Methods in Computational Design and Optimization

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters

<table>
<thead>
<tr>
<th>Citable link</th>
<th><a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:39945341">http://nrs.harvard.edu/urn-3:HUL.InstRepos:39945341</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Terms of Use</td>
<td>This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA">http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA</a></td>
</tr>
</tbody>
</table>
The undersigned, appointed by the Harvard John A. Paulson School of Engineering and Applied Sciences have examined a dissertation entitled:

“Methods in Computational Design and Optimization”

presented by: Gaurav Bharaj

candidate for the degree of Doctor of Philosophy and here by certify that it is worthy of acceptance.

Typed name: Professor H. Pfister

Typed name: Professor S. Gortler

Typed name: Professor R. Nagpal

Typed name: Professor T. Zickler

November 3, 2017
Methods in Computational Design and Optimization

A dissertation presented
by
Gaurav Bharaj
to
The School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts

November, 2017
Computational design allows creation of functional objects by using accurate computational models and 3D printing technologies. As a result, not only is customization of functional objects possible; but it also eliminates the need for expert domain knowledge required to intuitively model the functionality. Shape representation and parameterization, combined with numerical simulation, are used to create an accurate model of the real world response of an object. Then, to achieve the desired functionality, mathematical optimization is used to efficiently search over parameterized space for possible variations and physical energy spaces of an object’s design.

By using a structured method to search over these parametric spaces, i.e., using nonlinear and often constraint multi-objective optimization methods, we guarantee the most optimal design of the object. This optimized design is 3D print ready and functional. My work has focused on developing optimization schemes for computational design problems, namely – walking automata, bistable structures, and contact sound spectrum design. Such problems are variations of physical energy minimization of designs, and I propose ways to optimize them.

For walking automata, starting with an initial unstable non-walking linkage configuration, I develop a sampling based optimization scheme to search the highly nonlinear space of possibly stable walking automata. Further, improvements in the efficiency of the optimization strategy are gained by learning a space of valid linkage configurations. Linkage chains can also be used for creating bistable structures. I quantify static stability of the linkage structure through the geometry of the physical energy and develop a non-linear
constraint-optimization scheme to guarantee second-order stability for bistable planar structures. Finally, I study the sound spectrum design of metallic 2D and 3D objects. Here, I develop a hybrid local-global optimization scheme which gives precise control over the frequencies and corresponding amplitudes of objects.

With ubiquitous 3D printing and fast computational models, we can create personalized functional objects. My research helps realize this goal, leading to a world where objects are created for customized functions and increased efficiency. A systematic study of the aforesaid problems provides general recipes for problems of physical energy minimization in computational design.
## Contents

1 Introduction 1  
1.1 Computational design 1  
1.2 Contributions and Thesis Overview 5  

2 Related Works 10  
2.1 Computational design and optimization 10  
2.2 Animation, Control, and Mechanisms 12  
2.3 Acoustics and Contact Sounds 16  

3 Walking Automata 20  
3.1 Introduction 21  
3.2 Overview 22  
3.3 Quantifying Walking 23  
3.4 Optimization 26  
3.5 Experiments 32  
3.6 Limitations And Future Work 37  

4 Bistable planar structures 40  
4.1 Introduction 41  
4.2 Overview 43
7.7 Derivatives of the General Eigenvalues ........................................ 129
7.8 Derivative of Generalized Eigenvectors ........................................ 131
7.9 Free Vibration with Stands ............................................................ 133
# List of Figures

1.1 Computational Design Methodology .............................................. 2  
1.2 Common themes among our design problems ................................. 7  

3.1 Two walking automata designed with our system ......................... 20  
3.2 Walking Automata Overview .................................................. 21  
3.3 Kinematics of the mechanical toys ......................................... 24  
3.4 Gaussin Mixture Models (GMMs) Sampling .................................. 31  
3.5 Trial runs with CMA vs. CMA-GMM ......................................... 33  
3.6 The two automata for which we fabricate results ....................... 35  
3.7 Various automata produced by our system .................................. 38  

4.1 Bistable structures generated by our system ............................... 41  
4.2 Bistable structures: Overview ................................................. 43  
4.3 Springy Linkage ................................................................. 45  
4.4 The input curves ............................................................... 48  
4.5 Two morphable forms .......................................................... 55  
4.6 Spring assembly ............................................................... 57  
4.7 Spring calibration ............................................................... 59  
4.8 Higher Complexity .............................................................. 59  
4.9 Functional Metamorph .......................................................... 60
4.10 Posable Hand/Gripper .................................................. 62
5.1 Customized Metallophone instrument ............................. 65
5.2 Metallophone Contact Sounds: Overview ....................... 67
5.3 Latin Complement Sampling ........................................... 76
5.4 Effects of perforation .................................................... 81
5.5 Stand creation process .................................................. 83
5.6 1D function example comparing LCS to SA and CMA-ES ....... 84
5.7 A comparison of various optimization methods .................... 92
5.8 Scaling ................................................................. 93
5.9 The Zoolophone ....................................................... 94
5.10 Tea cups example ..................................................... 95
5.11 Cityscape ............................................................. 96
5.12 Dense Frequency Spectrum ........................................... 97
5.13 Dense spectrum Optimization ....................................... 98
7.1 Sample linkage configuration ......................................... 121
7.2 Higher order shape functions ......................................... 126
7.3 Material calibration .................................................... 128
7.4 Sound measurement booth ............................................ 129
List of Tables

3.1 Statistics for all examples ............................................. 34
4.1 Metamorphs: Space and time complexity ............................... 61
5.1 Comparing LCS to SA and CMA-ES ...................................... 85
5.2 The Zoolophone ................................................................. 89
7.1 Material properties and fabrication methods ............................ 127
To my parents.
Acknowledgments

This thesis is possible only due to the constant support and feedback of my adviser Prof. Hanspeter Pfister. Over the past five years, your mentoring has helped me grow both personally and professionally. With great pleasure, sincerity and gratitude, thank you Prof. Pfister.

Thank you Prof(s). James Tompkin and David Levine for your amazing support and patience over multiple projects. It not only helped me grow as a researcher, but also taught me how to evaluate my ideas and research. A big thank you to my thesis committee Prof(s). Steven Gortler, Radhika Nagpal, and Todd Zickler for their constructive feedback and discussions over the years.

The projects in this thesis would not have been possible without my amazing collaborators especially Prof(s). Bernd Bickel, Stelian Coros, Wojciech Matusik, Bernhard Thomaszewski, Etienne Vouga, and Changxi Zheng, and Dr. Danny Kaufman. It has been a pleasure working with everyone. I’d also like to take this opportunity to thank Prof. Christian Theobalt, who introduced me to the world of research!

I am also in-debt of my lab mates and colleagues over the years, including Emma Alexander, Johanna Beyer, Ioannis Gkioulekas, Daniel Haehn, Adi Peleg, and Hendrik Strobelt for providing a great work atmosphere, thank you. And, SEAS staff including Dr. John Girash, Michaela Kapp and Cate Lagueux for helping out with my many queries.

Thank you to my friends in Boston, and around the world for keeping me sane during the times of stress, and helping me appreciate life as it is. Thank you to my mother Dr. Meenakshi Bharaj, father Er. Satish K. Bharaj, and my sister Sheerin, for their unconditional love and support throughout my life.
Citations of Published Work

The work presented in Chapter 3 appeared in the following paper and was done during an internship at Disney Research:

The research presented in Chapter 4 was done during an internship at Adobe Research. It appears in the following paper:

Significant portions of Chapter 5 have appeared in the following paper:
Chapter 1

Introduction

1.1 Computational design

Computational design is a modern approach to additive manufacturing, where functional objects are created using accurate computational models and 3D printing technology. This leads the way to personalized functional objects and reduced need for expert knowledge for an intuitive understanding of functional behavior. Computational models are used to understand and simulate the functionality virtually. Then, by creating millions of virtual variations of these representations and search, a methodology is developed which can model an array of desired functionalities. This saves not only production costs, but also the man-hours needed to first fabricate objects and then understand their functional behavior.

A concrete example is car manufacture. To create a fuel efficient car with reduced air-drag, we create several variations of the car’s shape, and measure air drag on the fabricated physical models. Creating these variations by hand, or even fabricating them is practically impossible; imagine creating hundreds of cars with differently bent fenders! Even if automatic machines and assemblies are available, fabrication of a radically new
design would be possible only through 3D printing technologies, as automatic machines and assemblies are programmed for mass-production of a fixed design.

Computational design provides a comprehensive solution (Fig 1.1). We represent a car’s body virtually (shape representation), and create several variations in the shape (property parametrization). To model the air-drag on the car, we use fluid dynamics simulation (numerical simulation). Finally, we search for the shape with the least amount of drag from among the variations. There can be multiple goals; for example, the car’s shape not only has to be aerodynamic, but also provide enough trunk space; hence the need for multi-objective search.

Solving these complex multi-objective goals is often too complex for manual execution and we need better search strategies to go through millions of variations. Here, the aim is to search for the most optimal design while satisfying the various design goals. These search methods have to be fast, accurate, and reliable. In this chapter, I emphasize the ingredients needed by computational design to create multi-functional objects, and introduce how optimization plays an integral role in this process.
**Shape representation:** Computer-aided design (CAD) Farin [2014] and geometry processing communities Botsch et al. [2010] have made great strides in recent years to create accurate virtual representations of everyday objects. While the ultimate goals of these communities might vary, the underlying ideas and mathematical representations have been quite consistent among them. This has led to parametric shape representation models that can not only represent the shape of any real object, but also shape deformation methods needed for creating variations in the shape. With 3D printers, we can then physically print such shapes accurately and consistently.

Thus, shape representation plays a central role in creating the initial design, the properties of which can then be varied to create a functional object. For computational design needs, the challenge is usually the choice of shape representation model to be used, such that the model is simple enough, yet pronounced enough for us to create the variations needed to come up with a functional object.

**Property parametrization:** A functional object needs to possess a desired property. For example, a simple lever is useful due to its long, cylindrical shape, which enables it to acts as a fulcrum to pull on something. Here, the object’s shape facilitates functionality. Similarly, objects have geometric, material, optical, and acoustic properties that make them useful. Varying these properties often relies on the object’s shape and its deformation properties. By parameterizing the shape (length) of the cylindrical bar in the aforesaid example, we can change its effectiveness as a lever. Thus, this process properties parametrizes properties that can that can improve the desired function of an object. The challenge here is to come up with an accurate parameterization that can help achieve the functionalities optimally.
**Numerical Simulation:** We now have the initial shape and its parameterization for a design problem. Next, we need an accurate measure the object’s real world behavior and interaction (by simulating the physical energy change). Borrowing ideas from continuum mechanics, physics-based animation and robotics, we design (and choose) numerical simulation techniques, which model the real-world behavior most accurately, through kinetic and potential energies of the object. This then gives us a measure of efficiency for the desired functionalities. The challenges here are (1) choosing the correct simulation model, (2) calibration to ensure accuracy of the simulated models are accurate, and (3) validation, where we test fabricated objects on how faithfully they follow the simulation model. For example, the finite element method (FEM) is used for virtually understanding the stress strain behavior of material objects under varied loads. However, the choice of specific shape-functions (Babuška and Suri [1994]) for improving the accuracy of the FEM simulation is a research goal we pursue for the design problem.

### 1.1.1 Role of optimization in computational design

“People optimize. Investors seek to create portfolios that avoid excessive risk while achieving a high rate of return. Manufacturers aim for maximum efficiency in the design and operation of their production processes. Engineers adjust parameters to optimize the performance of their designs.

Nature optimizes. Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time”

– Wright and Nocedal [1999]

As this quote above suggests, every parameterizable problem can be thought of as a minimization problem. This is especially true of computational design problems. Most
such problems deal with creating functionality by changing the shape, and therefore the physical properties (e.g. moment-of-inertia) of the system. Hence, it is quite natural that we end up minimizing the physical energy functions of the designs. Bickel et al. [2010] and Skouras et al. [2013b] use FEM-based energy models to measure and fabricate shape deformations. Umetani et al. [2014] design model airplanes where air-drag is simulated. Chen et al. [2017] combine rigid body dynamics and contact simulation, along with deformable stresses and strains, to model jumping robots. These inter alia, are all examples of physical energy minimizations for various design problems. In the problems listed above, optimization is the quintessential part of the computational design process that is used to search for the best design with the desired energy profile. In this thesis, I concentrate specifically on eigenvalue and stability-based energy minimization problems, as explained in subsequent chapters.

1.2 Contributions and Thesis Overview

My work has focused on developing optimization schemes for computational design problems – walking automata, bistable structures, and contact sound spectrum design – that place the user as an initial designer with a goal, and pose the computer as a refiner that brings this design into compliance with real-world constraints. I design optimization schemes that search for optimal solutions through hybrid local/global and often nonlinear optimization strategies. By preprocessing feasible sub-problems and pattern-analysis, I achieve speed-ups for optimization convergence.

Walking Automata: Inspired by the Strandbeest Jansen [2015] a linkage-based walking automata, the goal of the project was to allow novice users to create novel walking automata (Fig 3.1 and Bharaj et al. [2015a]). The user specifies an initial configuration of
the linkage-based legs. These leg configurations are complex, and most initial user designs lead to failure, where the automata falls over. My computational design approach allows the computer to refine the design, by testing variations of the initial design and simulating their performance. As the simulation is physically based (rigid body simulation), this produces an automata which walks stably in the real world once fabricated. To accomplish this, the algorithm learns a function space of valid linkage configurations, which the optimizer utilizes to improve the computation time and cost.

**Bistable Structures:** Like automata, planar chain linkages can be used for creating bistable structures, where extreme deformation leads to a drastic, yet stable form change for a given structure (Fig 4.1 and Bharaj et al. [2017]). Inspired by the muscles surrounding the rigid-bones in animals, I use springs as a means of adding stability to the bistable structures. Here, several optimized springs together lead to a configuration which is bistable. Much like the previous project, a design-by-user refine-by-optimization scheme is adopted, where the user creates the initial structures. Then, an iterative optimization performs a two-fold operation to create these bistable structures. First, the algorithm calculates the deformation modes (using modal/eigenvalue analysis) for both forms of the bistable structure and determines where to add a spring. Second, the algorithm employs a constrained nonlinear optimization method, to optimize the physical energy (using rigid-body and spring energy simulation) of the structure, such that, both the structures are second-order stable (physical) energy configurations, simultaneously. These steps are repeated iteratively until the stability criteria is met.

**Contact Sounds:** The final project is on sound spectrum design, which, similar to the Bistable Structures’ project, uses eigenvalue analysis of the physical energy, but models the contact sound spectrum and optimizes frequencies and amplitudes of 2D and 3D objects
Parameterizing the deformation of the shape of an object helps control its sound spectrum (frequencies and amplitudes). Almost all such shapes, e.g., a glockenspiel, are cuboidal, because the task of tuning other shapes is too complex. Beginning with user-specified shape and target sound spectrum, my computational design process refines the shape such that, when the fabricated object is struck with a mallet, it vibrates to produce the desired frequencies. Further, this scheme allows us control over the frequencies and amplitudes of harmonic and even non-harmonic overtones. I demonstrate this control over the spectrum by creating metallophone keys that produce chords when struck – not one note, but three!

Figure 1.2: Common themes among the design problems (red) in this thesis, with respect to physical energy analysis (blue) and optimization methodologies (green).

A generic methodology develops, where energy minimization functions define the physical energy of the system. Shape deformation and other physical properties parametrize these energy functions. Novel optimization strategies minimize these energies. These strategies have proven to be generic and can be applied for solving problems in different domains. For example, I use eigenvalue analysis and continuum mechanics for studying...
both sound spectrum and stable spring configuration systems. Similarly, I use rigid body dynamics to model walking automata and bistable structures, as shown in Figure 1.2. Insights from these methods are useful for solving problems in the domains of computational design, graphics, and robotics.

The major contributions of this thesis are as follows:

- Quantification and modeling of stable walking for mechanical automata. An optimization approach for creating walkable automata by exploring the complex design space of these structures. A Bayesian learning strategy for learning a function of valid linkage configurations, and using the same for faster optimization convergence (Chapter 3).

- Quantification and modeling of stability of spring attached linkage-based bistable planar structures. An optimization formulation which models the nonlinear physical energy of the bistable structures, and provides second-order stability guarantee for the two forms of the bistable structure. Analytical derivatives of the system energy and its hessian with respect to physical parameters of the springs (Chapter 4).

- An algorithm to optimize the entire sound spectrum of a 3D object via shape variation. A new multi-objective formulation to optimize the frequency and amplitude of a linearly elastic object. A hybrid search method called Latin Complement Sampling tailored to non-convex constrained optimization, which provides probabilistic bounds on design landscape coverage. Analytical derivatives for sound frequency and sound amplitude with respect to the proposed design parameters. A method for creating a stand for the fabricated object, such that its sound quality is maximized (Chapter 5).

Chapter 2 presents related works. Chapters 3, 4, and 5 consist of detailed discussion on problem formulation, optimization strategies, and results for walking automata, bistable
structures, and sound spectrum design, respectively. Chapter 6 presents conclusions and
discussion of future research directions. Chapter 7 (Appendix) provides further details on
modeling the physical – rigid body dynamics, and continuum-based energy systems and
analytical gradient of system energies for various parameterizations.
Chapter 2

Related Works

2.1 Computational design and optimization

The works I describe in this section follow the computational design methodology defined in Figure 1.1. For the purposes of this thesis and the problems dealt with therein, I divide the design problems into two categories (1) Geometric and kinematic design, (2) Physical energy-based design and optimization.

**Geometric and kinematic design:** As the name suggests, methods defined in this category use the geometric shape and kinematics (motion) to define the numerical simulation of the design problem. Here, the physical energy of the system is not modeled, but the shape and motion are defined through complex mathematical functions and optimization. For example, in linkage design problems, Coros et al. [2013a] define the movement of the linkage by a constraint optimization of the connections; however, they do not model or optimize for the physical energy of the linkage structure. Some methods in this category aim to bring virtual characters to the real world. It is now possible to create 3D printable representations of virtual linkage-based characters with joints Cali et al. [2012],
and mechanical toys capable of interesting non-walking motions Ceylan et al. [2013], Thomaszewski et al. [2014], Yu et al. [2017]. Origami inspired geometric design also falls in this categories, where rigid origami O’Rourke [1998], Dudte et al. [2016], and pop-design Li et al. [2010] are used to create kinematics of a design. Mitani and Suzuki [2004] show the use of strip patterns to assemble 3D models. Other researchers like Skouras et al. [2015], create complex structures by creating user interfaces for interlocking elements by understanding the geometry of the atomic-elements, while Hildebrand et al. [2012] create puzzles by using planar slits abstracted from 3D shapes. Xin et al. [2011] create 3D puzzles by automatically disintegrating 3D models into interlocking elements.

**Physical energy-based design** In these works, the aim is to optimize the material and shape properties, where cost functions model the efficiency of functionality through *physically-based* numerical simulation. Such works use principles from continuum mechanics (finite element method, boundary element methods, etc), fluid mechanics, and physics-based wave propagation (optics) to model the material deformation and response and rigid body dynamics to model the energy dynamics and ground contacts. Various problems that have been worked on include:

Appearance-based material distribution for subsurface scattering Hašan et al. [2010], Dong et al. [2010], caustics Papas et al. [2011] or reflectivity Matusik et al. [2009], Weyrich et al. [2009], material and physical Bickel et al. [2010] behavior of fabricable shapes. Prévost et al. [2013] and Musialski et al. [2015] optimize shape via material carving to control the moment-of-inertia property of the rigid shapes, and as a result control static stability, given that the shape rests on a surface or on water. Li et al. [2016] optimize for the sound spectrum through voxel filters that act much like selective damping filters, while Martin et al. [2015] use fluid-dynamics principles to model the uplift and drag for 3D flying designs. Creating controllable deformation has been an active area of research among
various interdisciplinary areas. Skouras et al. [2013b], Schumacher et al. [2015], Bern et al. [2017] and Pérez et al. [2017] optimize for shape deformation properties to create articulate characters and shapes. Chen et al. [2017] created jumping robots with precise upright landing capabilities, by modeling the dynamics of robots. Finally, Chen et al. [2013] abstract previous methods by goal, parameter reduction scheme, optimization method, and simulation algorithm and provide a structured way to define computational design problems.

