuki Umetani, Athina Panotopoulou, Ryan Schmidt, and Emily Whiting. Printone: Interactive resonance simulation for free-form print-wind instrument design. *ACM Trans. Graph.*, 35(6):184:1–184:14, November 2016.

[van den Broek et al., 2008] Bart van den Broek, Wim Wiegerinck, and Bert Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. 32:95–122, 05 2008.

[Vargas and González-Hernández, 2013] A. M. Vargas and H. G. González-Hernández. Dynamic passive biped robot simulation based on virtual gravity using matlab. In *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*, pages 207–211, March 2013.

[Vuga et al., 2013] R. Vuga, M. Ogrinc, A. Gams, T. Petrič, N. Sugimoto, A. Ude, and J. Morimoto. Motion capture and reinforcement learning of dynamically stable humanoid movement primitives. In *2013 IEEE International Conference on Robotics and Automation*, pages 5284–5290, May 2013.

[Vukobratović and Borovac, 2004] Miomir Vukobratović and Branislav Borovac. Zero-moment point — thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004.

[Wampler and Popović, 2009a] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. *ACM Trans. Graph.*, 28(3):60:1–60:8, July 2009.

[Wampler and Popović, 2009b] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 60:1–60:8, New York, NY, USA, 2009. ACM.

[Wampler et al., 2014] Kevin Wampler, Zoran Popović, and Jovan Popović. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Trans. Graph.*, 33(4):49:1–49:11, July 2014.

[Wang and Chen, 2009] Michael Yu Wang and Shikui Chen. Compliant mechanism optimization: Analysis and design with intrinsic characteristic stiffness. *Mechanics Based Design of Structures and Machines*, 37(2):183–200, 2009.

[Waseda University, 1972] Waseda University. Wabot-1. `http://www.humanoid.waseda.ac.jp/booklet/kato_2-j.html`, 1972. Accessed on August 16, 2017.

[Whitney et al., 2014] J. P. Whitney, M. F. Glisson, E. L. Brockmeyer, and J. K. Hodgins. A low-friction passive fluid transmission and fluid-tendon soft actuator. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2801–2808, Sept 2014.

[Winkler et al., 2015] Alexander W. Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin Caldwell, and Claudio Semini. Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. In *IEEE International Conference on Robotics and Automation*, pages 5148–5154, 2015.

[Winkler et al., 2017] Alexander W. Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, and Jonas Buchli. Online walking motion and foothold optimization for quadruped locomotion. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 5308–5313, 2017.

[Witkin and Kass, 1988] Andrew Witkin and Michael Kass. Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and In-*

teractive Techniques, SIGGRAPH '88, pages 159–168, New York, NY, USA, 1988. ACM.

[Zang et al., 2017] Xizhe Zang, Xinyu Liu, Yongsheng Gao, Yixiang Liu, and Zhenkun Lin. Development of a passive dynamic walking robot based on mechanical structural parameters optimization. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1465–1470, July 2017.

[Zehnder et al., 2016] Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. Designing structurally-sound ornamental curve networks. *ACM Trans. Graph.*, 35(4):99:1–99:10, July 2016.

[Zhou et al., 2013] Qingnan Zhou, Julian Panetta, and Denis Zorin. Worst-case structural analysis. In *Proc. of ACM SIGGRAPH '13*, 2013.

[Zhu et al., 2012] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. Motion-guided mechanical toy modeling. In *Proc. of ACM SIGGRAPH Asia '12*, 2012.

[Zimmermann et al., 2015] Daniel Zimmermann, Stelian Coros, Yuting Ye, Robert W. Sumner, and Markus Gross. Hierarchical planning and control for complex motor tasks. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, pages 73–81, New York, NY, USA, 2015. ACM.