2.2 Animation, Control, and Mechanisms

Below I discuss related works in several interdisciplinary areas that overlap with my works as discussed in Chapters 3 and 4, including works from computer graphics and animation, robotics, and continuum mechanics communities.

Character Animation: One major inspiration is the pioneering work of Sims [1994] on virtual creatures that discover ways of locomotion. Sims uses genetic algorithms — a class of evolutionary optimization methods inspired by natural selection — to discover the structure and morphology of virtual creatures, and to discover time-varying actuation forces that lead to crawling or jumping motions. Similarly, Lipson and Pollack [2000] present a complete pipeline for design, control, and fabrication of simple locomotive robots using bars and ball joints, with the aid of evolutionary search. This general approach has been successfully adapted to a wide range of additional animation domains, such as realistic control of swimming Tan et al. [2011], muscle-driven biped simulation Geijtenbeek et al. [2013], gait discovery for quadrupeds Lee et al. [2013], or learning bicycle stunts Tan et al. [2014]. Recently, deep learning has been applied to learn physics-based locomotion, Peng et al. [2017].
Translating virtual walk simulations into the real world is non-trivial. For humans and animals, hundreds of muscles have to act in unison through a central nervous system, for stable and efficient gaits. In a robot, the orchestration of actuators as muscles requires many sensors and a complex controller. In this context, Sims-like virtual characters are difficult to fabricate: even if one could find physical actuators and joints for all virtual motors, the resulting cost would exceed what is acceptable for most applications, especially for the simple automata that I consider. For toys and educational use, they have one motor per limb, no sensors, and no high-level controller; yet, they can walk successfully, once fabricated.

Walking Motions Control in Animation and Robotics A variety of advanced control methods have been proposed for antropomorphic physically-simulated humans Geijtenbeek et al. [2013], Lee et al. [2010] and animals Wampler and Popović [2009a], Coros et al. [2011]. Such methods have been applied to sophisticated legged robots to generate controllers Gehring et al. [2013], or to increase the agility of locomotion controllers Gehring et al. [2014]. However, complex control strategies require complicated mechanics, sensors, and actuators, and the StarlETH robot is well beyond the complexity and cost of my target of automata as toys. My designs (Chapter 3), are significantly simpler in nature, but are still able to perform walking motions. As such, the work is much closer to recent work in computational design than to the general field of robotics.

Coros et al. [2013a] note that even if the motion of a mechanical character at first glance resembles walking, this does not mean that the character would actually walk if fabricated. In initial experiments, I was not able to create any automata in this way that were capable of walking stably, unless I used a large number of legs (i.e., hexapod). This highlights the need for automated methods. To my knowledge, my work presented in Chapter 3, is the first to investigate the challenge of designing automata that can walk stably.
**Virtual Deformation Design:** Creating user-controllable deformation to create an articulate virtual character has seen many wonderful research contributions lately. Here, the developed methods are user-assisted, and semi-automatic for physically plausible articulation. Martin et al. [2011] and the references within create virtual example-based material deformations, where the user-provided shape forms are replicated under force application. These methods are created for physical plausibility, and cannot be applied for fabrication. Along similar lines, Coros et al. [2012] present a method for creating virtual deformable characters with toon-like articulation, where the secondary animations are automatically created. While Xu et al. [2015] create deformation models for elasticity of continuum’s material, and recently Hongyi Xu [2017] provide a method for controlling the damping behavior for materials undergoing deformation. Such works are differentiated from the task of rigging-based (skinning) deformation Bharaj et al. [2012], Baran and Popović [2007], where the deformation is based on non-physical deformation energy.

**Extreme Mechanics:** Extreme mechanics is a sub-field of continuum mechanics, where large deformations of elastic materials and shapes are explored for controllable deformation. Bertoldi et al. [2010] create negative Poisson-ratio structures (which expand when compressed) by using the buckling principle of deformation. Yang et al. [2015] take this further and create movements such as rotation, extension, etc for pneumatically actuated soft-robots. These methods however, do not provide a design tool for creating example-based extreme deformations and are limited to the premeditated deformation types. Greater emphasis has been laid on understanding the real life properties of these deformations: for example, Marchese et al. [2016] use vision-based data-driven methods to create soft gripper, where system equations are learnt for predictable deformations. Complete understanding of extremely soft and compliant deformable materials is still a research question and hence using soft materials with nonlinear deformation behaviors
can prove tedious and unpredictable especially for large deformations. This led us to the use Hook’s springs for extreme deformation Chapter 4.

**Mechanism Design:** Coros et al. [2013b] developed an algorithm for design of target curve-based linkage character, while I extend these to create walkable and statically stable robots. Zheng et al. [2016] use scissor linkages for creating shape-shifting characters. They do not optimize for static stability; however, the optimized linkage-based shapes can be packed and unpacked without collisions. Gauge et al. [2014] similarly create characters connected by elastic wires. Culpepper and Anderson [2004] create a simple shape shifter, where the emphasis is on the design of the mechanism for bistable shapes. Ou et al. [2016] create pneumatically actuated shape shifters. In these works, the amount of deformation remains small, and the overall forms of the shape remain the same. Ion et al. [2017] use a bistable mechanics primitive to create programmable logic gates such as AND, OR, XOR, and show how these simple physical computing logics can be exploited.

**Stability Optimization:** Most computational design methods strive to create controllable shapes. The notation of static stability is to create an object that is true to its shape under external forces and perturbations. The notion of stability comes up in many forms over an array of research works. Chen et al. [2014] optimize material properties for creating fabricable stable structures that have been optimized to hold a single shape under gravity or preset forces. Garg et al. [2014] create wire-mesh designs, with the notion of a stable shape under gravity; similarly, Zehnder et al. [2016] and Miguel et al. [2016] create intricate object designs such that objects can hold their forms for a single form. Moreover, there is no notion of bistable structures or morphing (Chapter 4). Panetta et al. [2015] create deformable structures that deform under constant external force loads (other than gravity) to create different shapes. The deformations are small and not statically stable without the
constant external force loads.

### 2.3 Acoustics and Contact Sounds

A few works in graphics are closely related to my own, Chapter 5. The example-based synthesis work of Ren et al. [2013] matches an object’s sound spectrum to a recorded sound using modal analysis and a Nelder-Mead optimizer. This transfers recorded sounds to a virtual object by finding optimal material parameters. Throughout, the object geometry is given and is unchanged.

Umetani et al. [2010] produced the first interactive design tool for forward metallophone fabrication. My method solves the inverse problem non-interactively (Chapter 5), affording three advantages: First, my method can optimize full 3D shapes as opposed to just 2D thin plates and produces pieces with richer timbres. Second, my method controls multiple frequencies, not just the fundamental (lowest) frequency of vibration, and also controls the amplitude of each vibrational mode. This helps create and dampen overtones (i.e., vibration frequencies higher than the fundamental, which is an important part of the timbre of the sound). Third, with no real-time requirement, I use high-order finite elements for accurate simulation, which reduces the need for post-fabrication correction.

Concurrently to this work, Hafner et al. [2015] optimized object thickness to control the lowest vibration frequency of the shape. They control only the smallest non-zero modal eigenvalue, but not overtones or vibration amplitude. In a sense, their problem is a simplified special case of the problem I tackle herein.

**Acoustic Inverse Problem** Kac [1966] posed the isospectral shape question: “Can one hear the shape of a drum?”, or, Do there exist two distinct shapes of membranes that resonate at the same frequencies? While answered in the negative by Gordon et al. [1992a,b],
this famous question inspired many inverse acoustic works, given detected sound scattering patterns Angell et al. [1997], Feijőo et al. [2004] or room echoes Dokmanić et al. [2013] reconstruct the shape of the structures that affect sound propagation. My work is also inspired by Kac’s question, but has a very different problem formulation: the input specifies not only frequency values but also amplitudes, and my goal is to find a 3D shape composed of elastic materials. While not strictly an isospectral problem, these previous works provide insight into the difficulties I may encounter. Specifically, dense spectrum optimization of any shape would require a continuous space of isospectral shapes, something considered unlikely by Zelditch [2000]. I sidestep this by performing sparse spectrum control, leaving the rest of the frequency components uncontrolled, but forcing their amplitudes to zero. This opens up a continuous space of shapes for my optimization method to explore efficiently.

**Contact Sound Simulation:** Computer music has modeled percussive instruments such as drums Fontana and Rocchesso [1998] and Xylophones Essl and Cook [1999]. I exploit fast physics-based modal sound simulation from computer animation van den Doel and Pai [1998], O’Brien et al. [2002], Raghuvanshi and Lin [2006], James et al. [2006], Chadwick et al. [2009], Zheng and James [2010, 2011], Lloyd et al. [2011], Ren et al. [2013] and acoustics Chaigne and Doutaut [1997]. These rely on modal analysis De Poli et al. [1991] (§5.3.1), which is widely used in computational mechanics. I extend these methods to compute derivatives of vibration modes from eigenanalysis, with respect to shape parameters, for use in the new optimization method.

**Acoustics and Vibration in Engineering:** Recent works have used computational design to produce musical geometries such as a Saxophone or even speakers Diegel [2013], Ishiguro and Poupyrev [2014]. Engineering has explored non-interactive inverse shape
design for the vibrational modes or frequencies of an object. Some works Yoo et al. [2006], Yu et al. [2010] focus on specific geometries (such as rotating beams or thin plates), while others are only concerned with the frequency Choi and Kim [2006], Yamasaki et al. [2010] or the amplitude Yu et al. [2013] of the vibration modes. Other works focus on objects under harmonic loads Choi and Kim [2006] and are incompatible with the contact sound inverse shape design problem.

Some works optimize the geometry and topology of mechanical structures to control interaction with sound propagation, to reduce noise levels Marburg [2002], Dühring et al. [2008], Barbieri and Barbieri [2006] or improve sound quality Bängtsson et al. [2003], Wadbro and Berggren [2006]. These consider sound propagation through the wave equation or the Helmholtz equation. I focus on sound generation from modal vibrations of solid objects.

My work is distinguished by its ability to optimize both sound frequency and amplitude for arbitrary volumetric geometry under a general external load. In addition, I consider how the object is supported, and optimize a stand to further suppress unwanted vibration modes. Finally, none of these works validate their results with fabricated objects as I do.

**Material Properties:** Recent works find optimal material properties for deformation behavior under external forces for 3D fabrication Bickel et al. [2009] and animation control Lee and Lin [2012], Li et al. [2014], Xu et al. [2015]. Contact sounds are also affected by object material properties, where pleasing and repeatable sound requires stiffness for vibration. As changing metal properties within a single piece is difficult, I fix material parameters and focus on optimizing geometric shapes for sound control.

**Sound spectrum Optimization:** My inverse shape design problem for contact sounds takes the form of a non-convex optimization. Sampling-based schemes are commonly used
to solve such problems Hansen et al. [2003], Pettersson [2008], Bardenet and Kégl [2010], Snoek et al. [2012]. However, I propose a new shape optimization algorithm based on antithetical sampling Nagaraj [2014], called Latin Compliment Sampling, which leverages local search (akin to Wampler and Popović [2009b]) and provides probabilistic bounds on search space coverage. I show that my method outperforms popular alternatives for complex optimization problems.
Chapter 3

Walking Automata

Creating mechanical automata that can walk stably with pleasing manners is a challenging task that requires both skill and expertise. We propose to use computational design to offset the technical difficulties of this process. A simple drag-and-drop interface allows casual users to create personalized walking toys from a library of pre-defined template mechanisms. Provided with this input, our method leverages physical simulation and evolutionary optimization to refine the mechanical designs such that the resulting toys are able to walk. The optimization process is guided by an intuitive set of objectives that measure the quality of the walking motions. We demonstrate our approach on a set of simulated mechanical toys with different numbers of legs and various distinct gaits. Two fabricated prototypes showcase the feasibility of our designs.

Figure 3.1: Two walking automata designed with our system and the corresponding fabricated prototypes.
3.1 Introduction

Stories of walking automata date back to ancient Greece, with the Renaissance reviving the tradition and providing prominent historical examples: da Vinci’s mechanical Lion, from the 15th century, delighted audiences upon its rebuilding in 2009. Although toy stores abound with mass-produced walking automata, their design remains challenging. It is arguably for this reason that many commercial walking automata are based on a small number of template mechanisms such as the famous Klann or Jansen linkages. But, even in this restricted setting, the design problem is still far from obvious: in addition to satisfying kinematic requirements on individual mechanisms (i.e., producing desired end-effector motion), the designer must carefully select parameters for all mechanisms, including timing and inertia, to obtain stable walking motion. This task is challenging for experts, and so is often beyond the capabilities of average users.

Modern rapid and additive manufacturing techniques expand the scope of mechanical design and construction. Motivated by this tremendous progress, the graphics community has started to embrace the challenge of translating digital characters into physical artifacts.
Zhu et al. [2012], Coros et al. [2013a], Ceylan et al. [2013], Thomaszewski et al. [2014]. These methods can create virtual characters with a broad range of complex motions, such as an in-air mimic of a walking motion, but their fabricated physical counterparts have yet to demonstrate an ability to actually walk stably. This is because neither the mechanics nor the geometry of stable locomotion are considered during the design. Considering these constraints would allow users to explore the space of feasible linkages for stable walking, and so to more easily design walking automata.

Capitalizing on this work, our method allows users to intuitively create unique automata designs that walk stably once fabricated. First, we learn a space of linkage configurations which are likely to lead to stable walking. Then, the user designs the automata by placing 2–4 linkage templates onto a body at arbitrary positions. From this initialization, we optimize the overall mechanical structures of the automata, allowing them to automatically discover how to walk with the same intuitive set of objective functions. This approach integrates physics-based simulation of mechanical assemblies with an evolutionary optimization algorithm that is able to explore the complex design space of these structures. We demonstrate our approach with simulated designs, and we validate our simulations by fabricating two very different walking designs for a dog and a crab.

### 3.2 Overview

Each of our automatons has a body with 2–4 linkages attached. Each linkage belongs to a set of linkage classes, with each parameterizable by pin joints and timings (Fig. 3.3). The 12-16 kinematic parameters $p_{k,r}$ encode the location of each pin joint, and so define the mechanical configuration of the automata. Each linkage is assumed to be driven by a single servo motor with speed control, so the four timing parameters $p_t$ per timing mechanism are used to control the relative phase between different linkages by directly
specifying the phase profile functions of the virtual actuators. These linkages can be standard mechanisms, such as Klann or Jansen linkages, or designed automatically Coros et al. [2013a], Thomaszewski et al. [2014]. Our initial database of pre-configured linkages was created using the method of Coros et al. Coros et al. [2013a]. We show two initial linkage configurations in supplemental appendix B.

Initially, we provide a simple drag-and-drop design tool to the user, who chooses and places linkage classes at arbitrary positions on the body (Fig. 3.2a, and supplemental video). Following this, we employ a stochastic genetic algorithm (CMA) and a walking objective function (§3.3) to optimize the linkage instance parameters with respect to the body in a rigid-body physical simulation (Supplemental appendix A). The simulation measures the quality of walking motions as the parameters of the automata — the mechanical linkage structures — change. It also previews the physical prototype output.

Not all linkage parameterizations are valid configurations which will move or lead to smooth motion, and so to speed up this optimization, we precompute within linkage space a subset of good linkage configurations. This data-driven approach is a reparameterization equivalent to learning and navigating a manifold on the original high-dimensional space (§3.4).

Finally, we fabricate the optimal automata. The fabrication process is manual, with all kinematic parameters and motor controls output directly by the optimization process. Physical prototypes are manufactured using a laser cutter and plywood, with metal bolts for joints, and servo motors for drive (§3.5).

### 3.3 Quantifying Walking

Starting from the input configuration created by the user with our simple interface, which is unlikely to walk, we must automatically optimize the design to walk. Walking is
complex, requiring muscles (or motors and sensors) to act in unison for stable and efficient gaits. We assume arbitrary kinematic configurations, formed from high-dimensional parameterizations. Rather than attempting the very difficult task of explicitly applying locomotion control knowledge from robotics, instead we describe an intuitive set of sub-objectives, which allows for stable, efficient, and sometimes interesting gaits to arise. We define the walking objective as the weighted sum:
During optimization, for each explored set of parameters, i.e., each automaton configuration, we compute the mass and moment of inertia of each rigid body part, and then physically simulate the automaton. The simulation proceeds until a failure mode is encountered, or until a fixed amount of simulation time $T = 30s$ has elapsed. Simulation position and orientation states and derivatives, $s = (q, \dot{q})$, are recorded for each timestep $t$. The recorded simulation states are used to evaluate each individual sub-objective (below). Some parameter settings will result in infeasible mechanisms, so we monitor constraint values and terminate the simulation early if they exceed a threshold value.

**Distance:** We wish the mechanism to walk forwards, not on the spot, and so we measure the vector $d$ between the center of mass at the beginning and at the end of the simulation, and define $S_{\text{dist}}$ simply as $-v^T d$, where $v$ is a unit vector that points along the desired walking direction. The negative sign is used to promote walking longer distances, since we minimize the total score $S_{\text{total}}$.

**Upright:** In early experiments, the automaton would often perform a somersault, as this quickly increases the distance traveled. This is not successful walking, nor is it an incremental solution towards better walking. Worse, we found that the optimization could not recover from such local minima. To avoid these solutions altogether, we add a term which encourages the automaton to remain upright for as long as possible. We define $S_{\text{upright}} = (T - t)^2$, where $t$ is the time when the simulation is stopped. The simulation is stopped early (i.e., $t < T$) if parts of the automaton other than the feet or bottom of the body touch the ground.

$$S_{\text{total}} = \omega_d S_{\text{dist}} + \omega_t S_{\text{upright}} + \omega_s S_{\text{smooth}} + \omega_e S_{\text{effort}} + \omega_r S_{\text{regularizer}}$$ (3.1)
Smoothness: We penalize the acceleration of the center of mass at every timestep. The acceleration $a$ is estimated using finite differences on the generalized velocity vectors $\dot{q}$ at consecutive time steps. The penalty term is defined as $S_{\text{smooth}} = a' a$. Increasing the weight of this term leads to more conservative motions which we posit increases the likelihood of successful walking once fabricated.

Effort: While infeasible mechanisms (i.e., singular configurations) are pruned early on, this does not mean that all mechanisms are equally desirable. For instance, small moment arms require larger motor torques and result in large internal forces acting on the mechanical structures, increasing the likelihood of mechanical failure. Thomaszewski et al. [2014] analyzed the singular values of the constraint Jacobian $\partial C/\partial q$ (see Supplemental appendix A) as a continuous measure of the distance away from singular configurations. Since we physically simulate, we have direct access to the magnitudes of the forces needed to satisfy the joint and motor constraints (i.e., $\lambda$ in Supplemental appendix A, eq.1).

Therefore, we define $S_{\text{effort}} = \lambda^T \lambda$ and add it to $S_{\text{total}}$ to minimize the net internal forces acting throughout the mechanisms.

Regularizer: Since the input to our system consists of a user-created automaton, we would like to change the design as little as possible. Therefore, we introduce a simple regularizer for the kinematic parameters $p_k$: $S_{\text{regularizer}} = (p_k - p_k^0)^T (p_k - p_k^0)$, where $p_k^0$ represents the initial parameter values.

### 3.4 Optimization

The optimization problem introduced in the previous section is high-dimensional, non-linear, and non-smooth due to the unilateral and intermittent nature of the contact forces.
Consequently, gradient-based methods are ill-suited, and we resort to a derivative-free stochastic evolutionary optimization technique based on the Covariance Matrix Adaptation algorithm Hansen [2006]. Some works attempt to improve CMA by adding derivative-based local constraint satisfiers Wampler and Popović [2009a]; however, for our problem this would make little improvement as the spaces are not smooth.

CMA generates parameter samples according to an internal Gaussian distribution. After evaluating the objective value for each sampled point, the Gaussian distribution is updated, and the process repeats until convergence. However, as noted by Coros et. al. Coros et al. [2013a], the parameter spaces of mechanical linkages are highly non-linear and random sampling, even around valid configurations, can quickly lead to degenerate mechanism that grind to a halt in mid-step or cannot move at all. Our experiments confirmed these observations, indicating that only $\approx 30–40\%$ of the randomly generated samples correspond to valid configurations. To not waste computation time, it would be desirable to rule out such degenerate configurations even before running any simulations. One solution might be to first check all linkages of the automaton in isolation and reject samples for which at least one degenerate linkage was detected. However, while seemingly simple and efficient, this strategy would not allow CMA to adapt its internal distribution and ultimately prevent the algorithm from exploring more relevant regions of the parameter space (see also Xu et al. Xu et al. [2012]).

To alleviate this problem, we note that linkages can be analyzed for validity before optimization to learn valid configurations for walking. For example, an errant pin joint configuration which locks linkages is unlikely to lead to a successful walk. We would like to construct a reparameterization of the space of linkages such that (almost) all samples generated by CMA correspond to valid linkage configurations. Therefore, we compartmentalize the valid linkage kinematic parameter problem, and attempt to learn a function of valid kinematics parameters from which to draw more useful samples during
CMA. To this end, we turn to Gaussian Mixture Models.

### 3.4.1 Linkage Configuration Function Learning

**Learning Parametric Function**  We observe that there are many pockets of space in which valid samples exist. As such, we use a multivariate Gaussian mixture model (GMM) Bishop et al. [2006b] to create a parametric function over valid spaces:

$$f(x) = \sum_{m=1}^{M} \pi_m \mathcal{N}(x \mid \mu_m, \Sigma_m)$$  \hspace{1cm} (3.2)

Here, $0 \leq \pi_m \leq 1$ and $\sum_{m=1}^{M} \pi_m = 1$ are the mixing coefficients, $\mu_m$ is an $n$-dimensional mean vector, and $\Sigma_m$ is an $n \times n$ covariance matrix of the $m^{th}$ multivariate Gaussian in the mixture.

We want to model the space of valid configurations with multiple Gaussians in a mixture. Given a number of mixtures (which we discuss discovering later on), we introduce a random *latent* variable $C$ which assigns a linkage configuration to a Gaussian in the mixture. This is indicated by an $m$-dimensional binary variable where only one element is ever one, with all others zero (1-of-$M$ representation). $C$ is modeled as a distribution $p(C) = \prod_{m=1}^{M} \pi_m^{c_m}$ s.t. the marginal distribution over $C$ is given by the mixing coefficients, $p(c_m = 1) = \pi_m$. Random variable $K$ is a continuous and observed, and is given by a multivariate GMM, $\mathcal{N}(x \mid \mu_m, \Sigma_m)$, and models the kinematic parameters of the linkage configuration.

**Data Preprocessing**  For a linkage, each pin joint location, i.e., the kinematic parameter $p_k$ between two bars, can vary over the lengths of the connecting bars. We vary these points of location within $-\frac{1}{4} \leq p_k \leq \frac{1}{4}$, where $l$ is a vector of lengths of the corresponding bars in the linkage. Using these box limits, we draw samples $D$ from the space of pos-
sible linkage configurations using Latin-Hypercube Sampling McKay et al. [1979]. This sampling strategy uniformly subdivides the sample space and ensures that a sample is drawn from each division, thus sampling the whole space. Then, for each sample, we simulate the motion of the corresponding linkage by minimizing the constraint energy (see supplemental Appendix A) for pin-joints and motor-constraints. We sum the energies for each step of a full motion cycle and label a sample as valid, if the summed energy is below a given threshold value (we use $10^{-2}$).

**Parameter Learning**  Given sampled data $\mathcal{D}$, we would like to calculate Gaussian parameters $\Theta_m = \{ \mu_m, \Sigma_m \}$ and $\pi_m$ which best model it: we wish to maximize the likelihood of $\mathcal{D}$ given $\Theta_m, \pi_m$. Mathematically (in log space) this is defined as:

$$\sum_{i=1}^{N} \ln \left( \sum_{m=1}^{M} \pi_m N(d_i | \mu_k, \Sigma_k) \right)$$

(3.3)

Initially, the dimensionality of the 1-of-$M$ representation of $C$ is unknown, so we use Expectation Maximization Bishop et al. [2006b] to find the maximum likelihood estimation of the random variable parameters.

**Number of Mixtures & Over-Fitting**  The number of components in the mixture plays an important role in determining the effectiveness of the learned model. Depending on the dimensionality of the problem, using too few mixture components can lead to inaccurate model-parameters, while too many mixture components can lead to over-fitting. Hence, we use the Bayesian Information Criteria Schwarz et al. [1978] to determine the number of categories, i.e., the dimensionality of $C$, and thereby the number of Gaussians in the mixture. Similar to Chaudhuri et al. [2011], we penalize Equation 3.3 by $\frac{1}{2}M \log(|\mathcal{D}|)$, the BIC criteria, by sequentially varying the number of mixtures, and use the one that maximized the cost. For a given linkage, we found that 6 – 12 mixtures gave us good

29
results. For example for the linkage configuration given in Figure 3.4, we found that six components maximized the EM energy in Equation 3.3.

Another method that we employ to avoid over-fitting the data is Leave-p-out cross validation [Hastie et al. 2009], we perform this over multiple ps. This validation indicates how well the test samples fit the GMM model’s probability-density function. It is noted that full covariance matrices \( \Sigma_m \) can lead to data over-fitting, we experimented with only diagonal covariance matrices, when number of Gaussians was twice as large as the number of mixtures in the GMM. However, for our problem using 6 – 12 mixtures with full covariance matrices gave us the better results while still satisfying the above mentioned over-fitting checks. Finally, for a given linkage, we found that using \(|D| = 25,000 \) samples sufficed.

**Sampling by Inference**  We have learned a GMM over the space of kinematic parameters for a linkage, and now need a structured way to sample from the GMM for use in CMA. One advantage of this probabilistic approach is that the learned random variable parameters encode the joint probability distribution \( P(X) \), where \( X = \{C, K\} \) and factorizes as \( P(X) = P(K|C)P(C) \). Hence, this can be used to answer queries such as:

\[
p(x \mid c_m = 1) = \mathcal{N}_m(x \mid \Theta_m) \tag{3.4}
\]

More intuitively, this means that it is highly likely that a given linkage parameter \( x \) belongs to the valid space and is modeled by the \( m^{th} \) Gaussian (a conditional distribution). Using similar inference queries for all valid \( m \) mixtures, we precompute \( M \) multivariate distributions modeled by the GMM. This gives us the desired linkage-configuration parametric function (equation 3.2). Next, in an automaton, the kinematic parameters of a linkage \( p_k \)
Figure 3.4: We show a sampling of the learned linkage space, with examples from three different Gaussians in the mixture (color coded). As this is a high-dimensional space, we perform PCA on the learned space and show only the first two principal components.

are replaced by the following vector:

\[ p_k^* = [m, x] \]  

(3.5)

Here, \( m \) gives the mixture index and \( x \) (a bijective mapping to \( p_k \)) are the multivariate Gaussian function parameters. When the \( m^{th} \) index is selected, we set \( c_m = 1 \), which selects the corresponding precomputed Gaussian (Eq. 3.4). Since all the covariance matrices are symmetric positive definite, first we perform eigenvalue decomposition and extract the
eigenvectors \((\mathcal{E}^m)\) and eigenvalues \((\mathcal{A}^m)\) of \(\Sigma_m\). Then second, a sample is drawn as:

\[
p_k = \mu_m + (x\sqrt{A^m})\mathcal{E}^m
\]  

(3.6)

Since the mixture weights \(\pi_m\) give the probability that a sample from the valid configuration space belongs to Gaussian, it makes sense that Gaussians with higher probability are sampled more. Hence, the mixture-index parameters \(m\) are weighted by \(\pi_m\) and mapped. Figure 3.4 shows samples with random \(x\) inputs, drawn from three such Gaussians in the mixture.

**Optimization Summary** We precompute a database of linkage configurations, over which we learn an \(M\)-dimensional GMM. For each initial user design, we run CMA, but now in each iteration, CMA’s internal multivariate Gaussian is sampled giving \(p_k^*\). Then, the linkage parameters \(p_k\) are discovered using \(p_k^*\) and Equation 3.6. This process is transparent to any other workings of CMA, so is a drop-in replacement. With this procedure, the walking optimization can quickly move between different Gaussians and can explore valid configuration spaces quickly and effectively. This strategy of optimization can also be interpreted as a discrete (mixture-component selection)-to-continuous (Gaussian) sampling.

**3.5 Experiments**

We validate our CMA-GMM improvements over many trials, and design five walking automata, for two of which we also create physical prototypes. As suggested by Auger and Hansen [2012], we generate \(4 + [3 \times \log(N)]\) samples for every CMA iteration, where \(N\) is the number of parameters that are optimized in parallel. Statistics for all examples are listed in Table 3.1.
CMA vs. CMA-GMM  To validate the proposed alternative data-driven sampling strategy for our linkages, we ran eight trials for each optimization strategy for the Dog automaton. Figure 3.5 shows a comparison of the convergence rates for CMA and CMA with GMM linkage sampling. We see that, on average, CMA-GMM converges both faster and produces solutions with better scores—a trend that was confirmed in all examples that we considered.

Dog  Our first example is a quadrupedal automata with dog-like appearance. Its four legs consist of 6 bars each, with the two front and hind legs having the same mechanical structure. While each leg can have a different phase, we enforce a symmetry relation between left and right legs to reduce the complexity of the design. Accordingly, the set of parameters exposed to the optimization consist of 14 structural and 4 velocity profile
parameters for the two front and hind legs, as well as 4 phase offset, making for a total of 40 parameters. The initial design created by the user, and the starting configuration for the optimization, can be seen in Fig. 3.6a, left. While this starting configuration falls over immediately at the beginning of the simulation, our optimization automatically discovers a gait with a lowered center of mass (Fig. 3.6a, right), which increases stability and thus leads to a successful walking motion.

It is worth noting that the optimization of the structural parameters is essential for obtaining a successful gait: as can be seen in the accompanying video, optimizing only for the timing parameters, i.e., the 4 phase offsets and the 8 velocity profile parameters, leads to a very inefficient gait with the dog essentially moving in place.

**Lobster** Our second example, a lobster automata, consists of two legs with identical structure, each comprising 6 bars and 10 parameters. Again, the initial design, shown in Fig. 3.6b (a), is not able to walk. Optimizing for the full set of 20 structural and 10 timing

<table>
<thead>
<tr>
<th>Robot Name</th>
<th># Components</th>
<th>Timing Mechanisms</th>
<th># Params</th>
<th>CMA-GMM Iterations</th>
<th>Time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>34</td>
<td>4</td>
<td>20 (Purple) 40 (Blue)</td>
<td>450 500</td>
<td>3 4</td>
</tr>
<tr>
<td>Grandpa Bot</td>
<td>20</td>
<td>2</td>
<td>13 (Purple) 16 (Blue)</td>
<td>350 350</td>
<td>2 2</td>
</tr>
<tr>
<td>Lobster</td>
<td>18</td>
<td>2</td>
<td>17 (Purple)</td>
<td>450 2.5</td>
<td>2</td>
</tr>
<tr>
<td>Gorilla</td>
<td>38</td>
<td>3</td>
<td>12</td>
<td>300 2</td>
<td></td>
</tr>
<tr>
<td>Giraffe</td>
<td>38</td>
<td>4</td>
<td>36</td>
<td>550 4.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Statistics for all examples.
Dog Model: (a) initial configuration that is unable to walk; (b) optimized automata

Crab Model: a) the initial configuration is unable to walk; b) a dynamic walking motion is found if only the leg parameters are optimized; c) allowing the body to also change its shape, an alternate locomotion mode is discovered.

Figure 3.6: The two automata for which we fabricate results.

parameters leads to a walking motion, but the gait is very dynamic with the center of mass shifting back and forth (see accompanying video). While this is not a problem in simulation, the various imprecisions introduced through simplifying assumptions and manufacturing make such dynamic motions more susceptible to instabilities. To obtain a less dynamic gait, we increased the weights of the smoothness and effort objective and introduced the height of the body as an additional parameter for the optimization. As a result, the optimization discovered a new, less dynamic gait that uses the body as a third leg.

As an additional observation, we found that, when enforcing a symmetry relation
between the two legs, the resulting motion was inefficient, i.e., led to a slow propulsion of the center of mass. By allowing for an asymmetric structure, we invite configurations in which the hind leg pushes while the front leg pulls, leading to a more efficient gait.

**Grandpa-Bot, Gorilla, and Giraffe** We designed three additional automata in simulation to further explore the behavior of our method. Our Grandpa-Bot is based on an example used by Coros et al. Coros et al. [2013a]. Although conceived with the intent to walk, having no notion of the physics or geometry of locomotion, their kinematic design tool did not produce a walking motion for this character. While our initial design (Fig. 3.7a, top) was also not able to walk, the optimization discovered a parameter set that leads to a succesful and aesthetically pleasing gait (see Fig. 3.7a, bottom, and video).

The fourth example is our Gorilla-inspired autoamata (Fig 3.7b), we explore a three-legged design for which our optimization discovers a gait that alternates between swing-through and support for the outer legs and the middle leg, respectively.

Finally, we show our Giraffe-inspired automata. We explore how the optimization adapts to asymmetric design with a shifted center of mass. As can be seen in the video, with a long neck, the initial design for the automata falls over and hits the head (Fig. 3.7c, top). Our system is able to adapt the front linkages and make the automata walk with a smooth symmetric motion while balancing the heavy neck (Fig. 3.7c, bottom).

**Physical Prototypes** To validate the feasibility of our designs, we created physical prototypes for the Lobster and the Dog character, representative photographs of which are shown in Fig. 3.1. All body parts were laser-cut from a plywood material with 5mm thickness. We used metal bolts and quicklock rings for the pin joints as well as Dynamixel MX-64 servo motors and a Robotis CM-700 controller board for actuation. The resulting walking motion for these two prototypes can be observed in the accompanying video. Although our
—and especially the treatment of frictional contacts—is inevitably approximate, we observed agreement between the walking motion of the prototypes and their simulated counterparts. Finally, it should be noted that we show simulation results for geometrically more elaborate models to better portray the artistic intent of the characters. However, simplified geometries that are consistent with the fabricated prototypes were eventually used for optimization. While 3D printing the robots is technically possible (ignoring the motors), pin joints may break due to internal strains. Laser-cutting linkages and using metal pins proved to be sufficiently robust.

**Robustness**  To investigate the robustness and convergence properties of our method, we reran the optimization for the dog character with three slightly perturbed initial guesses (by 5% in relative magnitude). As can be seen in the accompanying video, each run converges to a different solution, although each one results in a successful walking motion. We conjecture that this behavior is owed to the complexity of the problem and the randomized nature of our evolutionary optimization scheme.

### 3.6 Limitations And Future Work

Our work successfully demonstrates that computational design can create walking automata from initial non-walking designs. A physics-based simulation framework is used in conjunction with an intuitive set of objectives to measure the quality of the walking motions as the mechanical structure of the automata is automatically adapted. The framework takes advantage of learned valid linkage spaces to explore the parameters more efficiently. However, there are limitations to our approach and these are only the first steps.

Due to the non-linear nature of the problem, we adopted a stochastic sampling strategy
Grandpa-Bot: A human-like automata that exploits a walker for additional support.

Gorilla: A three-legged design with each leg alternating between support and swing-through phase.

Giraffe: Heavy neck makes the optimization adapt the automata-linkages accordingly.

Figure 3.7: Various automata produced by our system. In red are initial configurations, while blue are optimized walking automata.
to search for a global minima. However, experiments suggest that the optimization landscape exhibits a multitude of local minima. Therefore, we would like to experiment with continuation methods that incrementally increase the difficulty of the problem to increase the chance of finding global optima. One way to do this, for instance, is to employ external hand-of-god forces that initially support the weight of the mechanism but are progressively made weaker until they eventually vanish altogether. This strategy is akin to a child learning to ride a bicycle, and could allow the optimization process to more thoroughly explore the parameter space, as fewer trials would lead to failures.

The complexity of the optimization leads to longer convergence rates, and so the user cannot directly interact with the system during the gait design and optimization process. Hence, we would also like to investigate methods that allow users to control the gaits at a finer level of granularity. This could be investigated with additional style objective terms, or by pre-computing a library of walking templates to use as a starting point for the optimization.

To decrease the discrepancy between simulated and fabricated motion, we could accurately estimate the force output of the motors, as well as the static and dynamic friction coefficients of the fabricated material and floor — currently these are generic. To improve motion quality in general, we could add on-board sensors and additional actuators to monitor and control the gait cycle dynamically as the robot moves across different surfaces.

Finally, our simulation has no knowledge of intra- or inter-linkage collisions, and so when fabricating the robots these conflicts are resolved manually by stacking linkage bars along the crank rotation axis. For example, the legs of our fabricated lobster are broader than those in the simulation. For novice users, these collision conflicts must be automatically resolved, and is an area of future work.
Chapter 4

Bistable planar structures

Extreme deformation can drastically morph a structure from one structural form into another. Programming such deformation properties into the structure is often challenging and in many cases an impossible task. The morphed forms do not hold and usually relapse to the original form, where the structure is in its lowest energy state. For example, a stick, when bent, resists its bent form and tends to go back to its initial straight form, where it holds the least amount of potential energy.

In this project, we present a computational design method which can create fabricable planar structure that can morph into two different bistable forms. Once the user provides the initial desired forms, the method automatically creates support structures (internal springs), such that, the structure can not only morph, but also hold the respective forms under external force application. We achieve this through an iterative nonlinear optimization strategy for shaping the potential energy of the structure in the two forms simultaneously. Our approach guarantees first and second-order stability with respect to the potential energy of the bistable structure.
4.1 Introduction

Controlling the morphing and stability properties of a structure under varied force application has been an active area of research in continuum mechanics, robotics and graphics communities. While in continuum mechanics, these deformations are often used to create functional and compliant objects, in robotics, morphed forms are used to create mechanisms, including soft-robots for safer human interaction applications. These morphed forms often have small deformations as compared to size of the initial form structure and do not have the stability guarantee when morphed. In computer graphics the emphasis is to create artist tools for extreme virtual deformations. While these methods work fluently for animators, they can not be employed for bistable morphable structures once fabricated. Other methods are limited to creating structures with a single stable form under external force applications e.g. gravitational forces.

Thus, an interesting question arises: Are there methods for creating two statically stable, morphable, and fabricable structural forms? One way to achieve extremely different forms for a single structure is deform it until it buckles and drastically changes its form. This notion of buckling of metallic beams has existed in the field of continuum mechanics for several decades, where the emphasis had been to avoid buckling of metallic structures.
At this point of buckling the structure would permanently deforms or damages. Bertoldi et al. [2010] showed the use of the buckling principle for extreme structural deformation of elastic structures. While there are research works where extreme deformation is exploited to create functional objects such as Yang et al. [2015], Overvelde et al. [2016], however, there are no computational methods for creating example-based morphable bistable structures under external forces.

The goal of this work is to create a structure with two different statically stable forms, we call these structures *Metamorphs*, due to their morphable properties. A simple hinged linkage structures with embedded springs is introduced. This hybrid hard-and-soft *springy-linkage* can undergo extreme deformations to morph into different forms. We achieve the stability guarantees by optimizing for first and second order stability of the structure’s potential energy, as explained in Section 4.2.

The proposed method can not only be used to create structure with varied forms, but also in the field of soft robotics for creating safe grippers, that can grip objects with various non-convex shapes, while not being specifically programmed for any particular shape. Other applications include creating bistable wing configuration of aeroplanes Thill et al. [2008]. Such plane’s wings can adapt to various turbulence conditions with increased efficiency or use different wing structural forms while take-off, landing and cruising. Finally, similar to satellite wings, that are packed according to origami principles Miura [1985], Metamorphs can be used in applications where a different packed and unpacked stable forms are needed.

Metamorphs also find use in animatroics. Artists create puppets and articulated character where a deforming structure can be used to express the various *moods* of a character, and in story-telling mediums such as pop-up books Li et al. [2010] by creating collapsible structure structures. Lastly, such a method may also be used to create shape shifting furniture and human-computer interaction devices Yao et al. [2013]. A single piece of
Figure 4.2: Overview: (a) The input curves of the proposed structures (b) Equivalent planar linkage structures, (c) iterative nonlinear optimization, (d) optimized bistable structure.

Metamorph furniture can be configured to take different functional forms, or folded into a space-saving form. For example, a structure can be used as a table or morphed into a stool or morphed into a compact form.

4.2 Overview

This section introduces the notion of stability and provides a general overview of our approach. Figure 4.2 summarizes the same.

Structure and kinematics The input to the method are two input curves used to create the forms that a linkage structure must morph into. We define our structures as rigid bars that are connected at designated end-points via hinge-joints. All forms of a structure share the same bar count and connectivity and are geometrically equivalent. Details on how the linkage-based shapes are created from input curves can be found Section 4.3.1. The user can also fix certain linkages as fixed if desired. Between the various bars, springs are added, these spring are used for to create stability for the two forms of the structure.
For the purposes of this work, all the bars and springs are planar while having a certain z-depth, hence the problem essentially simplifies to 2D. This simplification is done in-order to create a fabricable structure. Adding springs that lie on different planes in 3D would lead to self-intersections, and unfit for fabrication.

**Energy-based Stability**  Consider two bars connected at a hinge. Let the upper bar be fixed at the outer end. Next, we add a spring connecting the outer ends of the two bars. For a fixed rest-length of the spring, the *springy linkage* can take a particular kinematic form as shown in Figure 4.3 (a). The figure on the far right shows the plot of change in spring’s rest-length vs. the potential energy gained by the structure as a result of change in the spring’s length. When the spring is stretched the most, form (b), the springy linkage has the largest potential energy. At this stage (i.e. the point of bifurcation) the system can morph back into form (a) and with an equal probability morph into form (c), the second state at that the structure has the lowest potential energy. This phenomenon is also called *bistability*. The point of inflection in form (b) is also called point of bifurcation or buckling.

**First-Order Energy Stability**  Similar to the *First-order necessary conditions for optimality*, Wright and Nocedal [1999], first-order stability of potential energy $V(x)$ is the state of the structure where the gradient of the energy potential is zero, i.e. $\nabla_x V(x) = 0$. At this point, the rate of change of the energy in any direction (locally) is zero. Since rate of change of the potential represents the forces acting on the system, the first-order stability intuitively means that forces acting on the system balance out and total forces acting on the system are zero. For example, in the case above, at all three forms (a), (b) and (c) the forces balance out. However, in form (b), the system is not truly stable, as even a infinitesimal push will lead to the structure morphing into forms (a) or (c). This morph depends on the direction
of the push (force or torque). Hence, first-order stability is a necessary, but not sufficient condition for establishing stability.

**Second-Order Energy Stability**  If we want a truly stable structure, the it has to be a second-order stable structure. Similar to the Second-Order Optimality Conditions used in convex optimization, this condition suggests that second-order derivative of the energy should be greater than zero. For a \( n \)-dimensional structure, this suggests that the Hessian \( \mathcal{H} \) of the energy potential should be positive definite. That is, \( \mathcal{H}(x) = \nabla^2_x V(x) \succeq 0 \) where \( x \in \mathbb{R}^n \). In the example shown in Figure 4.3, while (a) and (c) satisfy the second-order stability condition, (b) does not. Thus by optimizing for first and second-order stability simultaneously, we can create a morphable structures with two stable forms.

To summarize, we start with a kinematically feasible initial structure (Section 4.3.1). That is, the linkage structures can morph into the two desired forms, while they may not achieve stability or retain these forms. These initial forms can be chains, or branched, or loopy structures. An example of the initial design is shown in Figure 4.4.
In order to model the physical energy of the linkage structure and joints, we then introduce in Section 4.3.2 a rigid-body framework. Since our problem is modeling static stability, we derive the system equations, Euler-Lagrange equations Goldstein [2011], such that the velocity terms are zero, that simplifies our formulation. We also introduce an iterative springs addition algorithm (Algorithm 4.4) to create statically stable structural forms. In Section 4.4, the notion of energy shaping for creating stable structures is detailed. The method guarantees second-order stability w.r.t energy potential, thereby making sure that the structural forms are stable, and robust against gravity and user employed forces.

Section 4.5 discusses the results. We fabricated some examples for validation, and show complex examples virtually. We also show examples with real world functional applications, for example, shape shifting wings. Finally, Section 4.6 presents the conclusion and a discussion of the limitations, and opportunities for future work. To summarize, our computational design method introduces the notion of second-order static stability for bistable structures with the following major contributions:

1. Novel computation design tool for morphable structures.
2. Novel optimization formulation for creating bistable structure forms via energy shaping.

### 4.3 Problem Formulation

#### 4.3.1 Structure from input curves

Input to the method are two 2D curves, such as bézier curves, these curves define the two forms for the proposed bistable structure. There are no restriction on convexity or continuity of curves. Both the curves are spatially normalized and translate so the
center of mass lies at the origin. Each form is defined using \( m \) rigid bars. For each curve corresponding \( m \) points on the curve are defined which serve as the correspondence points for an end of the \( m \) bars, with a prescribed bar length. Alternatively, the form curves are sampled into \( m \)-points which serve as corresponding input bar ends. Let us call the input form curves as \( C_i \) and the linkage structure forms as \( F_i \) for the \( i^{th} \) form. It it also assumed that each bar of \( F_i \) is connected to its immediate neighbor by a hinge connection (Section 4.3.2). We now define an optimization algorithm that is used to create form \( F_i \) which is as-close-as-possible to \( C_i \), while keeping the rigid bar assumption.

**Rigid Deformation**  A natural disposition is to form an deformation energy as described by Sorkine and Alexa [2007]. However, this method does not guarantee rigid-deformation, but only an approximation. Hence, we propose the following method.

The input structure form \( F_i \) is be deformed to match \( C_i \) while maintaining rigidity and hinge connectivity. That is, the degrees of freedom of deformation are the rotations of the bars of \( F_i \). One way to formulate this rigid-deformation energy is as follows. Let \( v_{F_i}^j \) be the \( j^{th} \) bar end on \( F_i \) and \( v_{C_i}^j \) corresponding sample on \( C_i \), as shown in Figure 4.4.

We want to minimize the distance between these two points while making sure the the neighboring linkages (due to the hinges), \( \{v_{j+1}^{F_i}, v_{j-1}^{F_i}\} \) are at a prescribed distance. This can be formulated as the following optimization:

\[
\arg \min_{\{v_j^{F_i}\}} \frac{1}{2} |v_{F_i}^j - v_{C_i}^j|^2 \\
\text{s.t. } |v_{F_i}^j - v_{F_i}^{j+1}|^2 = c \\
|v_{F_i}^j - v_{F_i}^{j-1}|^2 = c \\
\forall j \in \{1, 2, ..., m\}
\]
Here $| . |_2^2$ represents the squared L2-norm. The vertex constraints given by equations 4.2 and 4.3 can be written in the matrix view as well. The above is a quadratic cost with quadratic constraints and can solved via a quadratic programming solver with quadratic constraints. We repeat this procedure for both forms of a Metamorph structure, such that each curve is sampled into $m$ samples, and hence all forms have the same geometry and kinematics.

### 4.3.2 Physical Modeling

Similar to Chapter 3, the numerical simulation of the linkage structures is modeled via rigid-body dynamics equations (Appendix 7.1). Two types of constraints used for modeling linkage structures are:

1. **Hinge/Pin Joints**: Constraints the position and two orthogonal rotational degrees of freedom at local points on two rigid-bodies such that the third rotation degree of freedom becomes the hinge-axis along which the bodies can rotate.

2. **Fixed Joints**: All positional and rotational degrees of freedom are fixed at a certain point on both bodies. The bodies are essentially locked and held fixed at predefined local points.
Constraint Jacobian  As derived in the Appendix 7.1 and Cline [2002], let $C(q) = 0$ be the satisfied, that is, let us assume that the kinematic constraints of the structure are always satisfied. Accumulation of all the constraints is represented by the constraint Jacobian $J$ matrix for the complete rigid-body assembly.

Spring model for a springy linkage  Given a rigid-body $B_i$, the world position $u^w_i$ of local point $u^l_i$ is given by $u^w_i(p_i, a_i) = R_i(a_i)u^l_i + p_i$. Here, $q_i = [p_i \ a_i]^T$ represent the kinematic degree of freedom of rigid-body $B_i$. $p_i$ is the translational and $a_i$ is the axis-angle, the rotational degree-of-freedom of the rigid-body. Then, the spring potential between a local points $u^l_1, u^l_2$ on $B_1, B_2$ respectively is given by:

$$V(x_{kin}) = \frac{1}{2}k(l - \sqrt{f(x_{kin})})^2$$  (4.4)

where $x_{kin} = \{p_1, a_1, p_2, a_2\}$  (4.5)

and $f(x) = g(x)^T g(x)$  (4.6)

$$g(x) = [R_1(a_1)u^l_1 + p_1 - R_2(a_2)u^l_2 - p_2]$$  (4.7)

Here $x_{design} = \{k, l\}$ is the spring stiffness and spring rest-lengths respectively for a spring. The gradients of these quantities with respect to the kinematic degrees of freedom, used to calculate the forces and torques are described in-detail in Appendix 7.2.

4.4 Optimization

Iterative spring addition

We propose the following iterative algorithm for adding springs to create the Meta-morphs with stable forms. Starting with the initial forms (without springs), the algorithm automatically calculates where the next spring should be added. The algorithm stops as
Algorithm 1 Iterative static stability optimization

```
while (converged == false) do
    \( b_j = \text{ModalAnalysis}\left(\{F_i\}\right) \) \hspace{1cm} \( \triangleright \) Calculate most deformed bar (Section 4.4.3)
    \( \text{AddMinEnergySpring}\left(\{F_i\}, b_j\right) \) \hspace{1cm} \( \triangleright \) Add spring on \( b_j \) to all forms (Section 4.4.4)
    converged = \text{StaticStabilityOptimization}\left(\{F_i\}, \{F^*_i\}\right) \hspace{1cm} \( \triangleright \) Run static-stability optimization (Section 4.4.2)
end while
F^*_i = \text{FabricationReform}(\{F^*_i\}) \hspace{1cm} \( \triangleright \) Spring fabrication reformulation (Section 4.4.5)
```

soon as static stability is achieved for both forms of the structure. As a result the algorithm adds an *optimal* number of springs to the structure which achieves bistability. While the details of each step of the algorithm are described in subsequent sections, Algorithm 1 details the outline.

Reduction

Starting from first principles with the force-acceleration rigid-body dynamics equations along with the added springs forces gives us the following equation:

\[
J^T\lambda + f_{\text{ext}} - \nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}}) = 0 \tag{4.8}
\]

\( s.t. \ C(q) = 0 \) (assumed to be true)

Here \( f_{\text{ext}} \) are external forces such as gravity, while

\[-\nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}})\] represents external forces due to spring potential \( V \). Here, \( J^T\lambda \) is due to principle of virtual-work. Equation 4.8 above has three kinds of degrees of freedom, namely,

1. Lagrange multipliers \( \lambda \).
2. $x_{\text{kin}}$ which are the position and orientation of rigid bodies.

3. $x_{\text{design}}$ which are spring stiffnesses (material parameters) and rest lengths.

In our formulation $x_{\text{kin}}$ are held fixed, by solving the problem in the null-space $N_{J^T}$ of the constraint Jacobian, we further reduces the complexity of the optimization. The updated formulation is shown below:

$$N_{J^T}(J^T\lambda + f_{\text{ext}} - \nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}})) = 0$$  \hspace{1cm} (4.9)

$$N_{J^T}(f_{\text{ext}} - \nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}})) = 0$$  \hspace{1cm} (4.10)

Equation 4.9, leads to Equation 4.10 as $N_{J^T}J^T = 0$. As a result there is a reduction in the degrees-of-freedom of $|\lambda|$, which is equal to the number of constraints rows added by a hinge/pin and fixed joints.

**First-Order Stability Condition**

For a statically stable system the primary requirement is first-order stability given by equation 4.10 above. Thus, we want to satisfy equation 4.10 for all forms $\mathcal{F}_i$ simultaneous. This equates to the following energy minimization problem:

$$\arg \min_{x_{\text{design}}} \frac{1}{2} |N_{J^T}(f_{\text{ext}} - \nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}}))|_{\mathcal{F}_i}^2$$  \hspace{1cm} (4.11)

$$\forall i \in \{1, 2\}$$

Thus, we want to reduce forces (gradient of the energy) acting in the two different forms, which is equivalent to first-order stability for constraint rigid body systems.
Second-Order Stability Condition

Not only do we want the forces acting on the structure in the two forms $F_i$ to balance out (equation 4.10), but also the structure must guarantee second order stability, such that under local perturbations, the structure returns to its stable forms $F_i$. This is guaranteed when the hessian of the energy potential is positive-definite as explained in Section 4.2. The potential of the spring structure is given by $V(x_{\text{kin}}, x_{\text{design}})$. In-order to guarantee second-order stability, we want the Hessian $\mathcal{H}(x_{\text{design}}) = \nabla^2_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}})$ to be positive-definite.

Once again we reduce the above to by projecting $\mathcal{H}$ in the null-space of the constraint-Jacobian, given by $\mathcal{H}_{N,J} = N^T_{J,t} \mathcal{H} N_{J,t}$. In-order to guarantee that $\mathcal{H}_{N,J}(x_{\text{design}})$ be positive-definite, we add non-linear constraints of the form $\mathcal{E}_j(\mathcal{H}_{N,J}) > 0$. Here, $\mathcal{E}_j(\mathcal{H}_{N,J})$ is the $j^{th}$ eigenvalue of the null projected energy hessian. Thus, for $i^{th}$ form we get constraints of the form:

$$\mathcal{E}_j(\mathcal{H}_{N,J}(x_{\text{design}}))^{F_i} > 0 \quad (4.12)$$

4.4.1 Minimal Potential Regularizer

We want to guide the optimization towards a lower energy potentials $P$ as high energy structures will wound too tight and bound to eventually snap. In case of springs a high potential configuration is the one in which the springs are stretched or compressed much beyond the rest lengths. Although the optimization can balance out the forces and torques caused by such springs, the springed structure can eventually snap. To alleviate this, we add the following regularizer to the optimization energy defined by equation 4.11:

$$w(V^{F_i}) \quad (4.13)$$
Here $w \in \mathbb{R}^1$ is the potential regularizer.

### 4.4.2 Two forms optimization

With all the ingredients defined in previous sections, we are now ready to describe the overall optimization strategy for the two forms optimization simultaneously. By combining equations 4.11, 4.12 and 4.13, we define the following nonlinear optimization problem:

$$
\arg\min_{x_{\text{design}}} \frac{1}{2} \|N_{J_i} (f_{\text{ext}} - \nabla_{x_{\text{kin}}} V(x_{\text{kin}}, x_{\text{design}})) \|_2^2 + w(V(x_{\text{design}}))^{R_i}
$$

s.t. $E_j(H_{N:j}(x_{\text{design}}))^{R_i} > 0$  

$$
lb \leq x_{\text{design}} \leq ub$$

$\forall j \in \{1, \ldots, m\}$

$\forall i \in \{1, 2\}$

Where there are $m$ eigenvalues per form. Thus, all the eigenvalue constraints are stacked together. All the design variables also have box-constraints over them which are needed for modeling physically correct ranges for spring parameters. In our case, we allow the spring rest-lengths to vary between 50% of the initial rest-length of the spring. $w$ is set to 0.001 for all examples described in the results section (Section 4.5).

The above is a nonlinear optimization problem with nonlinear and box-constraints. We employ the Augmented Lagrangian method (ALM), Wright and Nocedal [1999] to solve the same. A good refresher for the ALM method is also available in Narain et al. [2012]. We use the standard ALM method and BFGS line search strategy in the inner loop. The above optimization also requires gradients of the energy and the Jacobian of the nonlinear constraints. We use finite-difference method for calculating these quantities.
4.4.3 New spring addition – modal analysis

Given a structure with a given spring configuration, modal analysis (Kry et al. [2009], Bharaj et al. [2015b]) is a tool which can help calculate the deformation modes (via Eigenvalue analysis) of the structure. These modes (deformations) are the most likely changes in the structure as a result of excitation. A mode with the smallest non-positive eigenvalue given by equation 4.12 is the most likely to deform. If we were to run a forward simulation for a given shape with given spring configuration and rigid-body constraints, we would visually see such a deformation. Based of this observations we propose the following spring addition strategy.

**Method** For all forms we perform eigenvalue decomposition to calculate eigenvalues $E_j(H_{Nj}(x_{design}))^{F_i}$ and corresponding eigenvalues $e_j(H_{Nj}(x_{design}))^{F_i}$. Then the largest non-positive eigenvalue/vector pair is selected. Intuitively, the eigenvector represents the velocities (linear and angular) in the null-space of the constraint Jacobian. Therefore, we back-project $e_j^{F_i}$ by the following operation $E_j \times (N_{jF} e_j^{F_i})$ to calculate the velocities. Here, the unprojected velocity $N_{jF} e_j^{F_i}$ is multiplied with the corresponding $E_j$ to factor the intensity of negativity of the selected eigenmode.

Using these velocities the rigid-body system is forward simulated by a single time-step (using Symplectic Euler) to calculate the positional and rotational deformations of the rigid-bodies. Finally, we calculate the deformation of the various vertices on the structure and select the corresponding rigid bodies (those vertices which deform the most) as candidate bars. These bars are then used to add a spring according the formulation proposed below.
4.4.4 Minimal Energy Springs

A spring which has the same rest length on both forms will not increase the system potential, as a spring at rest-length does not add extra energy to the system. On form $\mathcal{F}_i$, a point on one of the bars is given by: $p = tp_1 + (1 - t)p_2$, where $t$ is the linear interpolation operator and $\{p_1, p_2\}$ are ends of the bar. Similarly, candidate point on another bar is given by $p_c = sp_{c1} + (1 - s)p_{c2}$. A distance metric between the two points $f_j \in \mathcal{F}_i$ is $d_1 = |p_c - p_1|^2_2$. Then for $f_j$ on $\{\mathcal{F}_i\}$, we want to minimize the following energy:

$$\arg\min_{\{s,t\}} \frac{1}{2} (d_i - d_k)^2$$  

(4.17)

s.t. $0 \leq s \leq 1$  

(4.18)

$0 \leq t \leq 1$  

(4.19)

$i, k \in \{1, 2\}$

Here same $\{s, t\} \in \mathcal{R}^1$ are used for both forms. By controlling the limits $s, t$ we can avoid adding duplicates for contiguous bars on $h_i$. Thus by adding box-constraints in equations 4.18 and 4.19, which are greater than zero and less than one, consistent springs
positions can be calculated. We employ BFGS with box-constraints to solve the above optimization.

4.4.5 Fabrication reformulation – z-depth arrangement

In the current rigid body formulation and optimization, self-intersection is not modeled, as a result, the springs are added in the same plane (assuming that all spring and bars are in the $x - y$ plane, with $z = 0$). This method works for simulation and optimization, but will lead to sever self-intersections in the fabricated design. To avoid this situation much like Coros et al. [2013a], Bharaj et al. [2015a], we add $z$-depth to each bar and spring such that, all each bar and spring has an unique $z$-depth. As a result, self-intersections are avoided and springs and bars of a structure can move freely and switch forms.

4.5 Fabrication and results

This section discusses the various metamorphs we optimized and fabricated. In-order to fabricate the spring model used for the numerical simulation, we created a spring assembly process. The fabricated springs match the deformation behavior and potential energy properties used in numerical simulation. We first discuss the details for the spring assembly and then present optimized results with virtual and fabricated validation, timings, and implementation details.

4.5.1 Fabrication and Calibration

Fabrication methodology The basic requirements of a spring are that the compression and elongation of the spring should happen in a straight line (in 3D). If we have a simple steel-wire spring eventually it will bend without internal support. However, the internal
support should not lead to change in spring’s stiffness or rest length properties. With these requirements in mind, we create an assembly process shown in Figure 4.6 to create a spring. Each spring consist of: a steel wire spring, an internal support (3D printed support with a hollow cylinder), and an internal support cylinder made of carbon-fiber. The cylinders are light weight and have very low coefficient of friction, as a results can slide very easily into the internal support and more importantly do not increases or decrease the spring’s stiffness. Finally, all parts are put together, and the ends of the spring are super-glued to the ends of the internal supports as shown in Figure 4.6 (right). This assembly process leads to springs which move in a straight line in any 3D orientation.
**Spring calibration**  Our spring have two properties, stiffness \((k)\) and rest length. In-order to have simple near linear spring stiffness, we choose McMaster-Carr’s Corrosion-Resistant Compression Spring Stock with 0.25” OD, 0.216” ID. This spring has a near linear spring stiffness. We then use the setup shown in Figure 4.7 to measure the actual spring stiffness. As shown, a stand is created to hold the spring vertically in place, and various weights are rested on top of the spring or hung from it. Then vernier calipers are used to measure the compression/elongation in the spring for the said weight. We use Hook’s spring formula \(F = kX\), where \(X\) is the amount by which the free end of the spring gets displaced from its rest length, \(F = mg\) is the force acting on the spring, \(m\) is the mass of the weight and \(g\) is the acceleration due to gravity. For each spring and for each weight, we measure three times for \(X\), and then use the average \(X\) (over all measurements). The correct \(k\) for the spring (for a given weight) is calculated accordingly. Since our springs are linear, we get the nearly the same \(k\) for each weight for compression and elongation.

### 4.5.2 Results

**Generic metamorphs**  Figure 4.1 shows an example of structure that can morph from a duck-like into teddy-like structural form. This structure consists of four bars that are connected by hinge connections (purple) and held at end-points (white). The iterative scheme optimizes for gravitational external forces so that both forms of the structure are statically stable. The optimized forms are then fabricated using the methodology described above and is shown in the corresponding figure with black bars. We also create a more complex form of the duck and teddy example as shown in Figure 4.5, this example shows that we can scale-up in complexity for a given Metamorph. Figure 4.8 shows our most complex example, where two drastically different input curves, turtle and elephant are optimized for bistability. For all the above examples we provide
Figure 4.7: Spring calibration: (a) Compression, and (b) elongation measurement setup. Various weights are either rested on top or hung-on to measure compression and elongation, respectively of the spring.

Figure 4.8: Higher Complexity: Turtle (Left) and Elephant (Right) are used to create complex forms for a structure.

virtual validation by running a forward rigid body dynamics simulation for both optimized stable forms. While for Figure 4.1 we show fabricated validation results. The convergence
timings and complexity for each example are shown in Table 4.1.

Functional metamorphs A natural consequence of bistability is that the same structure can change form. As a result it becomes useful for multiple use scenarios. For example, Chen et al. [2014] shows an example of a cloth hanger and a phone holder, where a single form is optimized for. We show a use case inspired from recent work in the field of robotics for bistable wings Manchester et al. [2017], where a linkage-based structure was used for a change in the wing orientation. Our method is flexible enough that we can not only change the orientation of the wing, but also its form. Figure 4.9 show the concept of a functional wing that can take two different forms and change the amount of air-drag acting on the wing. Such a bistable wing is useful in different scenarios such as, plane landing, perching, or cruising. By combining our approach with Umetani et al. [2014] we can create two wing designs for a single plane!

Posable metamorphs Creating virtual character with deformable articulate forms is now possible Martin et al. [2011]. There has been a push to achieve the same for fabriable characters such as Skouras et al. [2013a]. In such works although deformations are quite pronounced, the deformed forms are not stable in the second-order sense (Section 4.2) and need constant external forces to hold the forms.

We create a example-based poseable hand (Figure 4.10) that does not have these limi-


<table>
<thead>
<tr>
<th>Metamorph</th>
<th>No. Bars</th>
<th>No. Springs</th>
<th>Timing (mins.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck teddy, low-res.</td>
<td>6</td>
<td>4</td>
<td>0.08</td>
</tr>
<tr>
<td>Duck teddy, high-res.</td>
<td>14</td>
<td>21</td>
<td>13.16</td>
</tr>
<tr>
<td>Turtle elephant</td>
<td>24</td>
<td>47</td>
<td>21.15</td>
</tr>
<tr>
<td>Plane wings</td>
<td>9</td>
<td>14</td>
<td>0.48</td>
</tr>
<tr>
<td>Hand finger</td>
<td>4</td>
<td>6</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 4.1: Space and time complexity of various examples

This can not only be used as a gripper, but also with puppets and paper mache characters. The example shown consists of two finger and a thumb, where each finger is designed to have different stable forms. Such a setup can also be used for holding non-convex shapes, where the desired grip (finger forms) are input to the optimization method.

Implementation The rigid body, spring energy and optimization frameworks were written in C++ and run on Intel Quadcore CPU, on a single thread. Alglib a C++ library was used for Augmented Lagragian Method and BFGS. Press et al. [1996]’s method based on QR decomposition was used for Eigen value decomposition. Matlab’s syms package was used for calculating and testing against analytically calculated gradient and Jacobian of spring’s potential energy.

Table 4.1 shows the complexity – number of bars, and springs, and optimization convergence timings for each example discussed above. While the fabricated example shown in Figure 4.1 took about a day to fabricate and assemble.

4.6 Limitations and Future Work

We propose a novel computational design tool for creating bistable planar structures with second-order guarantees on the stability of each form. The iterative optimization uses modal analysis to first choose a location to add an internal support spring, then a nonlinear optimization is used to optimize for first and second-order energy stability. Such
structures can have drastically different forms. While the current method is promising, there are limitations of the method that can lead to future research. Because the input forms (curves) for a structure can be non-convex, newly added internal support springs may not stay inside the structure during the optimization. Because the algorithm tries to find a minimal energy springs (Section 4.4.4), it can result in springs remaining out of the convex form. In Figure 4.5, the springs near the beak of the duck remain outside the form. Another limitation of the method is that it solves for planar structures only. Although, switching to 3D springs would be easy we limit ourselves to planar structures due to fabrication constraint. That is, each spring must be free to move in a $z$-plane and avoid self-intersections. Note that this also true for all general linkages. We experimented with ball-and-socket joints, but again due to self-intersections chose to use planar hinge joints only.

As shown in Section 4.5, we add about $n$ springs for a $n$ degrees-of-freedom structural forms. Each spring is about 0.5 cms thick when fabricated, as a result if one were to fabricate such a structure, we’d go $0.5n$ cms deep along the $z$-axis. This can lead to extra torques/forces along the $z$-axis and is aesthetically displeasing. Hence, newer ways of

Figure 4.10: Posable Hand: Selected key-frames of character’s hand are used to create the hand model. Different fingers have different stable forms. This can be used for stop-motion character animation and as a gripper.
fabrication are needed that can lead to thinner internal support springs with reduced 
z-depth. And, facilitate with building complex bistable structures.
Chapter 5

Sound Spectrum Design

Metallophones such as glockenspiels produce sounds in response to contact. Building these instruments is a complicated process, limiting their shapes to well-understood designs such as bars. We automatically optimize the shape of arbitrary 2D and 3D objects through deformation and perforation to produce sounds when struck which match user-supplied frequency and amplitude spectra. This optimization requires navigating a complex energy landscape, for which we develop *Latin Complement Sampling* to both speed up finding minima and provide probabilistic bounds on landscape exploration. Our method produces instruments which perform similarly to those that have been professionally-manufactured, while also expanding the scope of shape and sound that can be realized, e.g., single object chords. Furthermore, we can optimize sound spectra to create overtones and to dampen specific frequencies. Thus our technique allows even novices to design metallophones with unique sound and appearance.
5.1 Introduction

Metallophone instruments produce sound directly by vibrating metal objects with a mallet strike. As such, the tone and timbre of this sound depend on the object shape and material. Over millennia, humans have developed design spaces of shapes which produce desired acoustic responses. However, discovering these spaces is expensive, time-consuming, and often requires trial and error. Even in cases where the design space is well known is additional expertise required. Take the glockenspiel: a set of suspended metal bars arranged as a keyboard which, when struck, produce a pure bell-like sound. Despite the fact that the size and shape of glockenspiel bars is well-studied, careful drilling of dimples on the underside of the bars is needed to tune the instrument. We explore whether we can use computation and digital fabrication to simplify the process of metallophone design.

Acoustic design has been heavily explored in mechanical engineering to avoid structural resonances, e.g., in beams or turbine blades. Algorithms attempt to optimize either the amplitude or the frequency response of an object, typically in response to an applied harmonic load. Such analysis has also been applied to musical instrument design to study the emission of sound from horns and the effect of violin bridges on tonal quality.
In contrast, we explore the problem of optimizing a given input geometry to control both the frequency and the amplitude of the vibrational response of metal objects to contact. We also exploit recent advances in automated manufacturing such as 3D printing, computer numerical control (CNC) milling, and water jet cutters to manufacture several novel struck metallophones which we compare to a professionally-manufactured glockenspiel. Simulation of contact sounds has long interested the graphics community, as has computational fabrication. We seek to bridge these two disciplines and explore how much control one can garner over the contact frequency spectra of complex geometries.

We tackle this problem using a new functional specification method which automates the design process and controls the acoustic response of a rigid object by optimizing its geometry. Given an initial parameterized shape and a sparse target frequency spectrum, we solve an inverse shape design problem using a global nonlinear optimization algorithm. During each step of the design space exploration, the current contact sound spectrum (computed using high-order finite element analysis) is compared to the user-desired frequencies and amplitudes. The search ceases once the two spectra are sufficiently close.

Significant challenges arise when trying to solve such optimization problems. The relationship between object geometry and intrinsic vibration modes is complex, and depends on both geometric features and material properties. As a result, we meet a constrained optimization problem that is high-dimensional and non-convex. We derive formulas for computing the derivatives of vibration modes with respect to the shape parameterization. These derivative formulas are general, allowing different shape parameterizations for different applications. Further, we develop a new Latin Complement Sampling search algorithm which provides probabilistic bounds on design landscape coverage. This allows us to produce metallophones of sufficient quality (0–2% error) to serve as useful instruments with novel two- and three-dimensional shapes.
**Figure 5.2: Overview.** Our algorithm for contact sound functional specification requires a shape, design parameters, material parameters and a frequency spectrum as inputs. We use a new global optimization technique, in conjunction with modal sound synthesis, to optimize shape parameters. Finally, we fabricate the computed shape (in this case, produced by a waterjet).

**Contributions**  Over existing works, we contribute:

- An algorithm to optimize the *entire* sound spectrum of a 3D object via shape variation. We use a new multi-objective formulation to optimize the frequency and amplitude of a struck linearly elastic object.
- *Latin Complement Sampling*: a hybrid search method tailored to non-convex constrained optimization, which provides probabilistic bounds on design landscape coverage.
- Analytical derivatives for sound frequency and sound amplitude with respect to arbitrary design parameters.
- A method for creating a stand for the fabricated object such that its sound quality is maximized.

**5.2 Rational and Overview**

**Motivation**  Acoustic design for instruments is concerned with frequency spectrum sparsity. For melodic instruments, most frequencies have negligible amplitude while a few are loud. These correspond to the note the instrument is to produce — its fundamental
frequency — plus its overtones, which together describe the timbre of the sound. Producing a ‘pure’ note requires the amplitude of the fundamental frequency to be high compared to that of the overtones, while the remainder of the frequencies should be damped as much as possible. These observations inform the algorithm described below, which performs user-specified frequency spectrum sparsification.

**Pipeline Overview**  Our approach needs five user-supplied components (Fig. 5.2, left):

1. A shape parameterization and a design map to specific design instances — the 3D geometry.
2. A set of design parameters which specify the starting shape.
3. The object material parameters.
4. A contact region and force for striking the object.
5. The desired spectrum when struck, as a set of frequencies and corresponding amplitudes.

Our methodology assumes that the shape parametrization and material properties are known a priori. To begin, the user provides a sparse desired frequency spectrum and defines the contact region by labelling vertices on the surface of the initial design instance. Given this, we use isotropic scaling and spectrum simulation of a strike to match the fundamental frequency of our object to the desired fundamental frequency in a least squares sense. Then, for the overtones, we again simulate, and establish correspondences between the simulated frequency/amplitude values and their counterparts in the desired spectrum using absolute frequency difference. These correspondences are used to compute the amplitude and frequency cost functions that measure spectrum similarity (Fig. 5.2: Input). Our supplementary video provides an animation of this initialization process. During optimization we maintain this mapping by re-associating each user frequency/amplitude with the nearest eigenmode using absolute frequency and amplitude distance.

Our method produces a design instance that, when fabricated and struck as specified,
produces a vibrational acoustic response with the desired frequencies and amplitudes. As our simulations assume vibration in free space — something that is impossible in the real world — we produce an optimized stand for each object which avoids damping user-desired frequencies and ensures the object can vibrate as needed (Fig. 5.2: Output).

We formulate this shape design task as a computational optimization problem (§5.3). The optimization variables are the shape parameters which describe a geometry instance in the design space. Fabrication limits such as object thickness and fabrication clearance impose constraints on these parameters. Complicating matters is the fact that the linear modal sound model (§5.3.1) imposes a highly nonlinear mapping between geometry shape and contact sound spectrum. This necessitates the development of a carefully-tailored solver which uses Complement Sampling to maximize its exploration of the design space. While searching the design space, we map the contact region from the initial design instance into world space using the shape parametrization, simulate contact spectra, and compared them to the user-supplied sparse goal spectrum. Once a design instance with sufficiently low error is located, the method terminates (Fig. 5.2: Optimization).

5.3 Problem Formulation

Formally, our mathematical derivation maps the shape design parameters to the resulting sound spectrum. To establish this map, we use the standard linear modal sound model Shabana [1995], O’Brien et al. [2002], James et al. [2006], Cook et al. [2007]. This choice is motivated by the model’s proven ability to accurately predict sounds of stiff materials such as metal.

We start by briefly reviewing the linear modal sound model that predicts sound spectrum from a provided shape (§5.3.1). Then, we formulate the core optimization problem (§5.3.2). This is solved using higher-order finite elements, for which we justify the need in
5.3.1 Background on Modal Vibrational Sound

A linear modal sound model is built using a finite element approximation. Given the 3D geometry of a design instance and its material parameters, we discretize its volume using a tetrahedral mesh Labelle and Shewchuk [2007]. Applying the finite element method leads to discrete equations of motion which govern the instance’s dynamic response to external forces:

\[ f, \]

where \( M, K, \) and \( D \) are respectively the mass, stiffness, and damping matrices depending on the object shape and material; the vector \( d \in \mathbb{R}^{3n} \) describes the finite element nodal displacement with \( n \) nodes (i.e., \( n \) tetrahedral mesh vertices); and the right-hand-side vector \( f \in \mathbb{R}^{3n} \) stacks the external forces applied on tetrahedral nodes to excite the vibration.

\( D \) affects how fast the sound damps out. For stiff materials such as metal, the damping \( D \) is often small — this is the reason we hear long ringing sounds from a metallic tuning fork. Since \( D \) has little influence on the vibration frequencies, we assume it to be zero. Further, since in our formulation, we consider impact sounds, \( f \) is treated as a Dirac delta function which models an instantaneous impact force applied to a prescribed contact region.

The standard methodology for solving Equation (5.1) is linear modal analysis Shabana [1995]. Provided the mass and stiffness matrices, we first solve a generalized eigenvalue problem:

\[ KU = MUS \quad \text{and} \quad U^T MU = I. \]  

This computes a modal shape matrix \( U \) and a diagonal eigenvalue matrix \( S \). The former describes displacement patterns of individual vibration modes while the latter describes...
the square of the vibration frequencies. In other words, if we apply a force impulse $f$ to an object, the resulting sound spectrum will consist of $N$ individual frequency components, for which the frequency values $\omega_i$ and amplitudes $a_i$ can be estimated using (Fig. 5.10):

$$\omega_i = \frac{1}{2\pi} \sqrt{S_{i,i}} \quad \text{and} \quad a_i = |f^T u_i|, \; i = 1...N; \quad (5.3)$$

where $u_i$ is the $i$-th column vector of $U$.

Now we can summarize our *forward* use of the modal sound model. Given the shape of an instance, and a known fabrication material, we construct mass and stiffness matrices using a finite element mesh, and solve the generalized eigenvalue problem (Equation (5.2)). Then, using Equation (5.3), we predict the frequency spectrum produced when striking the instance at a user-specified location. This chain of operations establishes a relationship between the shape of the instance and its produced sound spectrum. It is this relationship that we exploit to solve the *inverse* optimal shape design problem.

### 5.3.2 General Contact Sound Optimization Problem

**Notation** Let $p$ denote the geometric parameters for shape design (i.e., the design-space parameters), and $\phi(p)$ denote a map between the design parameters and the derived 3D geometry (a design instance). The specific choice of $\phi(p)$ is application specific and depends on the type of deformations a user wishes to allow. For instance, for radially symmetric objects the parameters $p$ are the 3D control points of a 1D curve and the $\phi$ is a rotational extrusion. To preserve this flexibility when presenting the problem definition here and in our optimization method (§5.4), we express $\phi(p)$ as a general map. We will explain specific parameterizations with our implementation details and experiments in §5.8. Further, we use $\omega_i(\phi(p))$ and $a_i(\phi(p))$ to denote the frequency and amplitude value of $i$-th component in the predicted sound spectrum, because as introduced in §5.3.1, both
\( \omega_i \) and \( a_i \) depend on the 3D geometry of the object via a generalized eigen-decomposition (Equation (5.3)).

**Objective Functions**  We wish to control \( \omega_i \) and \( a_i \) where the frequency values control the pitch of the sound and the amplitudes control the loudness of specific frequencies. We consider this problem as a multi-objective optimization with two subgoals:

- **Frequency Composition.** We allow the user to select a subset of all frequency components and control their frequency values. Let \( \mathcal{K}_f \) denote this subset, so \( k \in \mathcal{K}_f \) is an index of a frequency component that the user wishes to control. For every \( k \in \mathcal{K}_f \), the user specifies the desired frequency value \( \omega_k^* \). Then, we define an objective function:

\[
E_\omega(p) = \sum_{k \in \mathcal{K}_f} \frac{w_k}{\omega_k^*} [\omega_k(\phi(p)) - \omega_k^*]^2, \tag{5.4}
\]

where \( w_k \in \mathbb{R} \) is a user-controlled weight assigned to balance the relative importance among those desired frequencies (e.g., the fundamental frequency against any overtones).

- **Frequency Amplitudes.** As the input to our system includes an expected contact region, we can compute the volume of each frequency component using Eq. (5.3). In practice, the user specifies a small region on the object surface. When computing \( a_i \) in Equation (5.3), we construct a force vector \( f \) that is uniformly distributed over the user-specified region. The direction of the force at each surface vertex is chosen to be normal to the surface and the summed magnitude of all forces is chosen to be one. Note that changing the force magnitude scales all frequency component amplitudes by a constant factor making the whole sound louder or softer, but not effecting the relative amplitudes of said components.

We allow the user to select another subset, \( \mathcal{K}_a \), of frequency components, for which
the desired frequency amplitudes are specified. Here $K_a$ and $K_f$ can be independent (e.g., to dampen undesired frequencies). If $K_a$ is identical to $K_f$, then both the frequency values and amplitudes of those components are optimized. Let $a_k^*, \forall k \in K_{a},$ denote the user-desired frequency amplitude. Then, the objective function takes the form:

$$E_a(p) = \sum_{k \in K_a} \frac{w_k}{\bar{a}_1} \left[ a_k(\phi(p)) - a_k^* \right]^2,$$  \hspace{1cm} (5.5)

where $w_k$ is again the weights to balance the relative importance of the amplitudes, and $\bar{a}_1$ is the amplitude of the fundamental frequency of the initial shape; used to normalize the frequency amplitudes across multiple components.

**Multi-objective Optimization Problem**  We combine both functions (5.4) and (5.5) into a multi-objective optimization problem and solve with a lexicographic approach Branke et al. [2008]. Using multi-objective optimization allows us to explore the pareto frontier of optimal solutions without resorting to the non-intuitive weight twiddling associated with weighted sums of energy terms.

$$p^*_\omega = \arg\min_p E_\omega(p),$$ \hspace{1cm} (5.6)

where $E_\omega$ is defined in Eq. (5.4). Next, this result is used to initialize the amplitude control optimization which requires solving:

$$p^*_\alpha = \arg\min_p E_\alpha(p), \text{ s.t. } \omega_k(p) = \omega_k^*(p^*_\omega) \forall k \in K_f,$$ \hspace{1cm} (5.7)

where $E_\alpha$ is defined in Equation (5.5). Meanwhile, the shape parameters $p$ in both optimization problems need to satisfy certain fabricability constraints such as thickness and
Algorithm 2 Latin Complement Sampling

Sample \( N \) parameter values \( \mathcal{P} \) via Latin Hypercube Sampling 
\( t \leftarrow \) randomly choose one parameter sample 

\[ \text{for } i \leftarrow 1, m \text{ do} \] 
\[ \text{Repeat at most } m \text{ times} \]

Run local SQP starting from the parameter sample \( t \) 
Cache all \( p \) values gradients reached by SQP 
Reject unnecessary \( p \) values near local minima 
Fit a GMM to the cached samples 
\( t \leftarrow \) the sample in \( \mathcal{P} \) with the least GMM PDF value 
\[ \text{if the GMM PDF of } t \text{ is less than a threshold then} \]
\[ \text{return} \]
\[ \text{end if} \]
\[ \text{end for} \]

clearance limits. If there are \( N_L \) limits, we express them generally as:

\[
C_i(p) \geq 0, \quad i = 1 \cdots N_L,
\] (5.8)

and defer the discussion of their specific forms until §5.8.

Both objective functions (5.4) and (5.5) are highly nonlinear because of the complex dependence of \( \omega_k \) and \( a_k \) on the 3D shape \( \phi(p) \). There is no analytic expression for this dependence as it involves a high-dimensional generalized eigendecomposition (5.2). How can we tackle such an optimization problem numerically?

5.4 Optimization

To solve the above optimization problems, we propose a new optimization algorithm, Latin Complement Sampling (LCS), which better navigates complex energy landscapes and provides probabilistic bounds on its exploration (§5.4.2). To facilitate the computation, we further develop mathematical formulas for the derivatives of \( \omega_i \) and \( a_i \) with respect to \( p \) (§5.4.3), so that we can directly compute their values with respect to a variety of
parameterizations. We begin this section with a discussion of the rationale behind our algorithm before proceeding to the details.

5.4.1 Method Rationale

Observations  The particular form of our multi-objective cost function (Eq. (5.4) & (5.5)) depends on the specific choice of parameterization $p$ and design map $\phi(p)$. However, in general, they share the following properties:

1. The dependence of our cost functions on $p$ is continuous and differentiable, suggesting that a gradient-based approach could efficiently find local minima.
2. However, we have both linear and non-linear constraints. For instance, the constraints in Equation (5.7) are nonlinear.
3. Thus, the energy landscape is highly non-linear, with many local minima, cliffs from constraints, and large flat regions (Fig. 5.8). This suggests that globally a gradient-based approach would be easily confused.

These properties inform our choice of optimization scheme (see §5.7 for representative, low dimensional, example cost functions). Specifically, we seek a global optimization scheme that is compatible with the general constraints we encounter (Equation (5.7) and (5.8)); yet, we would like to exploit the differentiability of our energy functions to perform robust search for local minima.

Previous Strategies  Global nonlinear optimization requires exploring large portions of the energy landscape, typically relying on sampling schemes. Algorithms like CMA-ES Hansen et al. [2003] and Bayesian Optimization Snoek et al. [2012] are good examples of this approach. In general, such methods attempt to fit a function to previous samples and estimate where new minima may appear. Exploration is then carried out near these minima Pettersson [2008], Bardenet and Kégl [2010].
These algorithms have two issues in our context: First, it is difficult to include hard constraints in their formulations, and second they do not take advantage of available gradient information. Basin-CMA Wampler and Popović [2009b] attempts to fix these issues by performing a local Newton search at each sample point. This search serves as a projection operator onto the constraint set, as well as to move samples to local minima in the energy function. A Gaussian distribution is fit to the projected samples, then new samples are drawn from this distribution to continue exploring the energy landscape. However, as with previous works, it still relies on a sampling scheme which searches near anticipated minima.

In our case, this type of sampling is prone to failure as samples can fall into the first local minimum they find, leaving the remainder of the energy landscape unexplored (Fig. 5.6). Our solution to this problem is based on the antithetical sampling of Nagaraj Nagaraj
but we extend the method with a Newton search procedure noting that *samples generated by a Newton search already denote the best solution locally, thus new samples should be generated far from them.* We call our variant of antithetical sampling *complement sampling* because it amounts to fitting a distribution to sets of local samples and then drawing samples from its complement (Fig. 5.3). We do this efficiently using Latin sampling and a sorting procedure to avoid explicitly computing the compliment of the sampling function (as in Nagaraj [2014] for the purpose of analysis).

**Method Overview**  As outlined in Algorithm 2, our implementation of complement sampling, which we call Latin Complement Sampling (LCS), is divided into four phases:

1. Uniform sampling of the parameter space (Line 1 of Algorithm 2).
2. A local gradient-based minimization is performed from the chosen sample. The resulting locally-optimal parameter vector and its cost are cached (Line 4-5 of Algorithm 2).
3. Construction of a Gaussian mixture model using the cached parameter samples and their cost function values.
4. Selection of the next parameter sample using the above model (Line 7-8 of Algorithm 2).

The latter three steps are repeated until convergence or until a user-defined number of iterations is exceeded.

### 5.4.2 Latin Complement Sampling

**Sampling Parameter Space**  Suppose we are given a design-space parameterized by \( p \in \mathbb{R}^n \). These parameters need to satisfy box constraints \( L_i \leq p_i \leq U_i, i = 1 \ldots n \) (see specific examples in Section 5.7), where \( p_i \) is the \( i \)-th component of \( p \).

Let \( P = \{ p_1, p_2, \ldots, p_N \} \) be samples from the parameter space. These samples will serve as candidates for performing local gradient-based optimizations. Therefore, ideally
they need to explore the entire valid parameter space and sample uniformly. To this end, we construct \( \mathcal{P} \) using Latin Hypercube Sampling (LHS) McKay et al. [2000], a statistical approach known for its ability to spread sample points evenly across the sampled space.

**Local Gradient Step** We begin by selecting a parameter sample \( p_s \in \mathcal{P} \), and using it as the initial point from which we perform a local gradient-based search (either on function (5.6) or (5.7) depending on the stage of the optimization). These are nonlinear least-squares problems with nonlinear constraints. Therefore, we adopt standard sequential quadratic programming (SQP) methods Wright and Nocedal [1999].

The SQP method is itself an iterative algorithm, generating a new set of local samples \( \mathcal{L} = \{l_1, l_2, ..., l_M\} \) along a descent direction, terminating at a local minima. We cache samples \( l \in \mathcal{L} \) along with their associated cost functions values, filtering points that are too close together by Euclidean distance (threshold values range from \( 10^{-4} \) to \( 10^{-3} \)). These cached parameter points depict local regions that we have explored in the parameter space and we use them to inform our next choice of \( p_s \in \mathcal{P} \).

**Complement Search** Throughout this iterative process, we need to continually choose parameter samples \( p_s \in \mathcal{P} \). During the first iteration, we select the sample randomly (Line 2 of Algorithm 2). After the first iteration, we choose the sampled parameters such that they maximize our exploration of the parameter space. Concretely, we fit an \( n \)-dimensional Gaussian mixture model (GMM) Bishop et al. [2006a] to the cached parameter samples from all previous iterations. Conceptually, given a parameter value \( p \), the resulting GMM indicates the probability that \( p \) was explored by previous gradient-based local navigation. Namely:

\[
\text{prob}(p) = \frac{1}{(2\pi)^{n/2}} \sum_{i=1}^{M} \frac{w_i}{|\Sigma_i|^{1/2}} \exp \left[ -\frac{1}{2} (p - \mu_i)^T \Sigma_i^{-1} (p - \mu_i) \right].
\]
Here, the number $M$ of Gaussian components is determined by a KMeans++ clustering step Arthur and Vassilvitskii [2007] to cluster the cached parameter samples; $w_i$, $\mu_i$ and $\Sigma_i$ are also learned from the sampled parameters Dempster et al. [1977].

Next, we evaluate $\text{prob}(p)$ for all $p \in \mathcal{P}$ and find the parameter sample $\tilde{p}$ with the minimum GMM probability value. In other words, $\tilde{p}$ is the sample least likely to have been explored in previous iterations. Thus, we choose $\tilde{p}$ as the next starting point, $p_s$, of a local SQP solve, and move on to the next iteration of the algorithm. A sufficiently large value of $\text{prob}(\tilde{p})$ for all members of $\mathcal{P}$ (with a threshold of 0.3 in our examples) indicates that all the parameter samples have been covered, and so we terminate the algorithm (Line 9-10 of Alg. 2). The GMM model also provides us a probabilistic bound for the optimality of the given energy.

### 5.4.3 Cost Function Gradients

One piece of our algorithm that remains to be presented is the computation of cost function gradients with respect to the design-space parameter $p$. We could use finite differencing to compute gradients; however, this would require $2n$ cost function evaluations where $n = |p|$ is the number of design-space parameters. Since each cost function evaluation requires solving a generalized eigenvalue problem, this approach can be slow. Worse still, finite differences are known for suffering from inaccuracy when functions vary rapidly. As illustrated in Figure 5.8, such rapidly changing functions are indeed what we encounter.

Using the chain rule we can compute gradients of both the modal sound frequency and amplitude cost functions, with respect to $p$. The most challenging step is the computation of $\frac{\partial \omega_i}{\partial p_i}$ and $\frac{\partial a_i}{\partial p_i}$. We defer the detailed derivations to supplemental material, and present the
final formulas:

\[
\frac{\partial \omega_i}{\partial p_j} = \frac{1}{4\pi \sqrt{S_{i,i}}} u_i^T \left( \frac{\partial K}{\partial p_j} - S_{i,i} \frac{\partial M}{\partial p_j} \right) u_i \text{ and }
\]

\[
\frac{\partial a_i}{\partial p_j} = -f^T (K - S_{i,i} M)^+ \left( \frac{\partial K}{\partial p_j} - S_{i,i} \frac{\partial M}{\partial p_j} \right) u_i + \frac{1}{2} \left( u_i^T \frac{\partial M}{\partial p_j} u_i \right) f^T u_i,
\]

where, following the notation in §5.3.1, K and M are respectively the stiffness and mass matrices; \( S_{i,i} \) is the \( i \)-th eigenvalue; and \( u_i \) is the \( i \)-th column vector of the modal matrix \( U \). 
\((K - S_{i,i} M)^+\) is the pseudo-inverse of \( K - S_{i,i} M \).

5.5 Amplitude Control via Perforation

We observe experimentally that cutting small holes in an object has a negligible effect on its frequency response but a large effect on the amplitude of its vibration modes. This observation is of practical value as, in certain cases, it allows us to remove the constraints from our amplitude modulation cost function (Equation (5.7)), significantly simplifying it. We implement this by allowing \( \phi(p) \) to encode the position and edge length of small square perforations on the object surface. We draw inspiration from small indents drilled into glockenspiel keys to damp spurious vibrations. Because indents are difficult to drill accurately, we chose to use perforations (which can be cut or included in a 3D print) instead. Figure 5.4 shows how perforations can be used to adjust the amplitude of peaks in an object’s frequency spectrum.

5.6 Free Vibration with Stands

Our physical model assumes that the object itself is vibrating in free space, a condition impossible to replicate in the real-world. Anyone who has ever watched a drummer damp
a ringing cymbal with his hand knows that contact can greatly affect the qualities of sound due to vibration. To ensure optimal sound production from our manufactured pieces, we construct optimized stands to avoid damping as much as possible. This is comparable to the rubber, string, or felt rail underneath the nodal point of a glockenspiel bar. Our stand creation methodology has three steps: determining candidate supporting vertices, sorting vertices based on their potential to induce damping in desired or undesired frequencies, and selecting a concrete subset of these vertices to support the object.
We rely on the observation that a frequency is damped if its modal shape is not allowed to vibrate as if it were free. This tells us that ideal support vertices have small maximum displacements in all desired frequencies and large minimum displacements in all undesired frequencies. In our case, we find a reasonable set of support vertices using an efficient 1D search over two sorted lists of vertices: one which contains vertices sorted in ascending order of their normalized maximum displacement in user desired frequencies, and a second which contains vertices sorted in descending order of their normalized minimum displacement in undesired frequencies. We choose the smallest subset of vertices from the beginning of these two lists that make our metallophone statically stable. Please see our supplemental material for complete details of the algorithm.

**Stand Shape Creation** Once a supporting vertex set has been chosen, we create geometry for the stand using the CSG union of a flattened cuboid (the base) and upward pointing thin half-ellipses centered on each vertex. An example 3D-printed plastic stand is shown in Figure 5.5. The supports are covered with foam padding to avoid dampening from direct contact with the plastic. Stands are manufactured in the shape of the supported geometry to ease alignment and so clarify how each object is to be positioned.

### 5.7 Validating Latin Complement-Sampling

We compare our LCS algorithm to two methods commonly used in computer graphics for stochastic optimization and computational fabrication: Simulated Annealing (SA) Kirkpatrick et al. [1983] and Covariance Matrix Adaptation (CMA-ES) Hansen et al. [2003] (see Chen et al. [2013] for a review). To show the benefit of LCS, we also compare to Random Sampling (RSM) followed by local Newton’s search from each sample.

First, we examine the behavior of LCS, CMA-ES and Simulated Annealing on a 1D
non-convex cost function given by $f(x) = \sin(x) + \cos(x^2)$, over the interval $-4.5 \leq x \leq 4.5$ (Fig. 5.6(a)). We initialize all algorithms with the same parameter values and report the number of iterations required for each to converge to a minimum. Simulated Annealing is able to reach the global minimum in 700 iterations/function-evaluations while CMA-ES fails to find the same minimum as its initial samples fall into a nearby local minimum and are unable to escape. In contrast, LCS converges to the global minimum in 19 iterations (Fig. 5.6(b)).

Next, we show the behavior of all four algorithms on two well-known non-convex benchmark problems from the optimization literature: the Egg function and the Holder function (Fig. 5.7). We ran each algorithm 50 times, with random start points, and report the mean number of iterations until convergence to the global minimum. CMA did not converge (DNC) for either test while Simulated Annealing failed to converge on the Egg
Function evaluations

0
200
400
600
Function cost
-2
-1
0
1
2
SA
CMA
LCS
x \([-4.5, 4.5]\]
-5
-3
-1
1
3
5
f(x)
-2
-1
0
1
2
SA
CMA-ES
LCS

Figure 5.6: 1D function example comparing LCS to SA and CMA-ES. Top: Function and found minima. Bottom: Energy vs. function evaluation count.

Function. LCS and RSM converged to the global minimum in both cases. However, LCS requires on average half the iterations of RSM, showing that LCS explores the parameter space more efficiently.

2D Scaling Frequency Optimization

We explore the performance of each algorithm on a sound fabrication test. We perform a frequency optimization task (Eq. (5.4)) using an extruded toric geometry as input (Fig. 5.8). The goal of the task is to compute a geometry instance with a fundamental frequency of 261.6 Hz (Middle C on the piano keyboard). Our design parameters, \(p\), control scaling in the \(x\) and \(y\) directions and, are limited to \(0.5 \leq x, y \leq 1.5\) with simple box constraints (Fig. 5.8(a)). This produces a non-linear energy function landscape for this simple case (Fig. 5.8(b)). As above, we seed all algorithms with identical starting points. In this case, neither SA nor CMA-ES reach the global minimum while LCS finds a lower cost solution in fewer iterations. This is important because each function evaluation requires the solution of an expensive generalized eigenvalue problem (Tab. 5.1).
Table 5.1: The number of iterations, function evaluations, and the function cost of SA, CMA-ES, and LCS on a 2D toric acoustic shape design problem (Fig. 5.8), minimizing $|\omega^* - 261.6 H z|$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Iteration</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated Annealing</td>
<td>250</td>
<td>250</td>
<td>0.034</td>
</tr>
<tr>
<td>CMA-ES</td>
<td>34</td>
<td>206</td>
<td>0.040</td>
</tr>
<tr>
<td>LCS (finite diff.)</td>
<td>7</td>
<td>40</td>
<td>0.006</td>
</tr>
</tbody>
</table>

5.8 Metallophone Results

We use our parameterization-independent acoustic optimization to design 2D and 3D struck metallophones. For each result, we describe the input goal, the details of the parameterization used, and any extra constraints inserted into the optimization formulation. Finally, we show images and quantitative error (see supplemental document for measurement details). We urge the reader to listen to the accompanying video for qualitative assessment. Material parameters for each example are listed in the accompanying supplemental material.

5.8.1 The Zoolophone

The Zoolophone is a glockenspiel with the ordinary rectangular keys replaced with animal shapes.

Input and Output  For each Zoolophone key we specify a fundamental frequency (the note of the key) as well as several overtones. Our optimization seeks to match these notes while reducing the amplitudes of all other frequencies.
Parametrization for Frequency Optimization  Each Zoolophone piece is modeled using b-spline curves Piegl and Tiller [1997], the control points of which become our $p$ (Fig. 5.9a-b). We generate 3D geometry by extruding these 2D curves in $z$. If the number of control points in the initial shape is expressive, then we use a free-form-deformation grid to create a reduced space of shape parameters to make our optimization more efficient (Fig. 5.9c-d).

Parametrization for Amplitude Optimization  We perforate the object to control the amplitude of the frequency spectrum (§5.5). This is accomplished using standard CSG operations (Fig. 5.9(e)), with each perforation parameterized by location and radius. As mentioned previously, perforating an object allows us to simplify the amplitude modulation cost function. It also avoids excessive deformation of the geometry for the sake of amplitude control.

Fabricated Result  We create one key for each note in the C Major scale, namely the notes C, D, E, F, G, A, B, and C. Figure 5.9 shows the final shapes for each key, along with the user-defined frequency goals, the initial frequency spectra, the spectra computed using only linear scaling, the result of our method, and the spectra from a professionally-manufactured glockenspiel. To aid visual comparison, we overlay raw frequency plots with bar charts that highlight the seven largest peaks in the frequency spectra. Furthermore, Table 5.2 shows all frequency errors for these examples. In terms of fundamental frequency error, our algorithm outperforms isotropic scaling in all but one case for which we are within 0.2Hz. For overtones, we always outperform isotropic scaling, reducing error by $\approx 4 \times$ to $\approx 70 \times$ depending on the example. The main source of our error is due to fabrication: the waterjet used to manufacture the Zoolophone could chip pieces and had difficulty following non-smooth paths in the optimized geometry. A better understanding of the constraints of this fabrication method would improve our ability to constrain our
optimization, thus improving our results.

We also created several special zoolophone keys. The first two are elephants which demonstrated the ability of our algorithm to trade accuracy for shape preservation (Fig. 5.1: good shape and satisfactory acoustics; Fig. 5.9: more deformed shape (‘anteater’) but with better acoustics). See our supplemental video for a comparison. The next special key is a larger giraffe optimized to have a fundamental frequency at C4. Initial, isotropic scaling of the giraffe allowed us to match the fundamental frequency, but left a dense frequency spectrum (Fig. 5.12, bottom row). Our optimization scheme was able to significantly suppress unwanted frequencies, leading to a cleaner sound (Fig. 5.12, top row). Finally we demonstrate the ability of our method to control overtones, with a giraffe key that produces a chord when struck. This key has frequency peaks of equal amplitude at C, E, and G. We believe this is the first multi-tonal glockenspiel key ever produced (Fig. 5.4).

5.8.2 Tea for Three

Input and Output   Our method is not limited to producing 2D geometries. We perform frequency and amplitude modulation for bell-like cups, each optimized to produce a specific note, one of C, F or A. The cups were 3D printed by an online service.

Parametrization for Frequency Optimization   We use surfaces of revolution to create our cup shapes (Fig. 5.10, first row). Here, the shape parameters for frequency optimization are the control points of cubic b-spline curves. These control points determine the shape change of the cup, as shown by green dots in Figure 5.10. Augmenting the parameterization with a wall thickness parameters gives us printable 3D geometry.

Parametrization for Amplitude Optimization   As described in §5.5, we first decide where generally we would like to hit the object to produce the desired frequency. Figure
5.10(a) shows vertices where the force is applied. Similar to what is described in §5.8.1, in 3D we achieve amplitude modulation by adding holes to the cup geometry. To this end, we perform a difference operation between the cup geometry and thin pipe-like cuboids (Figs. 5.10(b) & 5.10(c)). Alternatively, we could add holes using cylinders; however, difference operation with cylinders leads to unnecessary high tetrahedral count, with no added advantage for modulation. Finally, the parameters for energy described by Equation (5.5) are the width, thickness, angle-of-rotation, and height of the cuboids. All parameters had box constraints.

**Fabricated Result**  Using the same initial parameter values, we optimize for three different frequencies: C 7, 8; F 7, 8; A 7, 8 (Fig. 5.1). As with all 2D pieces, we optimize for and fabricate stands for the cups. However, our C cup has no active vibration modes on its bottom surface and so does not require a stand. Of all our results, the cups have the most fabrication errors. Two of the returned cups had significantly different densities than what was specified by the manufacturer. Despite this, the final products have average error $\approx 1.5\%$.

**5.8.3 Cityscape**

**Input and Output.** Thus far, we have optimized 2D and 3D object frequency responses for a single contact location. Next, we attempt the more challenging problem of a piece which produces two different notes when struck in two different locations.

**Parametrization for Frequency Optimization.** Our design space is a grid of vertical cylinders. The design parameters, $p$, are the heights of the sixteen cylinders and the thickness of the base (Fig. 5.11(a)). The vertical distance between the top of the cylinder and the contact point remains constant.
Table 5.2: Frequency percent errors for fabricated zoolophone pieces: for the initial shape, our preliminary isotropic scaling to match just the fundamental frequency, and our full non-linear optimization. For the fundamental frequency, we include a comparison to the professional glockenspiel; however, beyond this, the overtones are not intended to match.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Note</th>
<th>Fundamental</th>
<th>Overtone 1</th>
<th>Overtone 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Initial</td>
<td>Iso-scale</td>
<td>Full</td>
</tr>
<tr>
<td>Giraffe</td>
<td>C 7, 8, 9</td>
<td>32.20 0.81 0.04</td>
<td>0.51</td>
<td>2.00 7.38 0.31</td>
</tr>
<tr>
<td>Whale</td>
<td>D 7, 8, 9</td>
<td>30.78 0.37 0.56</td>
<td>0.37</td>
<td>5.82 13.61 0.54</td>
</tr>
<tr>
<td>Elephant</td>
<td>E 7, 8, 9</td>
<td>32.84 0.87 0.26</td>
<td>0.22</td>
<td>15.37 17.63 3.29</td>
</tr>
<tr>
<td>Rhino</td>
<td>F 7, 8, 9</td>
<td>45.27 0.18 0.18</td>
<td>0.18</td>
<td>1.92 9.25 0.00</td>
</tr>
<tr>
<td>Lion</td>
<td>G 7, 8, 9</td>
<td>18.46 0.35 0</td>
<td>0.44</td>
<td>5.10 7.68 2.70</td>
</tr>
<tr>
<td>Tortoise</td>
<td>A 7, 8, 9</td>
<td>66.96 0.62 0.02</td>
<td>0.02</td>
<td>6.25 12.52 0.17</td>
</tr>
<tr>
<td>Bird</td>
<td>B 7, 8, 9</td>
<td>32.82 1.08 0.40</td>
<td>0.68</td>
<td>0.87 14.16 1.02</td>
</tr>
<tr>
<td>Giraffe</td>
<td>C 8, 9, 10</td>
<td>33.77 0.57 0.21</td>
<td>0.57</td>
<td>7.76 26.24 0.65</td>
</tr>
</tbody>
</table>

5.9 Conclusion

We have presented a functional specification algorithm for acoustic shape design. We have applied our new method to produce a variety of 2D and 3D metallic objects with user-defined impact frequency spectra. Using our techniques, new aesthetic-acoustic design is available for musicians, artists, and engineers to explore. If we consider that in the future our approaches could be applied to problems such as creating dampened operating mechanisms, then our work holds promise as a tool to help solve a broader class of sound control problems.

Limitations and Future Work The main limitation of our technique is that the achievable control for objects with dense frequency spectra is limited. The C4 giraffe (Fig. 5.12) shows that our algorithm can perform simple frequency and amplitude control under these conditions. However, it was impossible to optimize for more than one frequency given the dimensionality of our shape parameterization. A similar failure case is shown in Figure 5.13, where we are unable to achieve control over the dense spectrum of a triangle chime. As a rule of thumb, denser spectrum control appears to come at the expense
of geometry control, leading to undesirable results when our parameterizations are to restrictive. This restriction extends to overtone control: we were able to produce single key chords for C6E6G6 and C7E7G7, but not for C5E5G5 due to the increased spectrum density.

In some cases such as our elephant example, our algorithm could produce an acceptable acoustic result at the expense of unacceptable object geometry distortion. In these cases, adding additional user-defined shape constraints could help, though it is unclear how this would effect the fitted spectrum.

In general, choosing an appropriate parameterization is crucial for algorithm success since the parameterized geometry must be able to replicate the user-supplied frequency spectrum. Smart parameterization refinement algorithms, as well as fast methods for detecting “impossible” goals, are important open research questions.

Finally, our output is at the mercy of fabrication accuracy. Manufacturing our tea cups required multiple attempts due to inconsistencies in the 3D printing process which led to inconsistent material properties. Accounting for such variance during optimization is an important next step for fabrication algorithms. Further, while our method is rooted in the mechanical engineering approach of controlling specific object vibration modes, more perceptual metrics might allow us to avoid the mode matching cost functions and simplify the implementation.

**Acknowledgements**

We thank the anonymous reviewers for their feedback. A warm thank you to Sebastian Joseph, Emma Steinhardt, Henry Kavle, and Javier Ramos for fabricating the shapes, without which this paper would not be possible. Thank you to Joseph Booth for the robot work. This research was supported in part by the National Science Foundation:
CAREER-1453101, IIS-1116619, IIS 1447344; by the Air Force Research Laboratory and DARPA Memex program, and by generous gifts from Intel. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies or others.
Figure 5.7: A comparison of Simulated Annealing (SA), Covariance Matrix Adaptation (CMA), Random Sampling + Local Search (RSM), and Latin Compliment Sampling (LCS) using the well-known Egg and Holder function benchmark problems. Each algorithm was run 50 times with random start points. We record the mean number of iterations required to converge to the global minima. DNC denotes that the algorithm did not converge to the global minimum.
Figure 5.8: **Left:** Toric section free to scale in $x$ and $y$. **Right:** Energy over scale space showing non-linearities. The energy function is defined as $|\omega - 261.6\text{Hz}|$. 

![Diagram of Toric Section and Energy Over Scale Space]
Figure 5.9: The Zoolophone. Top: Frequency and amplitude parametrization: (a) Extrusion operation. (b) FFD parameterization (control points) and deformation. (c) Hole difference operation. (d) Amplitude-modulation parameterization. Middle: Shapes before and after optimization. The two ‘after’ elephants demonstrate a trade-off: (top) good shape with satisfactory acoustics, or (bottom) more deformed shape (‘anteater’) but with better acoustics. Bottom: Frequency response for each key in the zoolophone. We show four spectra for each key. The spectrum of the initial shape (dark blue), the spectrum after isotropic scaling to match the fundamental (light blue), the result of our method (green), and the spectrum of a professionally-manufactured glockenspiel (grey). The five largest frequency peaks for each spectrum are denoted by vertical bars. Goal notes are shown as dashed red lines along the ground plane and denoted using scientific pitch notation. Note that our goal is not to match the overtones of the professional key, only the fundamental frequency, and so please consider any match/non-match here to be a coincidence. The reader is encouraged to zoom in to examine each plot.
Figure 5.10: Tea cups example. Top: Shape parameterization as a surface of revolution (a). Amplitude-modulation perforation (b). Bottom: Frequency spectra for the initial cup shape (simulated) as well as the three manufactured cups. The two goal frequencies are shown in the top right of each plot in scientific pitch notation.
Figure 5.11: Cityscape. (a) Initial design space and its parameterization. The location-arrows represent the contact-force locations and directions. (b) Fabricated result.
Figure 5.12: We optimize a larger giraffe with a lower fundamental frequency (C4). Top: Isotropic scaling yields a dense spectrum of higher frequencies. Here, the amplitude of the fundamental frequency is diminished. Bottom: Our algorithm not only matches the fundamental frequency correctly at 261 Hz, but suppresses the amplitude of the remainder of the frequencies. The reader is encouraged to zoom in to examine both plots.
Figure 5.13: We attempt to optimize a standard triangle, which typically has a very dense spectrograph, to produce note C3 with overtones of C4 and C5. While some peaks are matched, even drastic shape change (left) cannot dampen all other peaks in frequency.
Chapter 6

Conclusion

Early seminal works such as Bickel et al. [2009] and Hašan et al. [2010] laid the foundations for computational methods for fabrication. Simultaneously, with the availability of cheaper but better quality 3D printers we are on the cusp of ubiquitous 3D printing for personalized needs, Baudisch [2016]. Most of these needs arise due to the functionality that a design has to offer. Although the capability to 3D print just about anything is growing by the day, we are still far away from creating designs that we can customize completely and adapt according to users’ needs. I foresee a future where designs are created with complete control over the specifications, functionality, efficiency, and ergonomics. This requires modeling the physical behavior for the design’s goal, and search strategies to find optimal solutions.

My thesis has presented methods that can help achieve the same for walking automata, bistable structures and contact sound spectrum design. While each method contributes a novelty for the design problem, there are many avenues for improvement and future goals that are worth pursing. I discuss these below.
6.1 Future Work

We live in a world involving simultaneous interaction with all the elements (earth, water, etc) simultaneously. Most of the designs of Nature interact with several elements at the same time. The same should also be true of our designs. The designs we create be functional not only for a specific task, but also functional under various interaction scenarios. Thus, there is a need for multi-physics computational design methods. While the thrust of the computational design community has been to solve a specific problem, a truly functional design should be the one that models multi-scenarios interactions. For example, what is the best design that works well on land and in air for a locomotive automata robot? Can we model both design problems simultaneously? This requires modeling the physical response for possible interaction scenarios and coming up with optimal designs.

For the walking automata project, there could be scenarios where the same automata walks over land as well as in a fluid medium. How can we achieve the this? Or can we create such automata at nanoscale (for example similar to Sato et al. [2017]) and use them for drug delivery or fighting anti-bodies? This requires modeling not only the walking dynamics, but also fluid interaction. Another example is bistable wings (Chapter 4). A major application of such a wing design is to change wing configurations under turbulence; such a wing could also be adapted to be part of an artificial fish (as its fin). In this case, we need to model not only the physical structure of the shape, but also its interaction with fluids (water or air) to make a multi-functional bistable wing/fin.

Recently, a great achievement of artificial intelligence and deep learning has been their ability to learn highly nonlinear functions from data. Most design problems eventually lead to search over complex nonlinear functions, while many corner cases are modeled separately. Computational design methods could also benefit from data learning. We saw an example of the same in Chapter 3, where a parametric function was learnt for
valid linkage configurations. Similar to Chaudhuri et al. [2011] for shape modeling and Peng et al. [2017] for character animation, we could learn a space of generic family of linkages for new linkage designs. Convolution neural networks (CNNs) Goodfellow et al. [2016] could be used to understand how various bars of a linkage should be connected together to create a new valid linkage design, along with its animation curves. Similarly, one could learn non-linear deformation behavior functions of materials by running costly FEM-based simulation. These learnt functions can then be used for fast, yet accurate material deformation modeling. This could, for example, be used to speed up sound spectrum calculations, among many other applications.

During the course of this thesis, I designed optimization methods for the said design problem. However, these optimization methodologies are quite generic and a deeper dive is needed to prove guaranteed bounds over their convergence. For example, Latin Complement Sampling Chapter 5, is a general optimization strategy for nonlinear optimization problems. The methods are similar to Bayesian optimization Snoek et al. [2012] in its notion to exploit and explore the energy landscape and outperform many state-of-the-art methods. However, a systematic study of its convergence, and proposed probability bounds is needed to get provable lower bounds. This problem is also related to entropy measurement of probability distributions.

Finally, much like Bermano et al. [2017], I propose a research direction for an exploratory look at computational functional design. Since similar problems occur in mechanical design, mechanics, robotics and graphics communities, there is a need for a unified look at methods proposed in all these fields. For example, bistable structures while having applications with respect to robotics, can also be used for creating space saving furniture, a push of the design community. Another example is the sounds spectrum project, where spectrum analysis was used for microscope creation Moore and Yong [2017]. An amalgamation of ideas from interdisciplinary fields can help move many communities
forward. Thus, a comprehensive effort aimed at surveying and creating a unified theory is the need of the hour.
Bibliography


Amit H Bermano, Thomas Funkhouser, and Szymon Rusinkiewicz. State of the art in


107


Zachary R Manchester, Jeffrey I Lipton, Robert J Wood, and Scott Kuindersma. A variable


Jesús Pérez, Miguel A Otaduy, and Bernhard Thomaszewski. Computational design and


Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross.


Kees van den Doel and Dinesh K Pai. The sounds of physical shapes. Presence: Teleoperators


Hongyi Xu, Yijing Li, Yong Chen, and Jernej Barbič. Interactive material design using model reduction. *ACM Trans. Graph.*, 34(2), March 2015.


Chapter 7

Appendix

7.1 Rigid Body Dynamics - Physical Simulation

Each automata is modeled as a rigid multi-body system. Since the mechanisms we optimize typically exhibit numerous kinematic loops, we opt for a maximal coordinates dynamics formulation. Therefore, the state of each rigid body $i$ consists of position and orientation degrees of freedom $q_i$, and their linear and angular velocity derivatives $\dot{q}_i$. The vectors $q$ and $\dot{q}$ concatenate the states of all rigid bodies in the system.

We model joints, virtual motors, and frictional contacts using a set of constraints of the form $C(q) = 0$, and their time derivatives $\dot{C}(q) = \dot{C}^d$ Cline and Pai [2003]. According to the principle of virtual work, the constraints give rise to internal forces $f_c = J^T \lambda$, where $J$ denotes the Jacobian $\frac{\partial C}{\partial q}$, and $\lambda$ are Lagrange multipliers that intuitively correspond to the magnitudes of the generalized forces needed to satisfy each constraint. To integrate the motion of the mechanisms forward in time, we must first compute the constraint forces $f_c$. Without loss of generality, we can express their magnitudes implicitly as:

$$\lambda = -k_p C(q_{t+1}) - k_d (\dot{C}(q_{t+1}) - \dot{C}^d)$$  \hspace{1cm} (7.1)
where subscript $t$ indicates the time instance, and the coefficients $k_p$ and $k_d$ allow us to set the relative stiffness of different types of constraints. A Taylor-series approximation of the position constraints allows us to express $C(q_{t+1})$ as:

$$C(q_t + h\dot{q}_t + hJ^T\dot{q}_{t+1}) = C(q_t) + hJ^T\dot{q}_{t+1}$$

(7.2)

where $h$ denotes the time step. Using the chain rule, the time-derivative of the constraints can be written as $\dot{C}(q_{t+1}) = J^T\dot{q}_{t+1}$. This allows us to approximate Eq. 7.1 as:

$$J\dot{q}_{t+1} = -a\lambda - ak_pC(q_t) + k_d\dot{C}$$

(7.3)

where $a = \frac{1}{hk_p + k_d}$. Using the equations of motion of the multi-body system, the generalized velocities $\dot{q}_{t+1}$ are given by:

$$\dot{q}_{t+1} = \dot{q}_t + hM^{-1}(F_{ext} + J^T\lambda)$$

(7.4)

where $M$ denotes the system’s mass matrix, and the term $F_{ext}$ stores the gravitational forces acting on the system. Multiplying Eq. 7.4 by $J$, and combining the result with Eq. 7.3, results in the following system of equations that is linear in $\lambda$:

$$A\lambda = b$$

(7.5)

where $A = hJM^{-1}J^T + aI$ and $b = k_d\dot{C} - ak_pC(q_t) - J\dot{q}_t - hJM^{-1}F_{ext}$. Because the constraint forces arising from frictional contacts are subject to inequality constraints, as discussed shortly, rather than solving Eq. 7.5 directly, we follow the work of Smith et
al. Smith et al. [2012] and compute $\lambda$ by solving a quadratic program:

$$\min \frac{1}{2} (A\lambda - b)^T (A\lambda - b) \text{s.t. } D\lambda \geq 0$$

(7.6)

where the matrix $D$ stores all the inequality constraints that need to be enforced. Once the constraint forces are computed, we use Eq. 7.4 to compute the generalized velocity term $q_{t+1}$, and the positional degrees of freedom $q_{t+1}$ are integrated forward in time as described by Witkin Witkin [2001].

The derivation we provide here is related to methods implemented by some modern rigid body engines, such as the Open Dynamics Engine Smith [2008]. However, rather than being restricted to working with ad-hoc parameters that hold little physical meaning, such as the Constraint Force Mixing term, Error Reduction Parameter and the Parameter Fudge Factor, we control the behavior of our simulations by manipulating the stiffness and damping parameters, $k_p$ and $k_d$, which are set independently for each constraint type (as detailed below). In the limit, as $k_p$ goes to infinity and $k_d$ to 0 (i.e., infinitely stiff spring), this formulation remains well-defined, and corresponds to solving the constraints exactly. However, from the point of view of numerical stability, it is often better to treat the constraints as stiff implicit penalty terms.

**Pin joints** that allow a pair of components to rotate relative to each other about a pre-specified axis are implemented using two sets of constraints. First, we ensure that the coordinates of the pin coincide in world space using a vector-valued constraint of the form $C(q) = x(q_i(t), p_i) - x(q_j(t), p_j)$. Here, $x(q_a, p) = t_a + R_a p$ corresponds to the world coordinates of the point $p$, $t_a \in \mathbb{R}^3$ is defined as the position of center of mass of rigid body $a$, and $R_a$ corresponds to its orientation. The location of the pin joint is defined by specifying the local coordinates of the pin, $p_i$ and $p_j$, in the coordinate frames of the
two rigid bodies $i$ and $j$ that are connected to each other. To ensure that the two rigid bodies rotate relative to each other only about the pre-scribed axis, we use an additional vector-valued constraint, $C(q) = R_i n_i - R_j n_j$, where $n_i$ and $n_j$ represent the coordinates of the rotation axis in the local coordinates of the two rigid bodies, and are set to $(0, 0, 1)^T$ for all our experiments. The $k_p$ and $k_d$ coefficients for the pin joint constraints are set to $10^8$ and $10^4$, respectively.

**Motor constraints** are used to mimic the effect of physical actuators. For this purpose, we prescribe the time-varying, desired relative angle between a select set of rigid body pairs. In particular, we assume that each limb of the mechanical toys has an input crank that operates relative to the main body. As we already employ pin joint constraints between these pairs of rigid bodies, the motor constraints directly measure the difference between their relative orientation and the target motor angle. The target motor angles are specified by phase profile functions $f(\alpha)$, as described by Coros et al. Coros et al. [2013a]. The desired value for the time derivative of the constraint, $\dot{C}$, is set to $\dot{f}(\alpha)$, and it intuitively corresponds to the target velocity of the virtual motor. The $k_p$ and $k_d$ coefficients for the motor constraints are set to $10^8$ and $10^5$, respectively.

**Frictional contacts** move our automata around their simulated environments, and friction and contact forces must be bounded to generate physically-plausible results. Each contact introduces three constraints. Let $n$ denote the contact normal. The first constraint specifies that the penetration distance, measured along the normal, should be 0: $C(q_a) = n^T(x(q_a, p) - x_p)$. Here, $p_a$ corresponds to the coordinates of the contact point in the frame of rigid body $a$, and $x_p$ is the projection of the contact point onto the environment. For this constraint, $k_p = 10^8$, $k_d = 10^4$, and, importantly, the constraint force magnitude is constrained to be positive: $\lambda_n \geq 0$. 

119
To model friction, we employ a pyramid approximation to the friction cone, as is standard in real-time simulation systems. More precisely, we let $t_1$ and $t_2$ be two orthogonal vectors that are tangent to the contact plane, and define constraints similar to the one for the normal direction, but acting along the tangent vectors. However, friction forces should only act to reduce the relative velocity at the contact point to 0. For this reason, we set $k_p$ to 0 for these constraints, while $k_d$ is set to $10^4$. To ensure that tangential forces remain within the friction pyramid, we add inequality constraints of the form $-\mu\lambda_n \leq \lambda_t \leq \mu\lambda_n$ for the magnitude of the tangential forces acting along $t_1$ and $t_2$, where $\mu$ represents the friction coefficient.

7.1.1 Linkage database

7.2 Axis-Angle representation

A circular movement of angle $\theta$ around a specified axis $\bar{v}$ in $\mathbb{R}^3$ is given by axis-angle:

$$v = \theta \bar{v}$$     \hspace{1cm} (7.7)
$$\theta = ||v||$$     \hspace{1cm} (7.8)
$$\bar{v} = \frac{v}{||v||}$$     \hspace{1cm} (7.9)

The rotation-matrix (from the axis-angle) is given by Euler-Rodrigues’s exponential coordinates (Murray et al. [1994], Page 29):

$$R = I + sin(\theta)[\bar{v}] \times + (1 - cos(\theta))[\bar{v}]^2 \times$$     \hspace{1cm} (7.10)
Figure 7.1: Initial linkage configuration used for the Dog mechanical automata. Linkage (a) is also the linkage used for all legs (front and rear) for the Giraffe-like automata.

\( \bar{v} \) is a unit-vector, so,

\[
[\bar{v}]_x^2 = \bar{v}\bar{v}^T - I \tag{7.11}
\]

\[
R = \cos(\theta) I + \sin(\theta)[\bar{v}]_x + (1 - \cos(\theta))\bar{v}\bar{v}^T \tag{7.12}
\]
Also, \([a]_x\) is a skew-symmetric matrix:

\[
[a]_x = \begin{pmatrix}
0 & -a_3 & a_2 \\
a_3 & 0 & -a_1 \\
-a_2 & a_1 & 0
\end{pmatrix} \in \text{Skew}_3 \tag{7.13}
\]

**Axis-Angle — Gradient**  Let \(u' = R(v)u\), then we need to calculate \(\frac{\partial u'}{\partial v_i}\). As \(u\) is independent of \(v\). We get the following (derivation in Gallego and Yezzi [2015], Appendix E):

\[
\frac{\partial u'}{\partial v_i} = \frac{\partial R(v)}{\partial v_i}u \tag{7.14}
\]

\[
\frac{\partial R}{\partial v_i} = \cos(\theta)v_i[\vec{v}]_x + \sin(\theta)v_i[\vec{v}]_x^2 + \frac{\sin(\theta)}{\theta}[e_i - \vec{v}_i\vec{v}]_x + \frac{1 - \cos(\theta)}{\theta}(e_i\vec{v}^T - \vec{v}_i e_i^T - 2\vec{v}_i\vec{v}\vec{v}^T) \tag{7.15}
\]

Note that, \(\frac{\partial u'}{\partial v_i}\) is a \([3 \times 1]\) column vector for \(v = \{v_1, v_2, v_3\}^T\). More compact gradient is given by, for example, Gallego and Yezzi [2015].

### 7.3 Spring — Potential, Gradients and Hessians

Given a rigid-bodies \(B_i\), the world position \(p_i\) of local point \(u_i\) is given by:

\[
p_i(c_i, v_i) = R_i(v_i)u_i + c_i \tag{7.16}
\]
Then, the spring potential between local points \( u_1, u_2 \) on \( B_1, B_2 \) respectively is given by:

\[
V(x) = \frac{1}{2} k (l - \sqrt{f(x)})^2 
\]

(7.17)

\[
x = \{ c_1, v_1, c_2, v_2 \} 
\]

(7.18)

\[
f(x) = g(x)^T g(x) 
\]

(7.19)

\[
g(x) = [R_1(v_1)u_1 + c_1 - R_2(v_2)u_2 - c_2] 
\]

(7.20)

For vector-spaces, we have the following property:

\[
\frac{d}{dx} (r(x) \cdot r(x)) = r'(x) \cdot r(x) + r(x) \cdot r'(x) 
\]

(7.21)

\[
\equiv 2 \ r(x) \cdot r'(x) 
\]

(7.22)
Spring — Gradient

\[
\frac{\partial V(x)}{\partial x_i} = -\frac{k}{2} \frac{l - \sqrt{f(x)}}{\sqrt{f(x)}} \frac{\partial f(x)}{\partial x_i}
\]  

(7.23)

Using 7.19 and 7.22, for \( j = \{1, 2, 3\} \)

\[
\frac{\partial V(x)}{\partial x_i} = k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) \frac{\partial g(x)^T g(x)}{\partial x_i}
\]

(7.24)

\[\equiv k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) \left( g(x)^T \frac{\partial g(x)}{\partial x_i} \right)\]

(7.25)

\[
\frac{\partial V(x)}{\partial c_{1j}} = k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) (g(x)^T e_j)
\]

(7.26)

\[
\frac{\partial V(x)}{\partial c_{2j}} = -k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) (g(x)^T e_j)
\]

(7.27)

Using 7.15,

\[
\frac{\partial V(x)}{\partial v_{1j}} = k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) \left( g(x)^T \frac{\partial R_1(v_1)}{\partial v_{1j}} u_1 \right)
\]

(7.28)

\[
\frac{\partial V(x)}{\partial v_{2j}} = -k \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) \left( g(x)^T \frac{\partial R_2(v_2)}{\partial v_{2j}} u_2 \right)
\]

(7.29)
Let \( h(x)_i = g(x)^T \frac{\partial g(x)}{\partial x_i} \). The hessian is a \( 12 \times 12 \) square-matrix, with \( \{x\}^{12\times1} \). Using equation 7.25, we get:

\[
\frac{\partial^2 V(x)}{\partial x_j \partial x_i} = k \frac{\partial}{\partial x_j} \left( \left( 1 - \frac{l}{\sqrt{g(x)^T g(x)}} \right) \left( g(x)^T \frac{\partial g(x)}{\partial x_i} \right) \right)
\]

\[
= k \left( \frac{\partial}{\partial x_j} \left( g(x)^T \frac{\partial g(x)}{\partial x_i} \right) \right) - l \frac{\partial}{\partial x_j} \left( g(x)^T \frac{\partial g(x)}{\partial x_i} \sqrt{g(x)^T g(x)} \right)
\]

\[
= k \left( \frac{\partial}{\partial x_j} h(x)_i - l \frac{\partial}{\partial x_j} \frac{h(x)_i}{\sqrt{g(x)^T g(x)}} \right)
\]

Using 7.22,

\[
\frac{\partial}{\partial x_j} h(x)_i = \frac{\partial g(x)^T \partial g(x)}{\partial x_j \partial x_i} + g(x)^T \frac{\partial^2 g(x)}{\partial x_j \partial x_i} \quad (7.33)
\]

Using chain-rule:

\[
\frac{\partial}{\partial x_j} \frac{h(x)_i}{\sqrt{g(x)^T g(x)}} = -\frac{h(x)_j h(x)_i}{2(g(x)^T g(x))^{3/2}} + \frac{\partial}{\partial x_j} \frac{h(x)_i}{\sqrt{g(x)^T g(x)}} \quad (7.34)
\]

Now, we need to define the \( \frac{\partial^2 g(x)}{\partial x_j \partial x_i} \) term in equation 7.33, rest are defined below:

\[
\frac{\partial g(x)}{\partial c_{1i}} = e_i \quad (7.35)
\]

\[
\frac{\partial g(x)}{\partial v_{1i}} = \frac{\partial R_1(v_1)}{\partial v_{1i}} u_1 \quad (7.36)
\]

\[
\frac{\partial g(x)}{\partial c_{2i}} = -e_i \quad (7.37)
\]

\[
\frac{\partial g(x)}{\partial v_{2i}} = -\frac{\partial R_2(v_2)}{\partial v_{2i}} u_2 \quad (7.38)
\]

With \( l, k = \{1, 2\} \), all terms of the form \( \frac{\partial^2 g(x)}{\partial c_j \partial c_k}, \frac{\partial^2 g(x)}{\partial c_j \partial v_k}, \frac{\partial^2 g(x)}{\partial v_j \partial c_k}, l \neq k \) are 0. We now need
Figure 7.2: A comparison of the accuracy of 1st and 3rd order tetrahedral finite elements for modal sound simulation for a shape. On the left is comparison of 1st, 3rd-order vs recorded data, while on the right is 1st and 3rd percentage error vs. recorded data.

to define the following:

\[
\frac{\partial^2 g(x)}{\partial v_{1j} v_{1i}} = \frac{\partial}{\partial v_{1j}} \left( \frac{\partial R_1(v_1)}{\partial v_{1i}} \right) u_1 \\
\frac{\partial^2 g(x)}{\partial v_{2j} v_{2i}} = \frac{\partial}{\partial v_{2j}} \left( \frac{\partial R_2(v_2)}{\partial v_{2i}} \right) u_2
\]

(7.39) (7.40)

7.4 Higher-Order FEM for Fabrication

Accurate simulation is critical for sound design problems. Even a frequency error of a few percent can impact the perceived quality of the results. No matter how robust our optimization scheme, it can be doomed to failure if simulation results do not match real-world outcomes. Many fabrication algorithms in computer graphics rely on linear (referring to the shape function order) tetrahedral finite elements to predict the physical behavior of design instances (e.g., in Bickel et al. Bickel et al. [2010]). However, such finite elements are well-known to be extremely inaccurate; worse yet, they do not approach the correct solution value even as the simulation discretization is refined Hughes [2012].
Table 7.1: Material properties and fabrication methods for all examples.

<table>
<thead>
<tr>
<th>Example</th>
<th>Metal-Name Name</th>
<th>Fabrication Method</th>
<th>Young’s Modulus (Pa)</th>
<th>Poisson’s Ratio</th>
<th>Density (Kg/m³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoolophone</td>
<td>Aluminum 6063-T83</td>
<td>Water-jetting</td>
<td>6.9e10</td>
<td>0.33</td>
<td>2700</td>
</tr>
<tr>
<td>City-Scale</td>
<td>Aluminum 6063-T83</td>
<td>CNC-Milling</td>
<td>6.9e10</td>
<td>0.33</td>
<td>2700</td>
</tr>
<tr>
<td>3D-Cups</td>
<td>Stainless-Steel 420 SS + Bronze</td>
<td>3D-Printing</td>
<td>1.48e11</td>
<td>0.32</td>
<td>8093</td>
</tr>
</tbody>
</table>

In this paper we rely on 3rd and 4th order finite elements via COMSOL Comsol [2005]. COMSOL also performs online remeshing in order to guarantee solution quality. However, even in the presence of this remeshing scheme experimental evidence (Figure 7.2) shows that the results produced by our algorithm would not be possible without higher-order finite elements.

7.5 Fabrication and Materials

With optimized geometry in hand, we turn our attention to fabrication. Most struck idiophones are fabricated in wood or metal, as plastics produce dull sounds. While wood produces a rich sound, especially with a resonating chamber as in a marimba, it is difficult to work with as the material density and stiffness vary between pieces due to structural differences (e.g., fiber arrangement and knots).

This leaves metal: Modern CNC tools can accurately and automatically reproduce our geometries. All 2D examples in this paper were produced by water jetting Aluminum 6063-T83. Our 3D examples were, geometry permitting, produced using CNC milling of Aluminum 6063-T83; otherwise, we relied on 3D printing from a commercial vendor Sha. Table 7.1, shows the material properties of all fabrication materials and outlines which process was used to manufacture each example.
Material Calibration  Correct simulation of sound spectra requires an accurate Young’s Modulus and Poisson’s Ratio. Shapeways’ metal 3D printing process uses the Stainless-Steel 420 SS + Bronze alloy. Though its material properties are documented online, we chose to validate the specifications ourselves. Inspired by Bickel et al. Bickel et al. [2009], we optimize for material properties so that the simulated frequency spectrum matches a recording of a real material sample (Figure 7.3).

We solve this inverse problem using our contact sound design method, with one change. Rather than choosing the design map parameters, $p$, to be geometry modifications or perforations, we choose them to be the Young’s Modulus and Poisson’s Ratio of the object’s material. Validation was performed by simulating the sound spectra of an unmeasured object and comparing it to a real-world recording. Finally, we compared our computed values against vendor-supplied material parameters, yielding an error of 1%. We found that our optimized material properties produced more accurate results than those provided by Shapeways, and therefore we used them for all relevant experiments.

7.6 Sound Measurement

This quantitative error must be measured, but this is complex: our optimizations are surrounding-environment-free, but our recording environment is not. One option would be to try to simulate our real world environment, as per O’Brien et al. [2002], but this is complex and error prone. Instead, we try to isolate our samples from the real world.
goal here is to prevent sound leaving the piece, reflecting in the real world, and returning distorted to the microphone (Samson Meteor). As such, we use a reflection filter as an isolation booth (Fig. 7.4), and surround it with dense fabrics. Finally, to reliably strike pieces with consistent force while inside the covered booth, we build a robotic mallet from a striking solenoid and an Arduino.

7.7 Derivatives of the General Eigenvalues

To aid explanation, we reproduce the key equations (2, 3, 4, and 5) from the main paper. The linear modal analysis equation:

\[ KU = MUS \quad \text{and} \quad U^T MU = I. \]  \hfill (7.41)
The frequency and amplitude estimation equation:

\[ \omega_i = \frac{1}{2\pi} \sqrt{S_{i,i}} \text{ and } a_i = |f^T u_i|, \quad i = 1...N, \quad (7.42) \]

The frequency composition objective function equation:

\[ E_\omega(p) = \sum_{k \in \mathcal{K}_f} \frac{w_k}{\omega_k^*} [\omega_k(\phi(p)) - \omega_k^*]^2, \quad (7.43) \]

The frequency amplitude objective function equation:

\[ E_a(p) = \sum_{k \in \mathcal{K}_a} \frac{w_k}{a_k^*} [a_k(\phi(p)) - a_k^*]^2, \quad (7.44) \]

Quasi-Newton methods such as the SQP require evaluating the derivative of the objective function, which in our case amounts to evaluating the derivative of eigenvalues with respect to each parameter. The derivative of Equation (7.43) over the \( j \)-th parameter is:

\[ \frac{\partial E_\omega}{\partial p_j} = \frac{1}{2\pi} \sum_{k \in \mathcal{K}_f} \frac{w_k}{\omega_k^*} \left( \frac{1}{2\pi} - \frac{\omega_k^*}{\sqrt{\lambda_k}} \right) \frac{\partial \lambda_k}{\partial p_j}, \quad (7.45) \]

where \( \lambda_k \) is the object’s \( k \)-th vibration mode (i.e., \( \lambda_k = S_{k,k} \) in Equation (7.42)). Calculating the derivative \( \frac{\partial \lambda_k(p_j)}{\partial p_j} \) for the \( k \)-th eigenvalue \( \lambda_k \) is nontrivial. Based on de Leeuw [2007], we take the derivative of \( K u_k = \lambda_k M u_k \), where \( u_k \) is the \( k \)-th eigenvector in \( U \) corresponding to \( \lambda_k \).

We obtain:

\[ \frac{\partial K}{\partial p_j} u_k + K \frac{\partial u_k}{\partial p_j} = \lambda_k M \frac{\partial u_k}{\partial p_j} + \lambda_k \frac{\partial M}{\partial p_j} u_k + \frac{\partial \lambda_k}{\partial p_j} M u_k. \quad (7.46) \]
Rearranging Equation (7.46) gives:

\[(K - \lambda_k M) \frac{\partial u_k}{\partial p_j} + (\frac{\partial K}{\partial p_j} - \lambda_k \frac{\partial M}{\partial p_j}) u_k = \frac{\partial \lambda_k}{\partial p_j} M u_k.\] (7.47)

Pre-multiplying both sides by \(u_i^T\) gives the derivative of eigenvalues:

\[\frac{\partial \lambda_k}{\partial p_j} = u_i^T (\frac{\partial K}{\partial p_j} - \lambda_k \frac{\partial M}{\partial p_j}) u_k.\] (7.48)

Here, the derivatives \(\frac{\partial K}{\partial p_j}\) and \(\frac{\partial M}{\partial p_j}\) of mass and stiffness matrices depend on the specific material models and shape parameterizations.

For simple examples such as those in §9, we are able to analytically compute the derivatives of \(K\) and \(M\) matrices using symbolical derivatives provided in such commercial packages as \texttt{Matlab} and \texttt{Mathematica}. For complex geometries or material models, we simply use finite difference to estimate the derivative values. As a result, this derivative formula is general to different parameterizations while retaining the efficiency.

### 7.8 Derivative of Generalized Eigenvectors

If the desired modal vibration amplitudes are \(a^*\), we need to solve the nonlinear optimization problem to minimize the energy function (7.44). The derivative of this energy is related to the derivative of generalized eigenvectors with respect to the shape parameters \(p\). In particular, we have:

\[\frac{\partial E_a}{\partial p_j} = 2 \sum_{k \in K_a} \frac{w_k}{\bar{a}_1} \left[ a_k(p) - a_k^* \right] f^T \frac{\partial u_k}{\partial p_j}.\]
Therefore, core to this computation is the evaluation of $\frac{\partial u_k}{\partial p_j}$. Similar to optimizing the eigenvalues, we start from Equation (7.47) and notice that:

$$K - \lambda_k M = U^{-T}(\Lambda - \lambda_k I)U^{-1}, \quad (7.49)$$

and its pseudo inverse (Moore-Penrose inverse) can be expressed as:

$$(K - \lambda_k M)^+ = U(\Lambda - \lambda_k I)^+ U^T. \quad (7.50)$$

Next, we notice that:

$$(K - \lambda_k M)^+(K - \lambda_k M) = I - u_k u_k^T M, \quad (7.51)$$

and also:

$$(K - \lambda_k M)^+ M u_k = (U(\Lambda - \lambda_k I)^+_k) = 0. \quad (7.52)$$

Pre-multiplying the pseudo inverse $(K - \lambda_k M)^+$ on both sides of Equation (7.47) yields:

$$(I - u_k u_k^T M) \frac{\partial u_k}{\partial p_j} = -(K - \lambda_k M)^+ \left(\frac{\partial K}{\partial p_j} - \lambda_k \frac{\partial M}{\partial p_j}\right) u_k. \quad (7.53)$$

Finally, differentiating $u_k^T M u_k = 1$, we have:

$$u_k^T \frac{\partial M}{\partial p_j} u_k = -2u_k^T M \frac{\partial u_k}{\partial p_j}. \quad (7.54)$$

Substituting this expression in Equation (7.53), we receive:

$$\frac{\partial u_k}{\partial p_j} = -(K - \lambda_k M)^+ \left(\frac{\partial K}{\partial p_j} - \lambda_k \frac{\partial M}{\partial p_j}\right) u_k + \frac{1}{2} (u_k^T \frac{\partial M}{\partial p_j} u_k) u_k. \quad (7.55)$$

Again, this formula involves the derivative of mass and stiffness matrices with respect to the parameters, as well as the pseudo-inverse of $(K - \lambda_k M)$. They all can be numerically
computed or analytically computed when symbolic derivatives are derivable.

7.9 Free Vibration with Stands

We detail how our stand creation algorithm chooses support vertices. The method has three steps: determining candidate supporting vertices, sorting vertices based on their potential to induce damping in desired or undesired frequencies and selecting a concrete subset of these vertices to support the object.

Support Locations We initialize the set of candidate vertices, \( V_c \), with all object vertices that are in contact with the ground when our object is in its playable orientation. We define the playable orientation as the orientation in which the object is upright and its contact patch is easily accessible. In our case, this playable orientation is known \textit{a priori}. Our goal is to select support vertices, \( V_s \), from \( V_c \) such that our object is stable and has optimal sound quality. We define stability in the traditional sense: the center of mass of the object and its contact patch are inside the convex hull of the support vertices. We optimize sound quality by minimally damping the desired vibrational frequencies while simultaneously attempting to maximally damp all undesired frequencies. In the main paper, Figure 5 shows an example of the optimal placement of support vertices.

\textit{A frequency is damped if its modal shape is not allowed to vibrate as if it were free.} This tells us that ideal support vertices have small maximum displacements in all desired frequencies and large minimum displacements in all undesired frequencies. In our case, we find a reasonable set of support vertices using an efficient 1D search.

Let \( S = \{ S^1, \ldots, S^M \} \), \( S^k \in \mathcal{R}^N \) be the set of object vibration modes (i.e., eigenvectors of Equation (7.41)). Here \( N \) is the number of vertices in the tetrahedral simulation mesh. We denote the displacement of the \( i^{th} \) candidate vertex in the \( k^{th} \) mode as \( s^k_i \in \mathcal{R}^3 \). Furthermore
let $J = \{1, \ldots, M\}$ be the indices of our user-defined frequencies. Our goal is to select $V_s$ such that $\max (s_j^i)$ is small and $\min (s_l^i)$ is large $\forall v_i \in V_s$, $\forall j \in J$ and $\forall l \notin J$.

We build two sorted lists of vertices, $F$ and $D$. $F$ is the free-vibration list. We insert all $v_i \in V_c$ into this list and sort them in ascending order of $\max \left( \frac{s_j^i}{s_j^{i*}} \right)$ where $s_j^{i*}$ is the maximum vertex displacement for the $j^{th}$ user-desired mode. Conversely $D$ is the damping list. Into this list we insert all $v_i \in V_c$ and sort them in descending order of $\min \left( \frac{s_l^i}{s_l^{l*}} \right)$ where $s_l^{l*}$ is the maximum vertex displacement for the $l^{th}$ undesired mode. We now filter $F$ and $D$ using a scalar threshold $t$, rejecting candidate vertices in $F$ with cost above $t$ as well as vertices in $D$ with cost below $t$. The remaining vertices form $V_s$. We perform a 1D binary search for $t$ such that $|V_s|$ is minimized and our stability criterion are met.